

機器學習於材料資訊的應用

Machine Learning on Material Informatics

陳南佑(NAN-YOW CHEN)

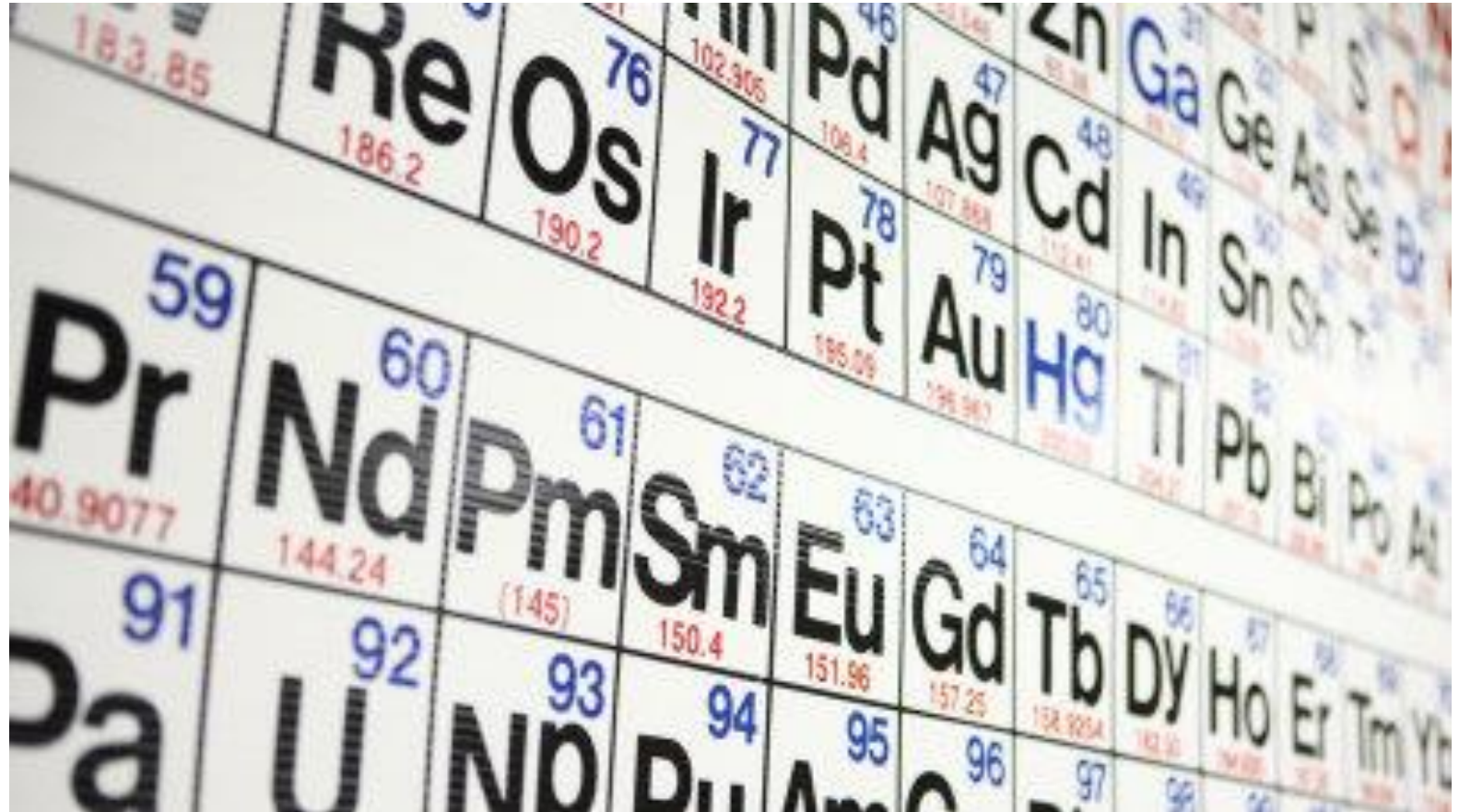
nanyow@narlabs.org.tw

楊安正(AN-CHENG YANG)

acyang@narlabs.org.tw

Material Properties Prediction

digging into the periodic table

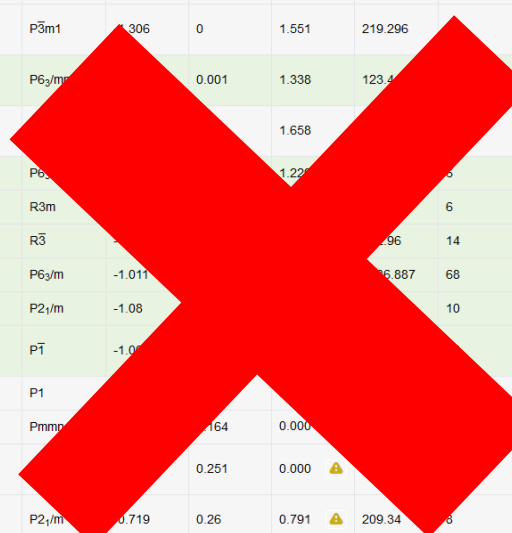


Problem Define

- 在早期，元素被發現的種類不多，化學家只能局部的對某些性質相近的元素進行歸類整理，例如1865年英國化學家伍德林（W. Woodling）按原子量排列元素順序，初步排出今日元素週期表中的鹵族、氮族、氧族。
- 俄國化學家門得列夫（Dmitri Ivanovich Mendeleev, 1834 – 1907）全面考慮了元素的各種性質，不僅根據元素的原子量，而且很重視元素的性質及其與其他元素的關係，他依原子量遞增的順序把元素排列成幾行，同時把各行中性質相似的元素左右對齊，這樣使得每一橫排化學元素的性質相近，每一縱列化學元素性質的變化也呈現着規律性，整個元素系列呈現出周期性變化。
資料的整理歸納
- 1869年2月，門得列夫發表了《元素性質和原子量的關係》論文，同時公布了他的第一張化學元素週期表，周期表中留下了四個空位，空位上沒有元素名稱，只有預計的原子量，表示尚待發現的元素。
新元素的預測
- 那化合物的特性能不能找出週期性？
- 可不可以從既有的材料資料庫預測出新材料的性質？

Get Data

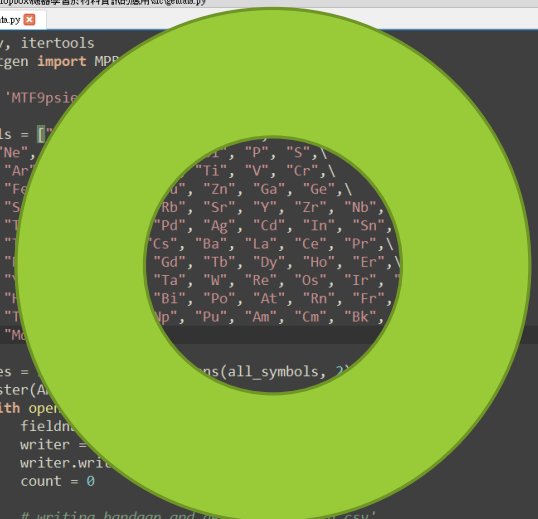
手動下載



Materials Id	Formula	Spacegroup	Formation Energy (eV)	E Above Hull (eV)	Band Gap (eV)	Volume	Nsites	Density (gm/cc)
mp-1027525	MoS ₂	P3m1	-1.306	0	1.746	350.447	12	3.034
mp-1025874	MoS ₂	P6m2	-1.306	0	1.783	284.872	9	2.799
mp-1023939	MoS ₂	P3m1	-1.306	0	1.551	219.296		2.424
mp-1018809	MoS ₂	P6 ₃ /mm2	-1.306	0.001	1.338	123.4		4.306
mp-1023924	MoS ₂				1.658			1.729
mp-2815	MoS ₂	P6 ₃ /mm2			1.22			4.503
mp-1434	MoS ₂	R3m				6		4.558
mp-2164	Mo ₃ S ₄	R3				14		5.062
mp-31257	Mo ₁₅ S ₁₉	P6 ₃ /m	-1.011			6.887	68	4.734
mp-1627	Mo ₂ S ₃	P2 ₁ /m	-1.08				10	5.702
mp-1104577	Mo ₃ S ₄	P1	-1.0					6.031
mp-673645	Mo ₇ S ₉	P1						5.674
mp-990083	MoS ₂	Pmm2		0.164	0.000			2.672
mp-1210708	Mo ₂₁ S ₈			0.251	0.000			8.75
mp-1239169	MoS ₃	P2 ₁ /m	0.719	0.26	0.791	209.34	8	3.048
mp-11780	MoS ₂	F43m	-1.041	0.265	0.000	221.127	12	4.808
mp-558544	MoS ₂	R3m	-1.029	0.277	0.000	111.639	6	4.762

自動化下載

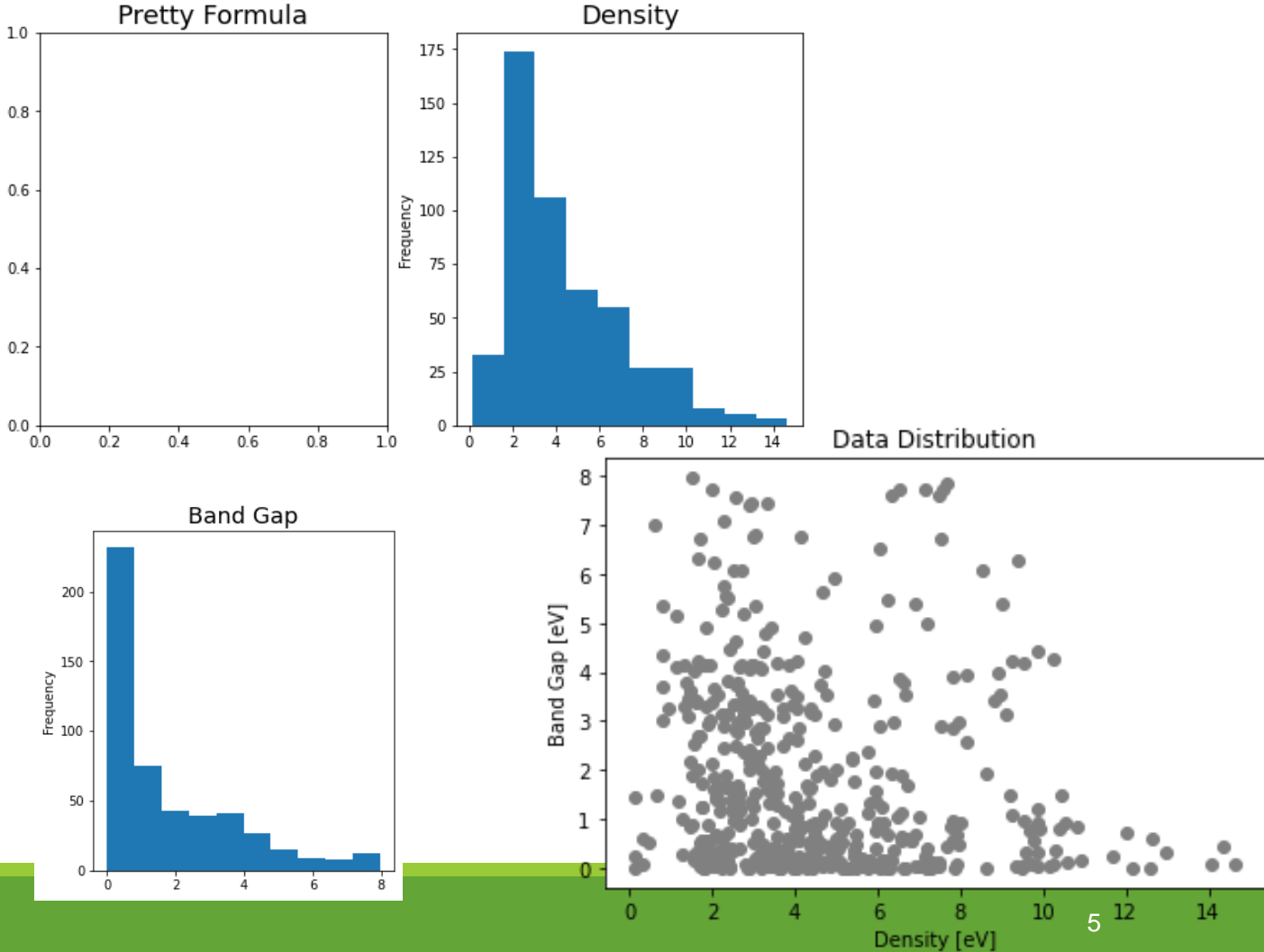
open Materials Application Programming Interface (API)



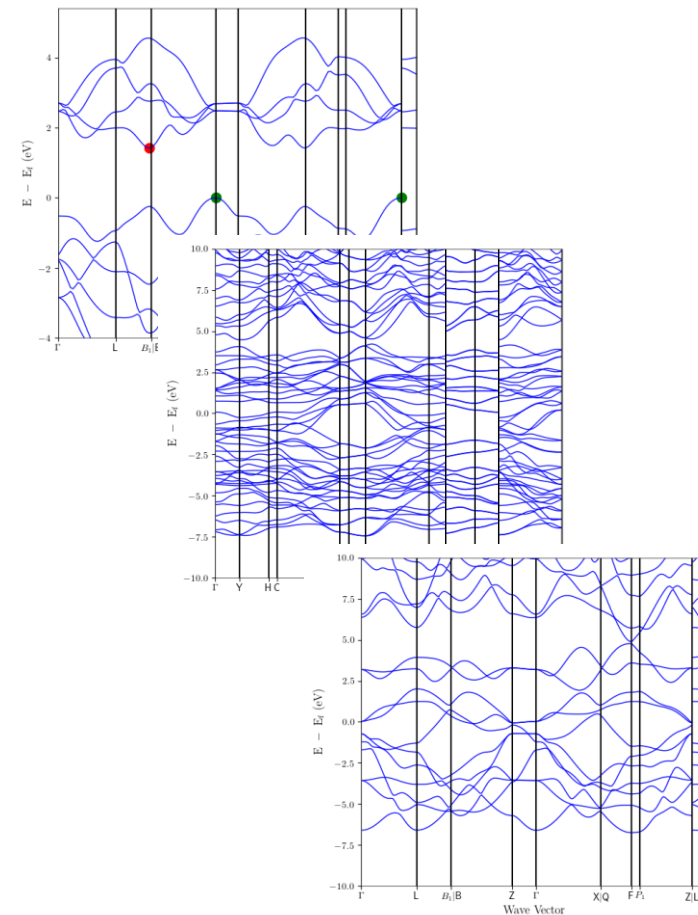
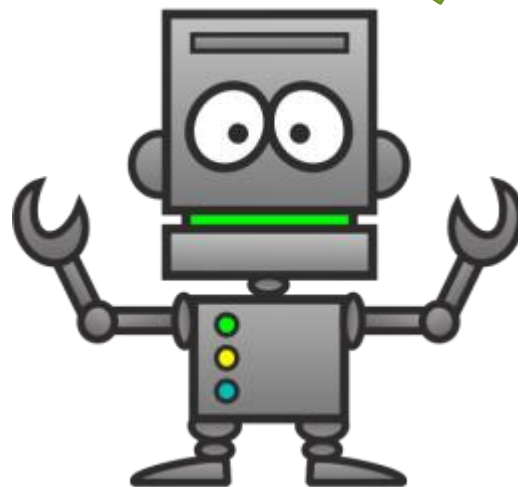
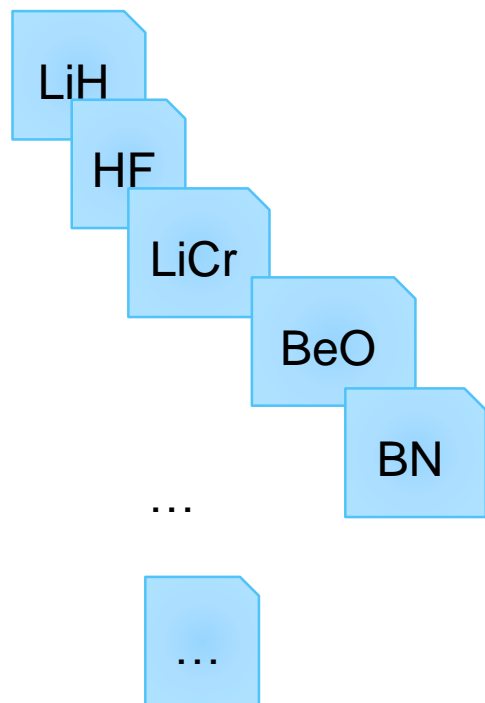
```
1 import csv, itertools
2 from pymatgen import MP
3
4 API_KEY = 'MTF9psie'
5
6 all_symbols = [
7     "F", "Ne",
8     "Cl", "Ar",
9     "Mn", "Fe",
10    "As", "S",
11    "Mo", "Ag", "Cd", "In", "Sn",
12    "Sb", "T",
13    "Nd", "Tb", "Dy", "Ho", "Er",
14    "Tm", "Y",
15    "Au", "Po", "At", "Rn", "Fr",
16    "Ac", "Th", "Pa", "U", "Np", "Pu", "Am", "Cm", "Bk",
17    "Fm", "Md"
18
19 allBinaries = [
20     with MPRester(A
21         with open
22         fieldn
23         writer =
24         writer.writ
25         count = 0
26
27     # writing bandgap and density to a csv
28     for system in allBinaries:
29         if count > 500:
30             break
```

Clean, Prepare, Manipulate Data

Pretty Formula	Density	Band Gap
LiH	0.814395727	3.018
BeH2	0.807534506	5.3418
B3H5	0.811598971	3.6983
HC	1.409269759	3.091
HN	1.336811255	4.1598
H2O2	1.834767366	4.1525
HF	1.721540912	6.7187
NaH	1.394380186	3.7974
MgH2	1.450102639	3.6284
AlH3	1.459511166	2.1855
...



Train Model



這是你寫的門得烈夫機器人

Model

Finding a function from data

$f_1(\text{LiH}) = \text{aaaaaa}$ $f_2(\text{LiH}) = \text{eeeeee}$
 $f_1(\text{HF}) = \text{bbbbbb}$ $f_2(\text{HF}) = \text{ffffff}$
 $f_1(\text{LiCr}) = \text{cccccc}$ $f_2(\text{LiCr}) = \text{gggggg}$
 $f_1(\text{BeO}) = \text{dddddd}$ $f_2(\text{BeO}) = \text{hhhhh}$

訓練的過程說穿了就是找出一個合適
的function來描述輸入和輸出的關係

Scikit-Learn Regression algorithm

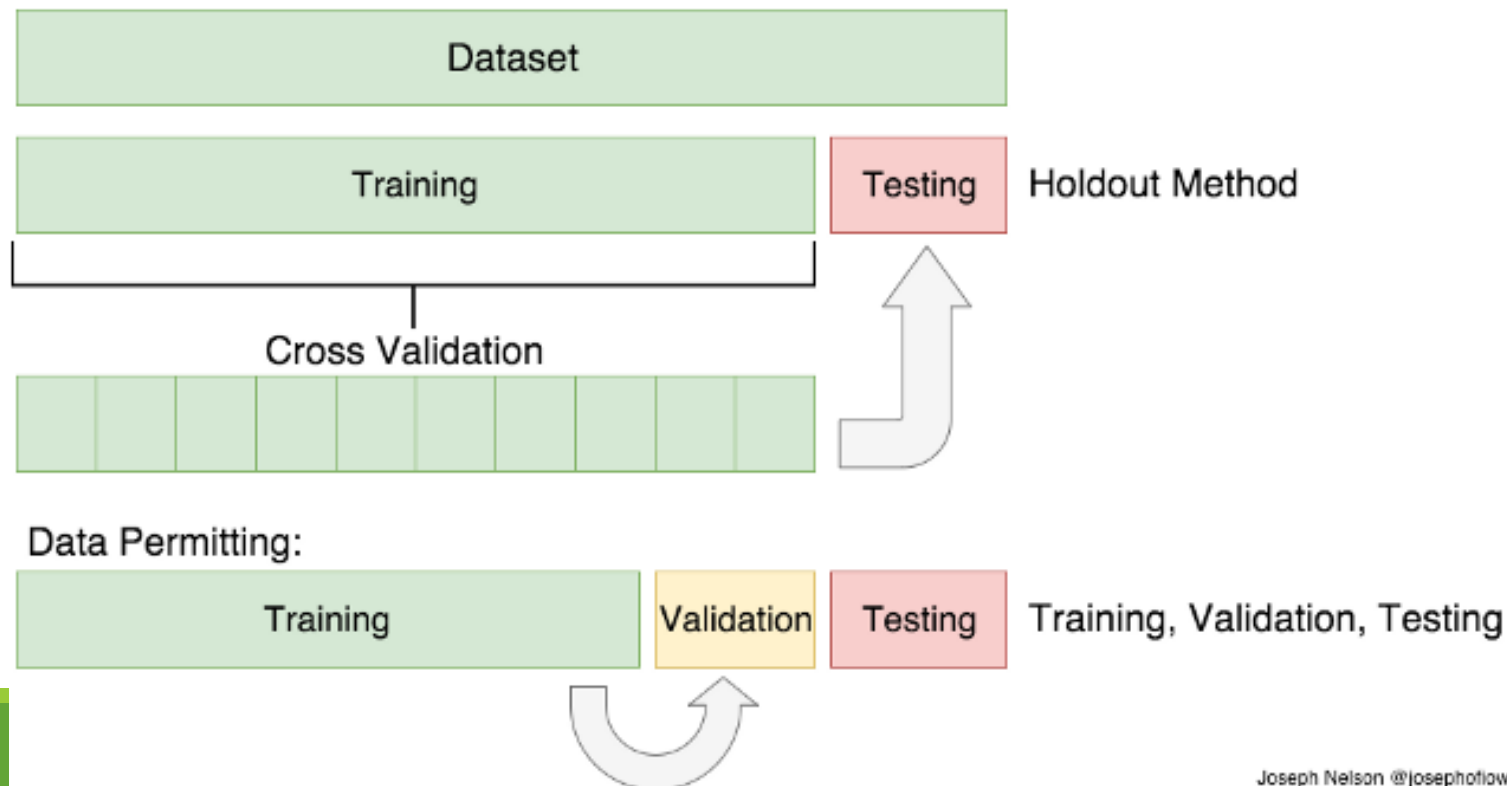


- Linear Models
- Kernel ridge regression
- Support Vector Machines
- Gaussian Processes
- Decision Trees
- Ensemble methods
- ...

https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

Test Data (Test Model)

- 不要把所有所有的資料都餵進去給model，只要把一部分的資料餵進去(Training Dataset)訓練模型，需要保留一些資料拿來檢驗模型(Testing Dataset)。
- Cross Validation(交叉驗證)的部份之後會再講。



Train/Test Split(Manually)

□ 使用python list 物件的slice功能

my_list[start(開始的index):end(結束的index):sep(間隔)]

取出來的值只會包含開頭的元素，不包含結束的元素。間隔如果沒有特別輸入的話，預設值為1。

```
x = range(10) # [0, 1, 2, ..., 9]
```

```
x[1:5] ---> [1, 2, 3, 4]
```

```
x[:3] ---> [0, 1, 2] #省略開始的元素，表示從第一個開始取。
```

```
x[3:] ---> [3, 4, 5,..., 9] #省略結束的元素，表示取到最後一個。
```

```
x[0:-1] ---> [1, 2, ..., 8] #-1表示取到倒數第一個元素。
```

Train/Test Split

□ sklearn.model_selection.train_test_split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.2, random_state = 0)
```

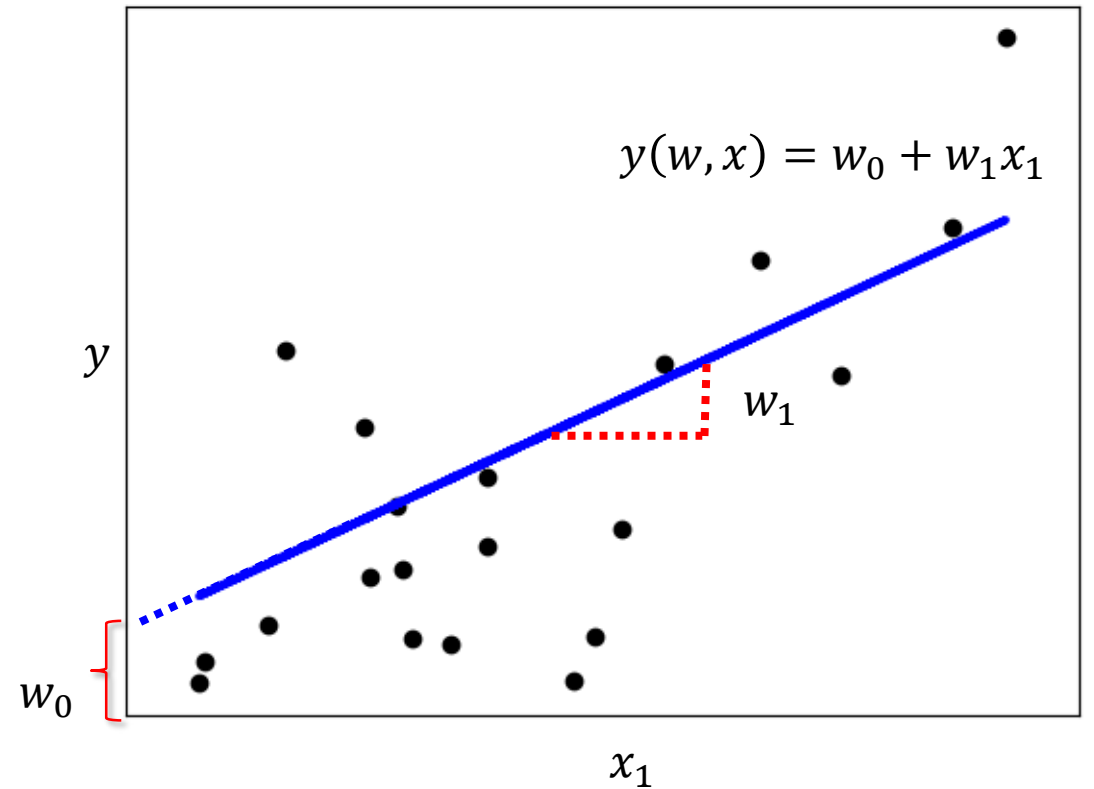
https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

Regression algorithm(Manually)

- Regression的過程是找出輸入(independent variable, feature)和輸出(dependent variable, target)之間的關係。
- 使用線性關係(模型)描述feature與target就稱為 Linear regression。
- Simple linear regression
$$y(w, x) = w_0 + w_1 x_1$$
- multiple regression
$$y(w, x) = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p$$

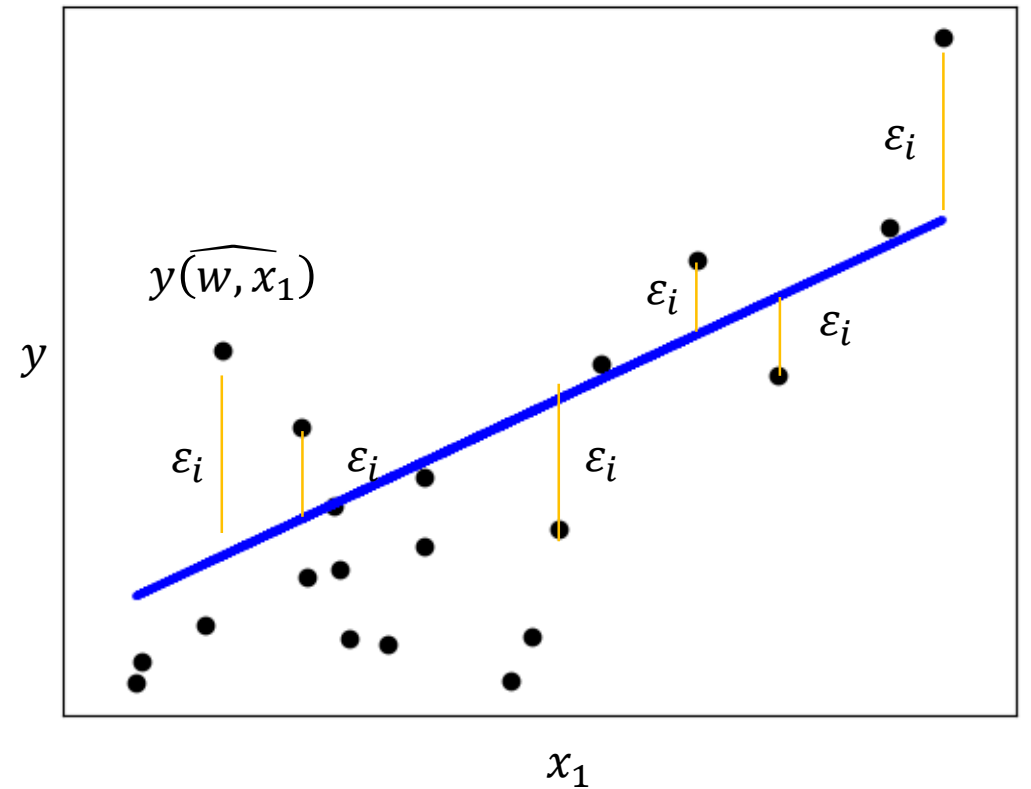
w_0 是截距(Intercept)

$w = (w_1, w_2, \dots, w_p)$ 是斜率(Slope)



Regression algorithm(Manually)

- Regression的過程是找 w_0 和 $w = (w_1, w_2, \dots, w_p)$ 。
- 收集一組資料 $(x_i, y_i), i = 1, 2, \dots, n$ ，將每個點都帶到模型內可以得到模型的預估值
$$\widehat{y(w, x_i)} = w_0 + w_1 x_i, i = 1, 2, \dots, n$$
- 預估值和實際值的差異稱為誤差(error)或稱為殘差(Residual)
$$\varepsilon_i = y(w, x_i) - \widehat{y(w, x_i)}$$
- Regression的目標是希望找到一組參數 $(\widehat{w_0}, \widehat{w_1})$ 使得模型的殘差越小越好，數值上有許多種方法可以找出這組參數，最小平方法是一種常用的方法。



Regression algorithm(Manually)

□ 因為誤差值有正有負，取平方後皆為正值，所以我們會很希望所有訓練樣本的誤差平方和(Sum Square error, SSE)接近0。

□ $\text{Loss}(\widehat{w}_0, \widehat{w}_1) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n (y_i - (\widehat{w}_0 + \widehat{w}_1 x_i))^2$

□ 極值會出現在微分為0的地方。對殘差分別做 $\widehat{w}_0, \widehat{w}_1$ 的偏微分。

□
$$\frac{\partial \text{Loss}(\widehat{w}_0, \widehat{w}_1)}{\partial \widehat{w}_0} = \frac{\partial \sum_{i=1}^n (y_i - (\widehat{w}_0 + \widehat{w}_1 x_i))^2}{\partial \widehat{w}_0} = 0$$
$$\rightarrow -2 \sum_{i=1}^n (y_i - \widehat{w}_0 - \widehat{w}_1 x_i) = 0 \rightarrow \widehat{w}_0 = \bar{y} - \widehat{w}_1 \bar{x}$$

□
$$\frac{\partial \text{Loss}(\widehat{w}_0, \widehat{w}_1)}{\partial \widehat{w}_1} = \frac{\partial \sum_{i=1}^n (y_i - (\widehat{w}_0 + \widehat{w}_1 x_i))^2}{\partial \widehat{w}_1} = 0$$
$$\rightarrow -2 \sum_{i=1}^n (y_i - \widehat{w}_0 - \widehat{w}_1 x_i) x_i = 0 \rightarrow \widehat{w}_1 = \frac{\sum_{i=1}^n (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

Regression algorithm

□ sklearn.linear_model.LinearRegression

```
from sklearn import linear_model
lr = linear_model.LinearRegression()

lr.fit(X_train.reshape(-1, 1), y_train)
y_pred = lr.predict(test)
```

Regression algorithm

□ sklearn.neural_network.MLPRegressor

```
from sklearn.neural_network import MLPRegressor
mlpr = MLPRegressor(hidden_layer_sizes=(5, 3),
activation='relu', solver='adam', alpha=0.0001,
batch_size='auto', learning_rate_init=0.001, max_iter=10000,
random_state=497)
mlpr.fit(X_train.reshape(-1, 1), y_train)
y_pred = mlpr.predict(test)
```