

Python 與迴歸分析

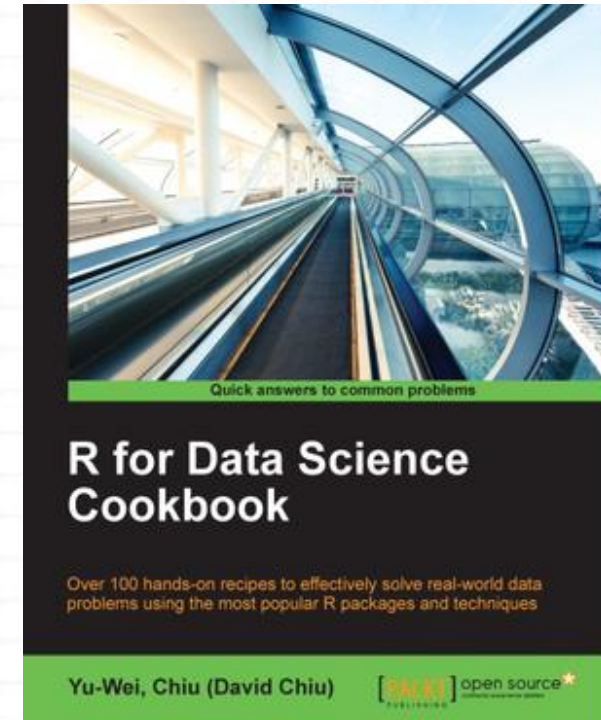
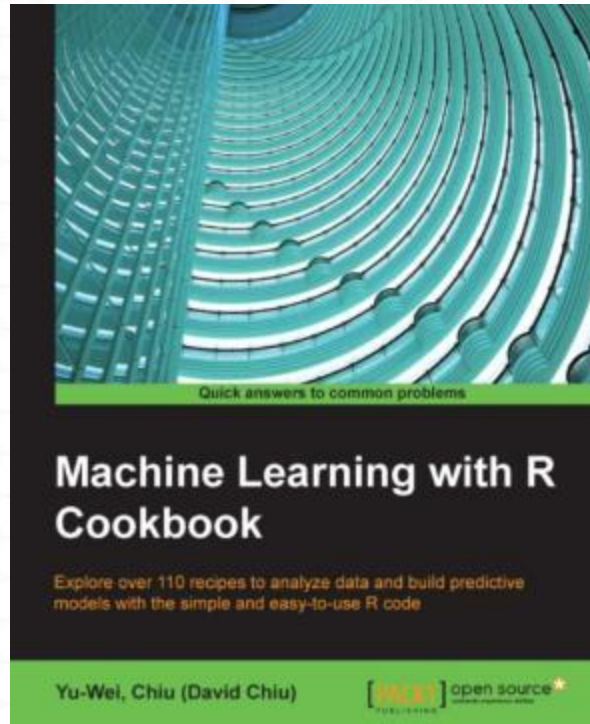
David Chiu

關於我



- 大數軟體有限公司創辦人
- 前趨勢科技工程師
- ywchiu.com
- 大數學堂 <http://www.largitdata.com/>
- 粉絲頁 <https://www.facebook.com/largitdata>
- R for Data Science Cookbook <https://www.packtpub.com/big-data-and-business-intelligence/r-data-science-cookbook>
- Machine Learning With R Cookbook <https://www.packtpub.com/big-data-and-business-intelligence/machine-learning-r-cookbook>

Machine Learning With R Cookbook (机器学习与R语言实战) & R for Data Science Cookbook (数据科学：R语言实现)



Author: David (YU-WEI CHIU) Chiu

課程資料

- 所有課程補充資料、投影片皆位於
 - <https://github.com/ywchiu/holintech>

Python 開發工具簡介

安裝Anaconda

選擇Python 3.6 版

Python 3.6 version *

↓ Download

[64-Bit Graphical Installer \(537 MB\)](#) ?

[32-Bit Graphical Installer \(436 MB\)](#)

Python 2.7 version *

↓ Download

[64-Bit Graphical Installer \(523 MB\)](#) ?

[32-Bit Graphical Installer \(420 MB\)](#)

<https://www.anaconda.com/download/>

Python 2.x v.s. Python 3.x

- *Python 2.x is legacy, Python 3.x is the present and future of the language*
- *Python 3.x*
 - 更簡潔的語法，Unicode 支援與強大內建函式庫
 - 尚在持續更新與開發中
- *Python2.x*
 - 支援主要作業環境
 - 協力廠商函式庫主要支援版本

從命令列執行Python

- 互動式Shell – 即時看到結果，難以編修程式

C:\Users\david>python

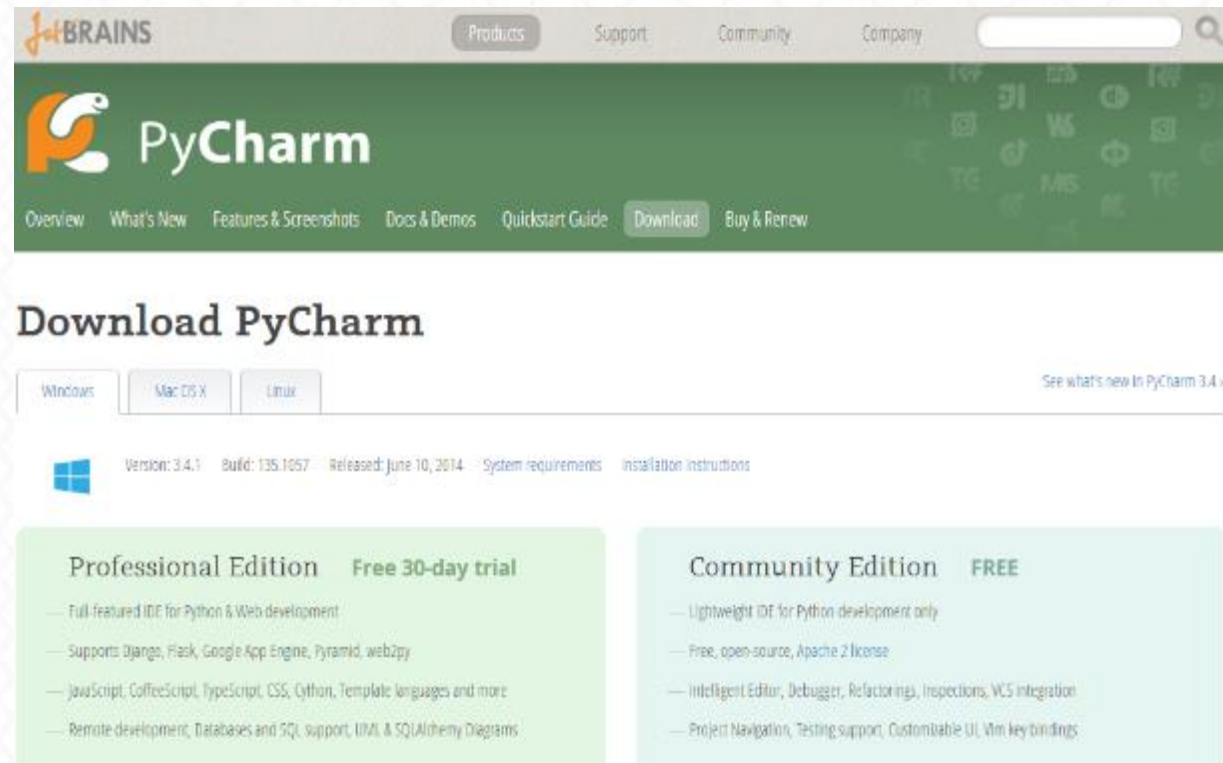
- 從檔案執行Python – 無法即時看到結果，容易編修程式

C:\Users\david>python test.py

Python IDE - PyCharm

■ 建議使用該工具做專案開發

□ <https://www.jetbrains.com/pycharm/download/>



The screenshot shows the JetBrains website's PyCharm download page. The header includes the JetBrains logo and navigation links for Products, Support, Community, and Company. The main banner features the PyCharm logo and a list of links: Overview, What's New, Features & Screenshots, Docs & Demos, Quickstart Guide, Download, and Buy & Renew. Below the banner, the section is titled "Download PyCharm" and includes tabs for Windows, Mac OS X, and Linux. A link to "See what's new in PyCharm 3.4" is also present. The page lists the version (3.4.1), build (135.1057), and release date (June 10, 2014), along with links for system requirements and installation instructions. Two main options are presented: the Professional Edition, which is a free 30-day trial, and the Community Edition, which is free. Both editions list their respective features.

Professional Edition **Free 30-day trial**

- Full-featured IDE for Python & Web development
- Supports Django, Flask, Google App Engine, Pyramid, web2py
- JavaScript, CoffeeScript, TypeScript, CSS, Cython, Template languages and more
- Remote development, Databases and SQL support, UML & SQLAlchemy Diagrams

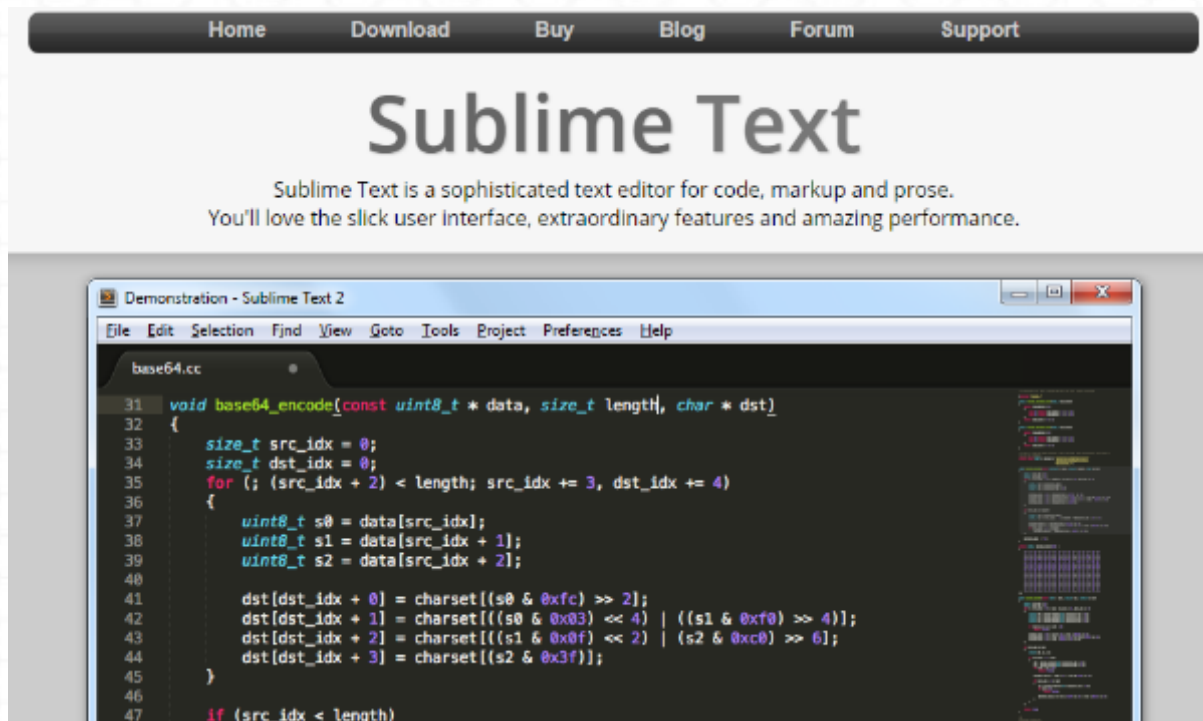
Community Edition **FREE**

- Lightweight IDE for Python development only
- Free, open-source, Apache 2 license
- Intelligent Editor, Debugger, Refactorings, Inspections, WCS integration
- Project Navigation, Testing support, Customizable UI, Win key bindings

Sublime Text

■ MAC 使用者建議使用該工具做專案開發

□ <https://www.sublimetext.com/>



Jupyter Notebook

■ 使用於教學與資料探索，不適合開發大型專案

□ `pip install jupyter`

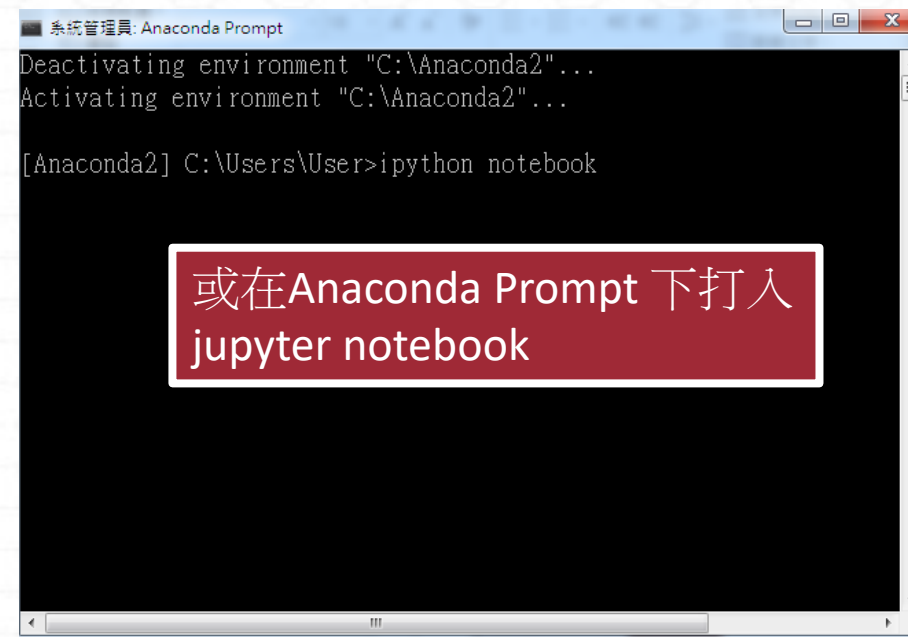
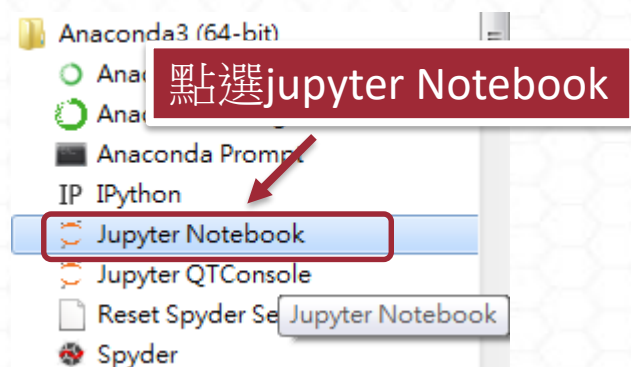


[INSTALL](#) [PROJECT](#) [COMMUNITY](#) [DOCUMENTATION](#) [NBVIEWER](#) [BLOG](#) [DONATE](#)



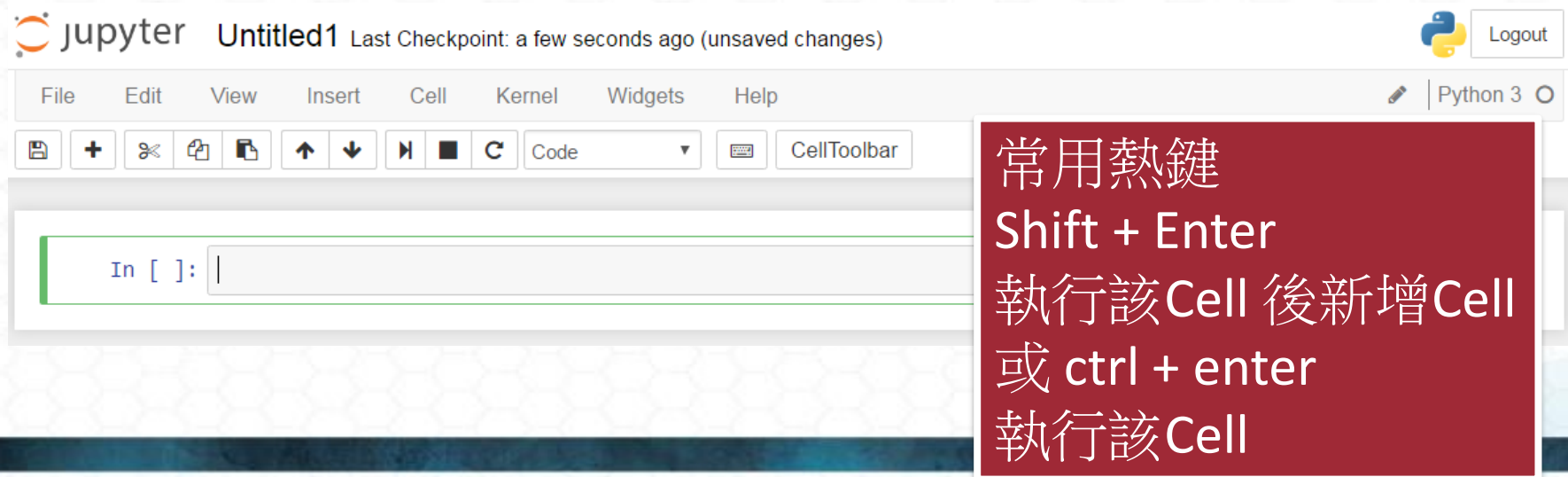
Open source, interactive data science and scientific computing across over 40

使用 Jupyter Notebook



啟用 Jupyter (jupyter Notebook)

- 在命令列下打:
 - jupyter notebook
 - 自動開啟瀏覽器後便可瀏覽 (預設為localhost:8888)
- 可匯出.ipynb, .py 各種不同格式檔案
- 瀏覽快捷鍵 Help -> Keyboard Shortcuts



機器學習與Python語言

機器學習

- 機器學習的目的是：歸納 (Induction)

從詳細事實到一般通論

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 (1998)

- 找出有效的預測模型

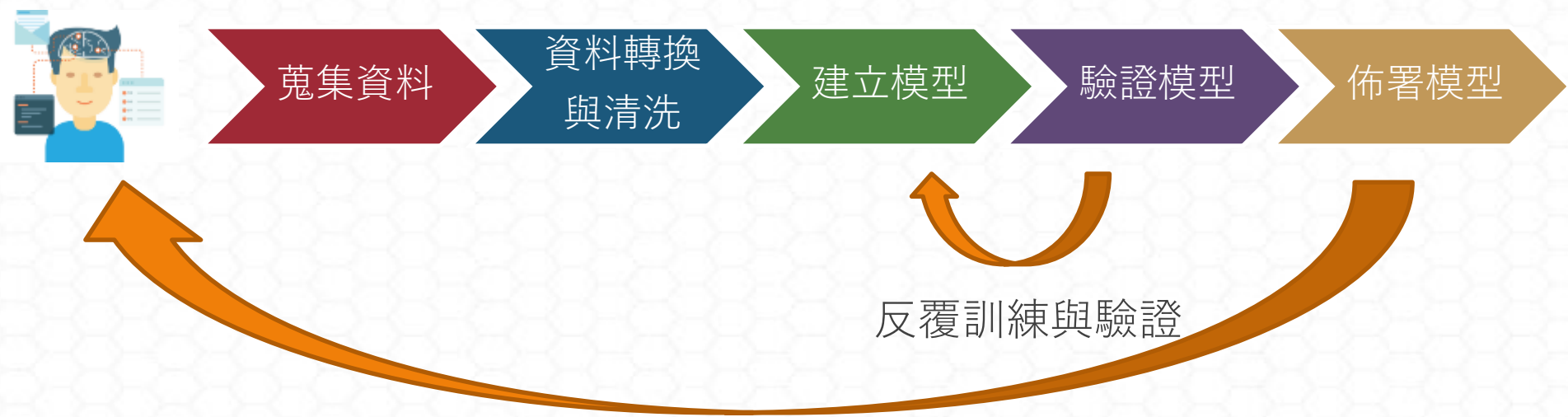
一開始都從一個簡單的模型開始

藉由不斷餵入訓練資料，修改模型

不斷提升預測績效

機器學習步驟

使用者行為



機器學習問題分類

- 監督式學習 (Supervised Learning)
 - 回歸分析 (Regression)
 - 分類問題 (Classification)
- 非監督式學習 (Unsupervised Learning)
 - 降低維度 (Dimension Reduction)
 - 分群問題 (Clustering)

使用監督式學習進行預測

■ 分類問題

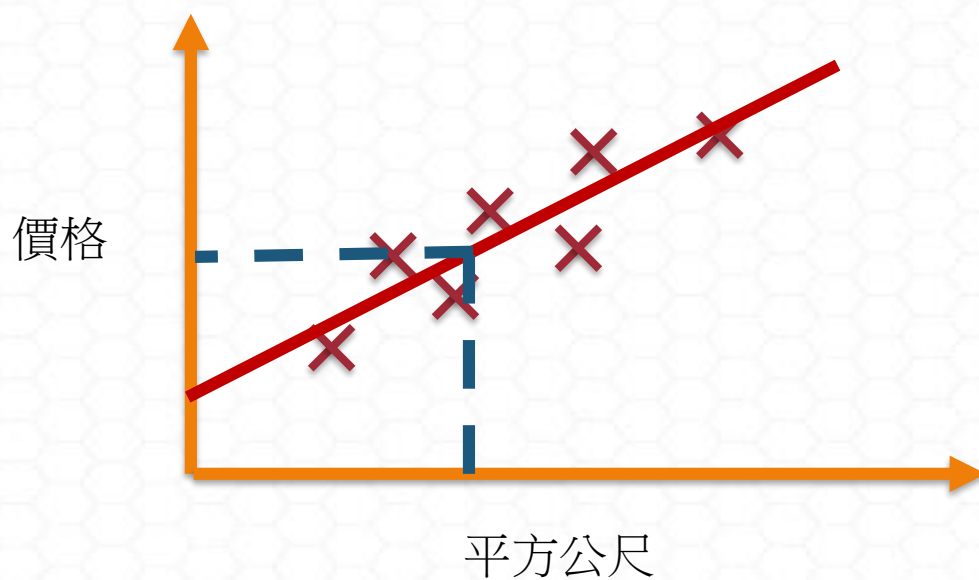
- 根據已知標籤的訓練資料集(Training Set)，產生一個新模型，用以預測測試資料集(Testing Set)的標籤。
- e.g. 客戶流失分析

■ 回歸分析

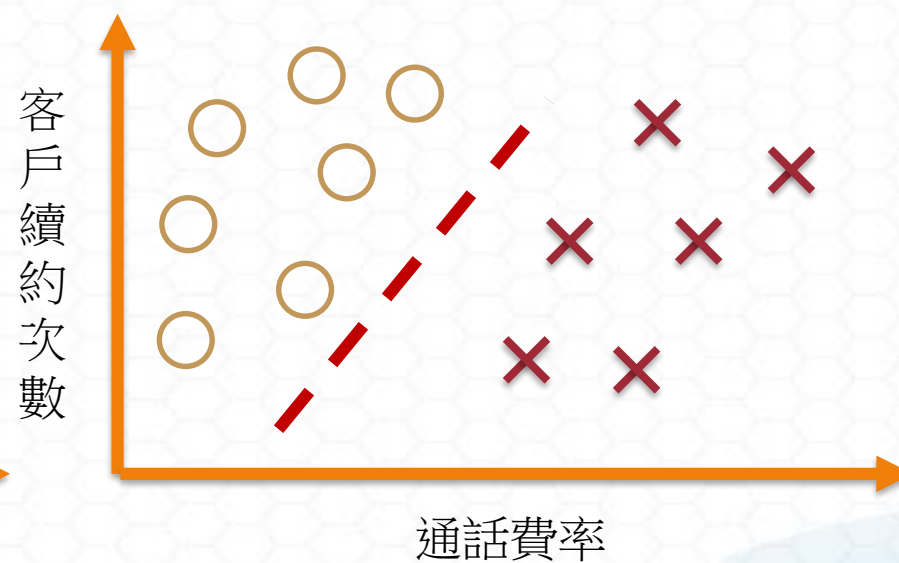
- 使用一組已知對應值的資料產生的模型，預測新資料的對應值
- e.g. 股價預測

使用監督式學習進行預測

回歸分析



分類問題



使用非監督式學習找出隱藏的架構

■ 降低維度

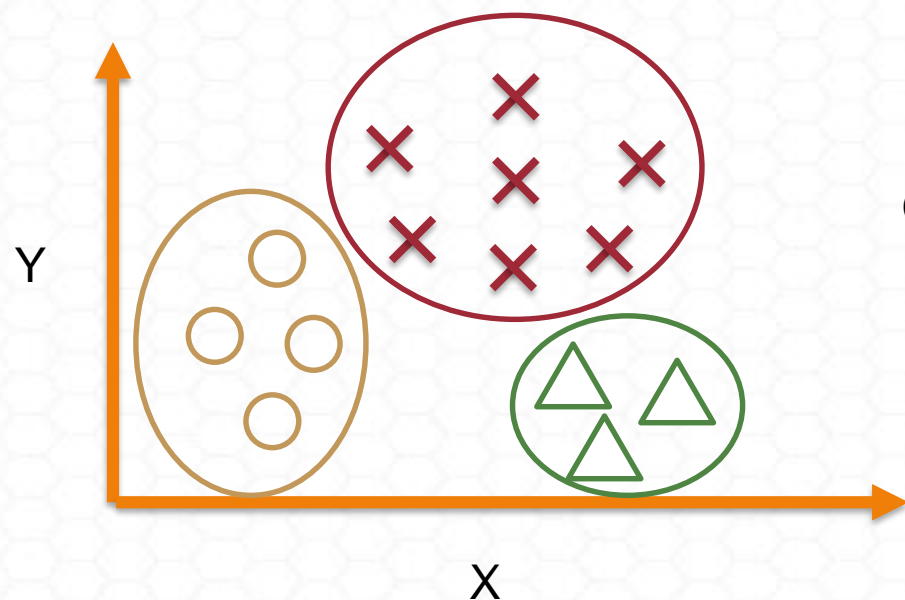
- 產生一有最大變異數的欄位線性組合，可用來降低原本問題的維度與複雜度
- e.g. 濃縮用到的特徵，編纂成一個新指標

■ 分群問題

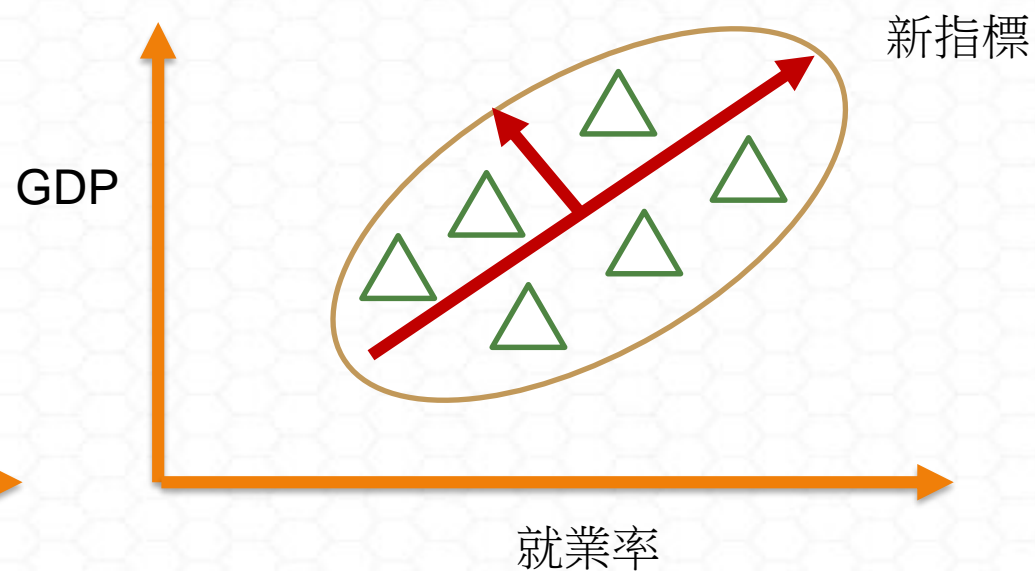
- 物以類聚 (近朱者赤、近墨者黑)
- e.g. 將客戶分層

使用非監督式學習找出隱藏的架構

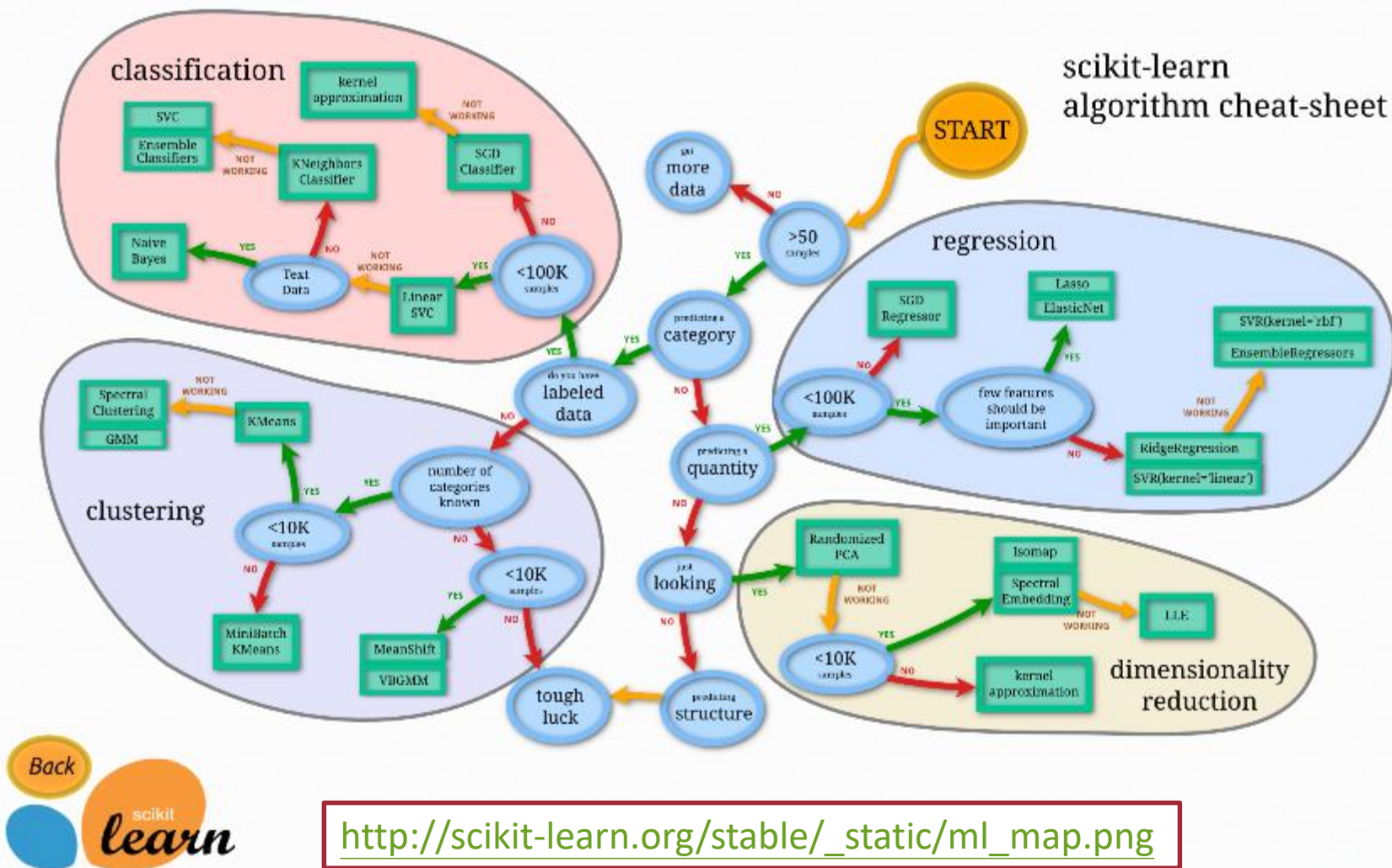
分群問題



降低維度



機器學習地圖



線性迴歸

回歸分析

- 線性回歸是研究單一依變項(dependent variable)與一個或以上自變項(independent variable)之間的關係
- 線性回歸有兩個主要用處：
 - 預測指的是用已觀察的變數來預測依變項
 - 因果分析則是將自變項當作是依變項發生的原因
- **Francis Galton 在1886 年發表論文Regression Towards Mediocrity in Hereditary Stature，認為孩子身高跟父親的身高成正相關，而身高的變異數不會隨時間而增加。**

簡單線性回歸

■ 數學模型

□ $y = \beta_1 x + \beta_0$

□ y 是依變數

□ x 是自變數

□ β_1 是回歸係數

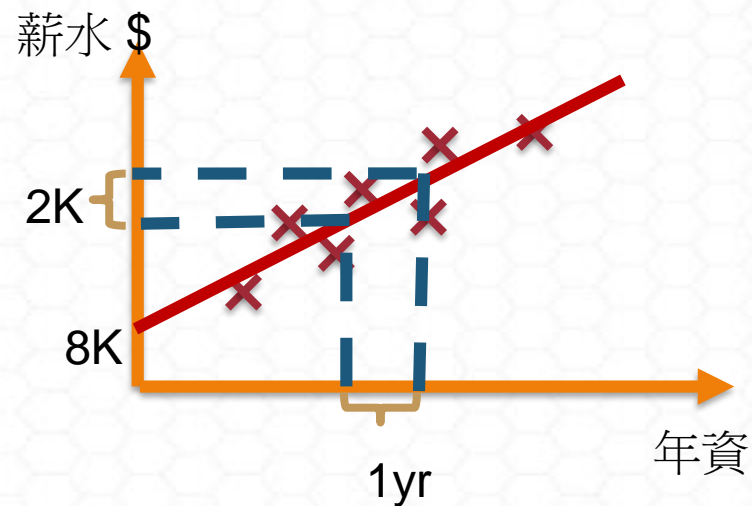
□ β_0 是截距



$$\text{薪資} = \beta_1 \text{年資} + \beta_0$$

$$\beta_0 = 8k$$

$$\beta_1 = 2k/\text{年}$$

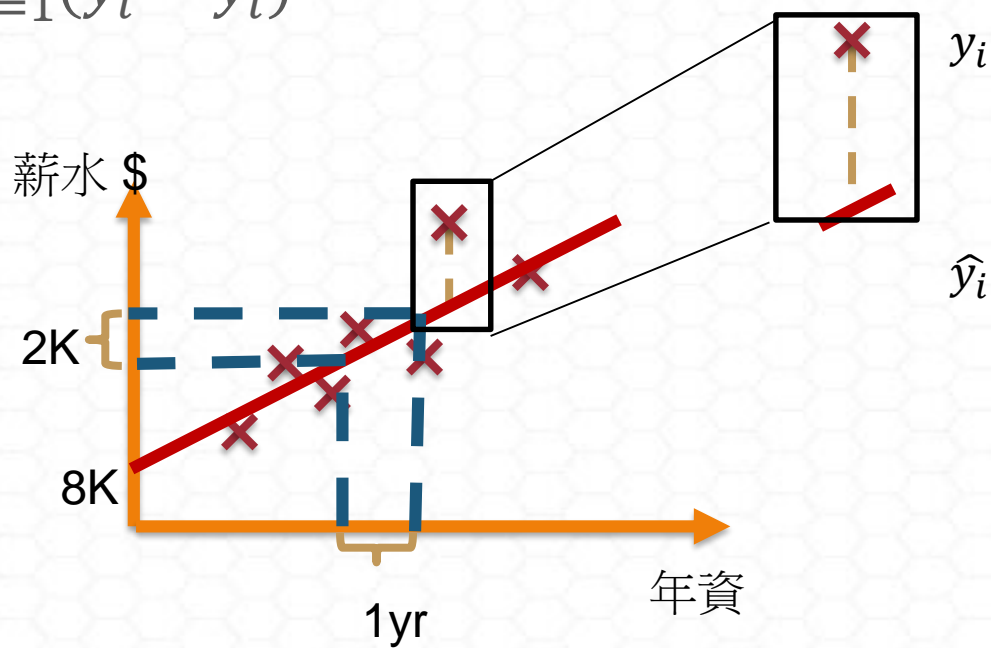


最小平方估計法 - OLS

■ 找出殘差平方和最小的一條線

▣ 殘差 $e_i = (y_i - \hat{y}_i)$

▣ 殘差平方和 $SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$



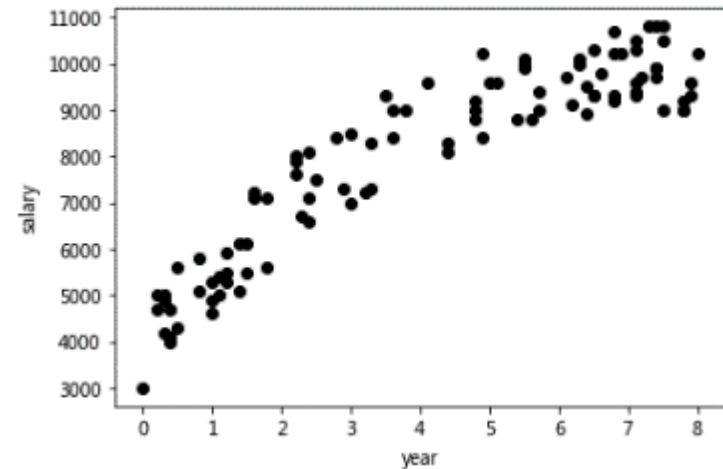
讀取薪水數據

```
import pandas  
df = pandas.read_csv('data/salary.csv', index_col=0)  
df.head()
```

	year	salary
1	2.4	6600
2	5.5	10100
3	3.3	7300
4	0.2	5000
5	1.5	6100

繪製資料

```
from matplotlib import pyplot as plt  
X = df[['year']]  
Y = df['salary'].values  
plt.scatter(X, Y, color='black')  
plt.xlabel('year')  
plt.ylabel('salary')  
plt.show()
```



使用scikit-learn 進行預測

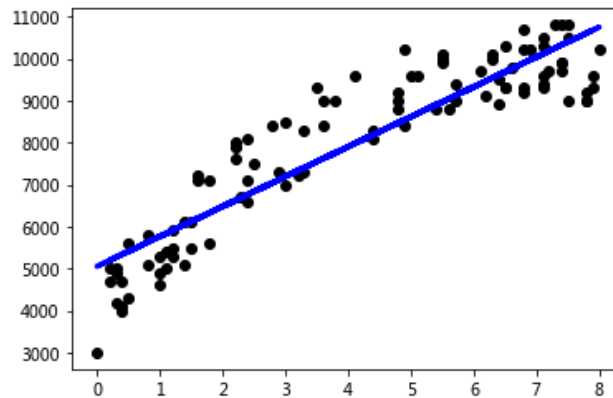
```
from sklearn.linear_model import LinearRegression
```

```
regr = LinearRegression()
```

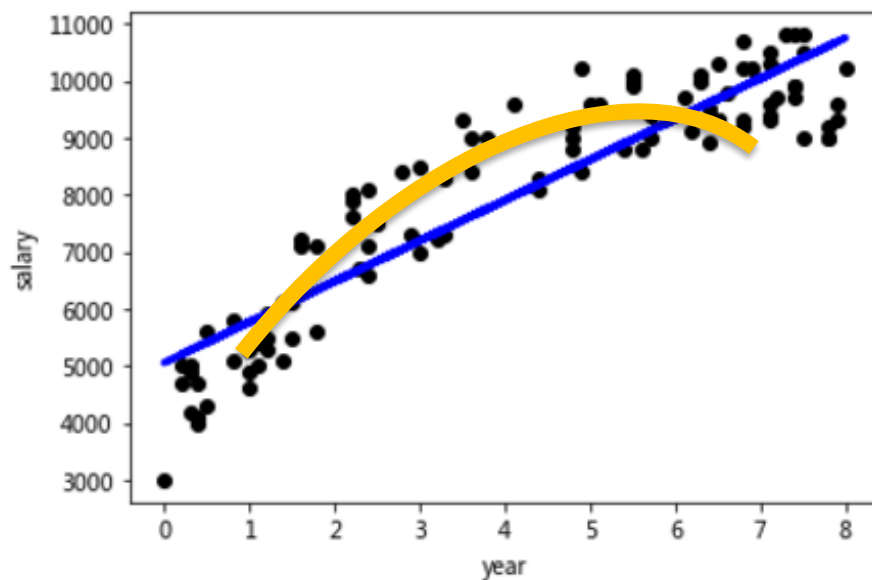
```
regr.fit(X,Y)
```

將回歸線繪製在圖上

```
print('Coefficients:', regr.coef_)  
print('Intercept:', regr.intercept_)  
plt.scatter(X, Y, color='black')  
plt.plot(X, regr.predict(X), color='blue', linewidth=3)  
plt.show()
```



薪資會無限成長嗎？



$$y = \beta_2 x^2 + \beta_1 x + \beta_0$$

多項式線性回歸

建立多項式線性回歸

```
from sklearn.preprocessing import  
PolynomialFeatures  
from sklearn.linear_model import  
LinearRegression
```

```
poly_reg = PolynomialFeatures(degree=2)  
X_ = poly_reg.fit_transform(X)
```

```
regr = LinearRegression()  
regr.fit(X_, Y)
```


增添多項式回歸線

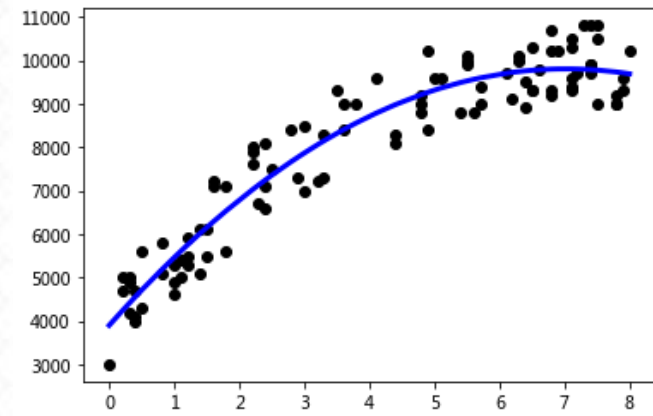
```
X2 = X.sort_values(['year'])
```

```
X2_ = poly.fit_transform(X2)
```

```
plt.scatter(X, Y, color='black')
```

```
plt.plot(X2, regr.predict(X2_), linewidth = 3, color="blue")
```

```
plt.show()
```



多元線性回歸模型

簡單線性回歸

$$y = \beta_1 x + \beta_0$$

y 是依變數

x 是自變數

β_1 是回歸係數

β_0 是截距

多元線性回歸

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots \beta_k x_{ki} + \beta_0$$

$x_1 \cdots x_k$ 為引數

y 是依變數

β_1 是回歸係數

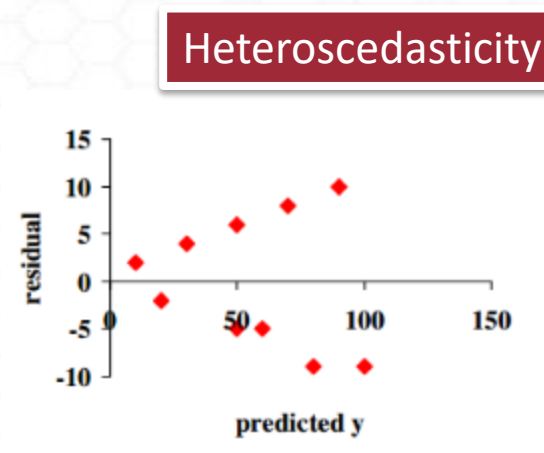
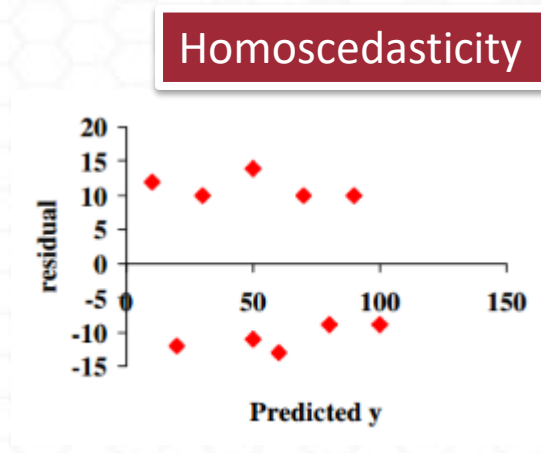
β_0 是截距

線性模型的假設

- 最小平方估計法 OLS 適用於
 - 誤差變數是常態分佈
 - 可以使用Shapiro 檢定,或用長條圖驗證
 - 錯誤變數的期望值是0
 - 使用最小估計法求是否誤差的期望值是0
 - 誤差的變異數是常數
 - 驗證變異數齊一性 (Homoscedasticity)
 - 任兩個依變數所關聯的錯誤互為獨立
 - 驗證 $Cov(\varepsilon_1, \varepsilon_2) = 0$
 - 缺乏共線性 (Multicollinearity)

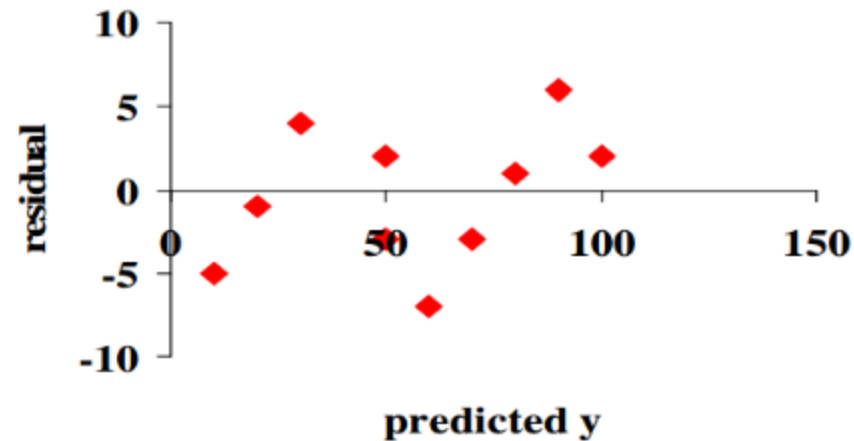
驗證變異數齊一性(Homoscedasticity)

- 齊一性(Homoscedasticity): 增加X 時, Y 會穩定增長
- 相異性(Heteroscedasticity): 增加X 時, Y 增長幅度較大(小)



驗證錯誤是否獨立

- 如果 $Cov(\varepsilon_1, \varepsilon_2) \neq 0$ 代表所預測出來的Y 值會受X值產生的順序影響，稱為自我相關性(autocorrelation)。通常發生在時間序列上



讀取房屋數據

```
import pandas  
df = pandas.read_csv('data/house-prices.csv')  
df.head()
```

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood
0	1	114300	1790	2	2	2	No	East
1	2	114200	2030	4	2	3	No	East
2	3	114800	1740	3	2	1	No	East
3	4	94700	1980	3	2	3	No	East
4	5	119800	2130	3	3	3	No	East

如何建立多元回歸模型

建立 Dummy Variable

$$\text{Price} = \beta_1 \text{Sqft} + \beta_2 \text{Bedrooms} + \dots ?$$

Dummy Variable

	Home	Price	SqFt	Bedrooms	Bathrooms	Offers	Brick	Neighborhood		No	Yes
0	1	114300	1790	2	2	2	No	East			0
1	2	114200	2030	4	2	3	No	East			0
2	3	114800	1740	3	2	1	No	East			0
3	4	94700	1980	3	2	3	No	East			0
4	5	119800	2130	3	3	3	No	East			0

NO = 1- YES

會產生Multicollinearity 問題

永遠都只留一個Dummy Variable 即可

轉換類別資料

```
house = pandas.concat([df,pandas.get_dummies(df['Brick']),  
pandas.get_dummies(df['Neighborhood'])],axis=1)  
del house['No']  
del house['West']  
del house['Brick']  
del house['Neighborhood']  
del house['Home']  
house.head()
```

	Price	SqFt	Bedrooms	Bathrooms	Offers	Yes	East	North
0	114300	1790	2	2	2	0	1	0
1	114200	2030	4	2	3	0	1	0
2	114800	1740	3	2	1	0	1	0
3	94700	1980	3	2	3	0	1	0
4	119800	2130	3	3	3	0	1	0

建立預測模型

```
from sklearn.linear_model import LinearRegression
```

```
regr = LinearRegression()
```

```
X = house[['SqFt', 'Bedrooms', 'Bathrooms', 'Offers', 'Yes', 'East', 'North']]
```

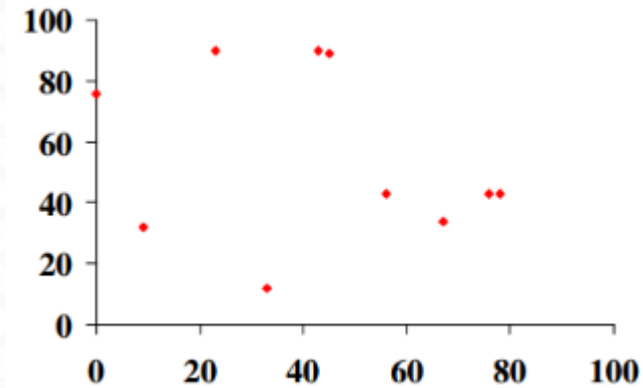
```
Y = house['Price'].values
```

```
regr.fit(X, Y)
```

```
regr.predict(X)
```

線性關係顯著性檢定

■ $y = \beta_1 x + \beta_0 + \epsilon$



如數據之間沒有相關性， β_1 應該接近於0，但事實上 β_1 不太可能等於零，因此要如何驗證 β_1 近似於0？

驗證相關性

- 假設如下:

- $H_0: \beta_1 = 0$

- $H_a: \beta_1 \neq 0$

- test : $t = \frac{\hat{\beta}_1 - \beta_1}{s_{\hat{\beta}_1}}$

- 假設虛無假設正確

$$t = \frac{\hat{\beta}_1}{s_{\hat{\beta}_1}}$$

假設檢定

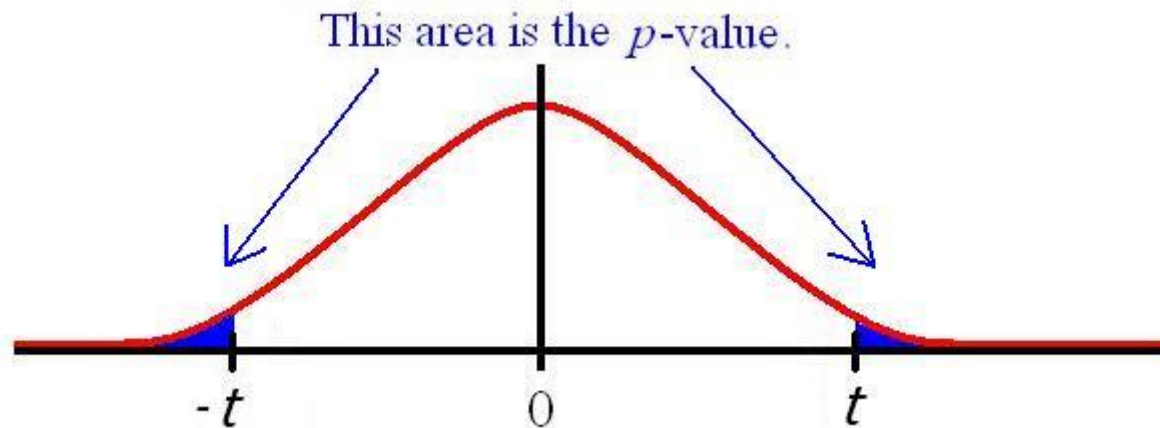
- 兩種假設：虛無假設(null hypothesis)與對立假設(alternative hypothesis)
- 檢定步驟
 - 寫出所有假設
 - H_0 原始假設: 觀測值是隨機結果
 - H_1 對立假設: 存在一些因素影響結果
 - 檢查統計量
 - 檢驗數量與分佈
 - 檢驗P值
 - 觀察到極端機率有多少，P值越小越不利於原始假設
 - 比較P值與顯著性水準
 - e.g. <0.05 代表顯著

驗證假設檢定

- 進行假設檢定是為了驗證實驗結果為顯著(Significant)。但為了驗證對立假設正確，即實驗變數之間是有相關的，則必須推翻虛無假設(即實驗變數之間毫無關聯)
- 非A即B 的驗證：一個為真(對立假設)，另一個即非真(虛無假設)

Student's t test

- 用於樣本含量較小（例如 $n < 30$ ），總體標準差 σ 未知的常態分佈資料。它是用T分佈理論來推斷差異發生的概率，從而判定兩個平均數的差異是否顯著
- T檢驗是William Sealy Gosset了觀測釀酒質量而發明的，Gosset在Guinness 釀酒廠擔任統計學家。戈斯特於1908年在Biometrika上公佈T檢驗，但因其老闆認為其為商業機密而被迫使用筆名（Student）



回歸模型評估

```
import statsmodels.api as sm
```

```
X2 = sm.add_constant(X)
```

```
est = sm.OLS(Y, X2)
```

```
est2 = est.fit()
```

```
print(est2.summary())
```

OLS Regression Results						
=====						
Dep. Variable:	y	R-squared:	0.869			
Model:	OLS	Adj. R-squared:	0.861			
Method:	Least Squares	F-statistic:	113.3			
Date:	Sat, 06 May 2017	Prob (F-statistic):	8.25e-50			
Time:	23:56:36	Log-Likelihood:	-1356.7			
No. Observations:	128	AIC:	2729.			
Df Residuals:	120	BIC:	2752.			
Df Model:	7					
Covariance Type:	nonrobust					
=====						
	coef	std err	t	P> t	[95.0% Conf. Int.]	

const	2.284e+04	1.02e+04	2.231	0.028	2573.371	4.31e+04
SqFt	52.9937	5.734	9.242	0.000	41.640	64.347
Bedrooms	4246.7939	1597.911	2.658	0.009	1083.042	7410.546
Bathrooms	7883.2785	2117.035	3.724	0.000	3691.696	1.21e+04
Offers	-8267.4883	1084.777	-7.621	0.000	-1.04e+04	-6119.706
Yes	1.73e+04	1981.616	8.729	0.000	1.34e+04	2.12e+04
East	-2.224e+04	2531.758	-8.785	0.000	-2.73e+04	-1.72e+04
North	-2.068e+04	3148.954	-6.568	0.000	-2.69e+04	-1.44e+04

參數估計

	coef	std err	t	P> t	[95.0% Conf. Int.]	
const	2.284e+04	1.02e+04	2.231	0.028	2573.371	4.31e+04
SqFt	52.9937	5.734	9.242	0.000	41.640	64.347
Bedrooms	4246.7939	1597.911	2.658	0.009	1083.042	7410.546
Bathrooms	7883.2785	2117.035	3.724	0.000	3691.696	1.21e+04
Offers	-8267.4883	1084.777	-7.621	0.000	-1.04e+04	-6119.706
Yes	1.73e+04	1981.616	8.729	0.000	1.34e+04	2.12e+04
East	-2.224e+04	2531.758	-8.785	0.000	-2.73e+04	-1.72e+04
North	-2.068e+04	3148.954	-6.568	0.000	-2.69e+04	-1.44e+04

回歸係數

標準差

當虛無假設成立時的t

t發生的機率

95%信心水準估計

參數估計(續)

- 假設顯著性標準是0.01
- 推翻虛無假設的標準是 p 值 < 0.01

	coef	std err	t	P> t
SqFt	52.9937	5.734	9.242	0.000
Bedrooms	4246.7939	1597.911	2.658	0.009
Bathrooms	7883.2785	2117.035	3.724	0.000
Offers	-8267.4883	1084.777	-7.621	0.000
Yes	1.73e+04	1981.616	8.729	0.000
East	-2.224e+04	2531.758	-8.785	0.000
North	-2.068e+04	3148.954	-6.568	0.000

- $t = 2.658$, $P(>t) = 0.009$

驗證兩者關係顯著

檢驗多元回歸模型

■ 如同檢驗簡單回歸模型

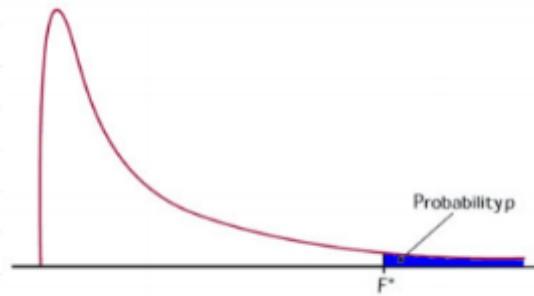
- 檢視模型是否適配資料
- 檢驗每個引數是否都為顯著

■ 檢驗假說

$$H_o : \beta_1 = \beta_2 = \beta_3 = \dots = \beta_k = 0$$

$$H_A : \text{at least one } \beta_k \neq 0$$

■ 使用F統計



計算F統計

- 回歸平方和(regression SS) – 依變數的變化歸咎於回歸模型
- 誤差平方和(error SS) – 依變數的變化非歸咎於線性模型
- 總和平方和 (total SS) – 依變數整體變化

計算F統計 (續)

■ 回歸平方平均(Model Mean Square)

- Regression SS / Regression d.f (k)

- k = 自變數的數量

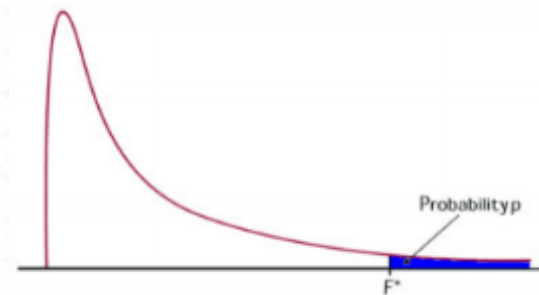
■ 誤差平方平均(Error Mean Square)

- Error SS / Error d.f. ($n-k-1$)

- n = 觀測值的數量

■ F統計(F-Statistic)

- $$F = \frac{\text{Model Mean Square}}{\text{Error Mean Square}}$$

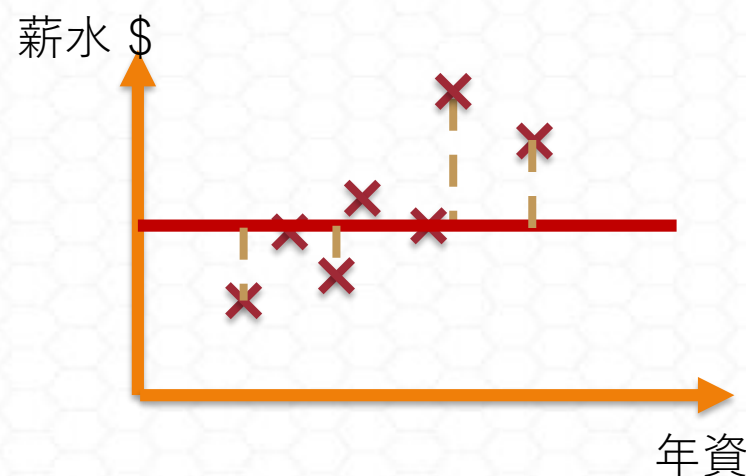
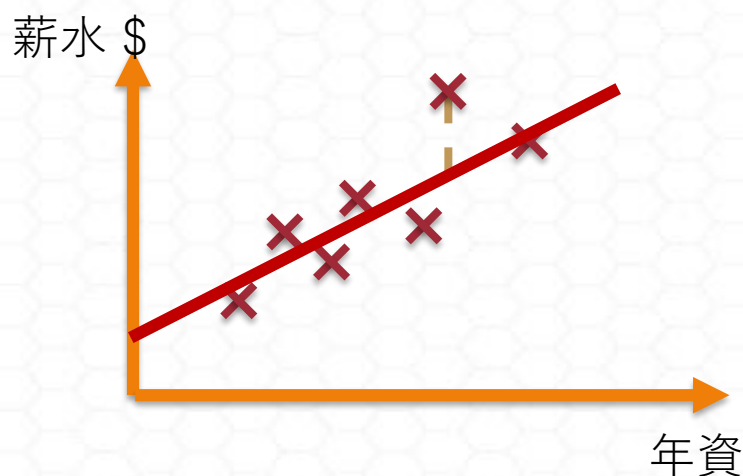


R Squared

- 回歸可以解釋的變異比例，可作為引數預測依變數準確度的指標

殘差平方和 $SSE = \sum_{i=1}^n (y_i - \hat{y}_i)^2$

整體平方和 $SST = \sum_{i=1}^n (y_i - y_{avg})^2$



$$R^2 = 1 - \frac{SSE}{SST}$$

Adjusted R Square

$$R^2 = 1 - \frac{SSE}{SST}$$

SSE 最小 $\Rightarrow R^2$ 永遠不會遞減

$$y_i = \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots \beta_k x_{ki}$$

$$Adj R^2 = 1 - (1 - R^2) \frac{n - 1}{n - p - 1}$$

增加任何一個變數還是會增加 R^2

p = 多少回歸因數
 n = 總體大小

AIC & BIC

- The Akaike information criterion (AIC) & The Bayesian information criterion (BIC)

$$AIC = 2k + n \ln(SSE/n)$$

- 其中： k 是參數的數量， n 為觀察數， SSE 為殘差平方和
- AIC鼓勵資料擬合的優良性但是儘量避免出現過度擬合（Overfitting）的情況。所以優先考慮的模型應是AIC值最小的那一個。赤池信息量準則的方法是尋找可以最好地解釋資料但包含最少自由參數的模型

使用Python 實做stepAIC

```
import itertools
AICs = {}
for k in range(1, len(predictorcols)+1):
    for variables in itertools.combinations(predictorcols, k):
        predictors = X[list(variables)]
        predictors2 = sm.add_constant(predictors)
        est = sm.OLS(Y, predictors2)
        res = est.fit()
        AICs[variables] = res.aic
```

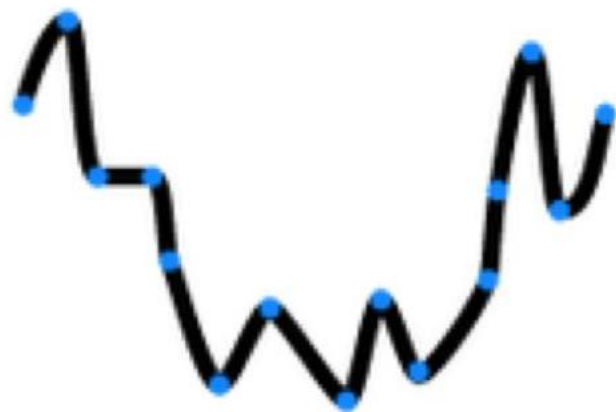
```
from collections import Counter
c = Counter(AICs)
c.most_common()[::-10]
```




正則化 (Regularization)

降低損失函數

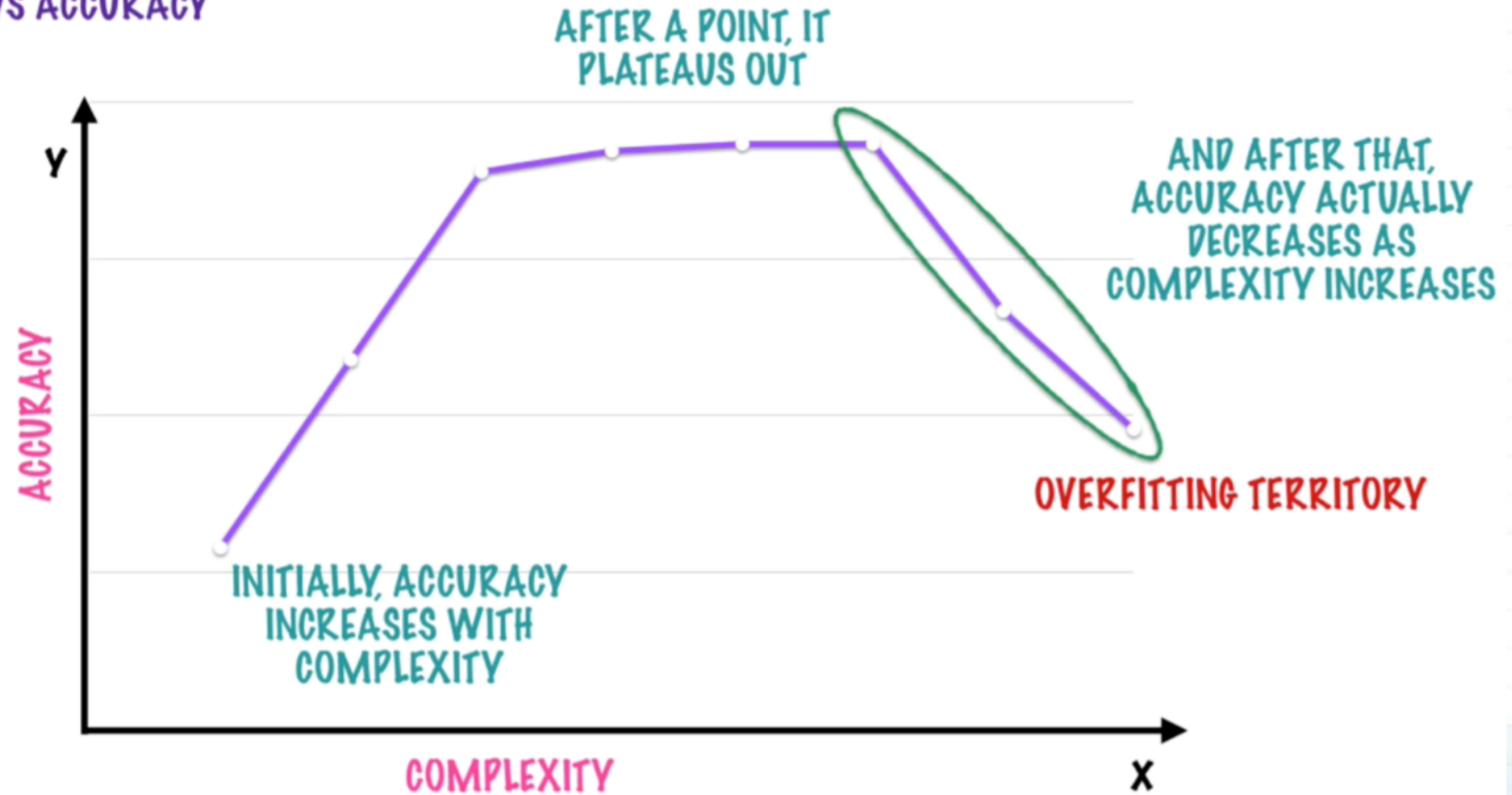
- 機器學習的目的在於降低損失函數 (Minimizing Loss Function)
- 但模型有的時候會建立的過於複雜 (過度適配資料)



Overfitting

過度適配的案例

LET'S SAY YOU PLOTTED COMPLEXITY OF A
MODEL VS ACCURACY



正規化的必要性

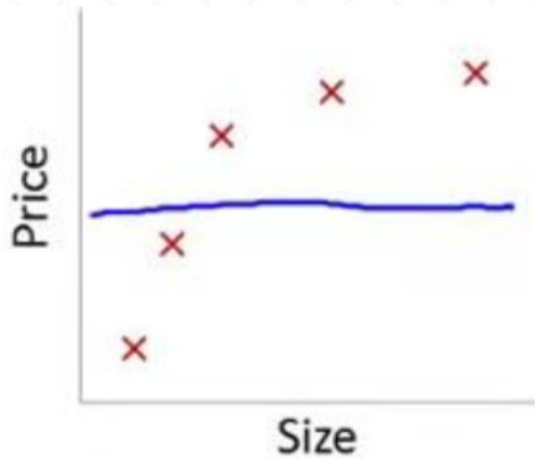
$$E'(f) = E(f) + \lambda R(f)$$

新的 ERROR Function

調整Regularization Term
的重要性

Regularization Term
隨模型複雜度增加

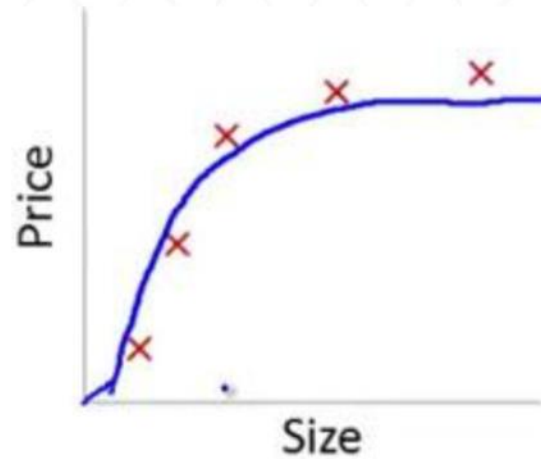
Lambda



Large λ ←

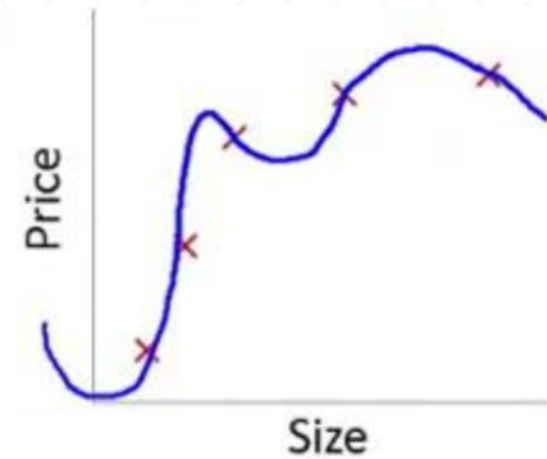
→ High bias (underfit)

$$\lambda = 10000. \quad \theta_1 \approx 0, \theta_2 \approx 0, \dots$$
$$h_{\theta}(x) \approx \theta_0$$



Intermediate λ ←

"Just right"



→ Small λ

High variance (overfit)

$$\rightarrow \lambda = 0$$

L1 正規化 & L2 正規化

■ L1正規化和正規化可以看做是損失函數的懲罰項

■ L1正規化

- 權值向量 w 中各個元素的絕對值之和
- 產生一個稀疏模型，可以用於特徵選擇

■ L2正規化

- 權值向量 w 中各個元素的平方和然後再求平方根
- 防止模型過度適配(overfitting)

$$\lambda \sum_{j=0}^M |W_j|$$

L1 Penalty

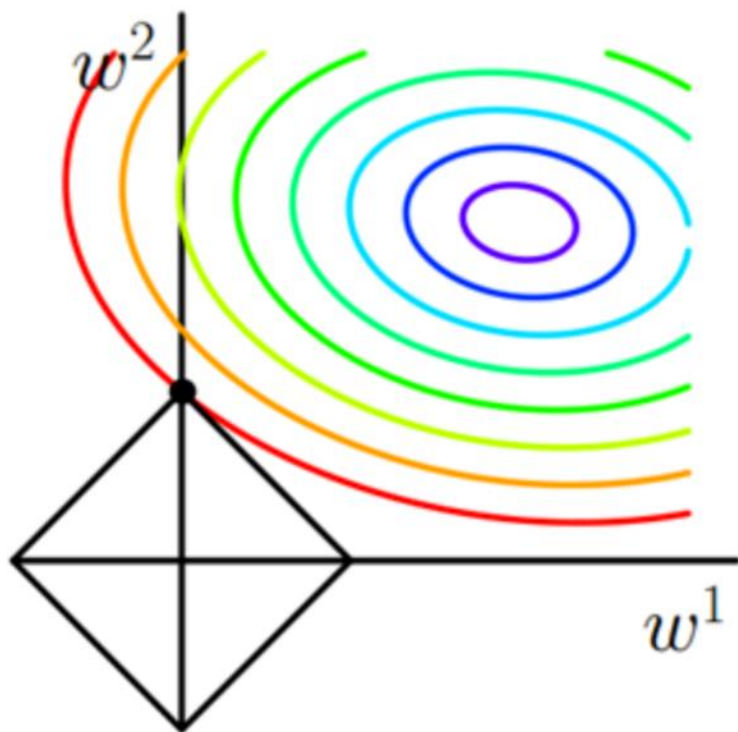
$$\lambda \sum_{j=0}^M W_j^2$$

L2 Penalty

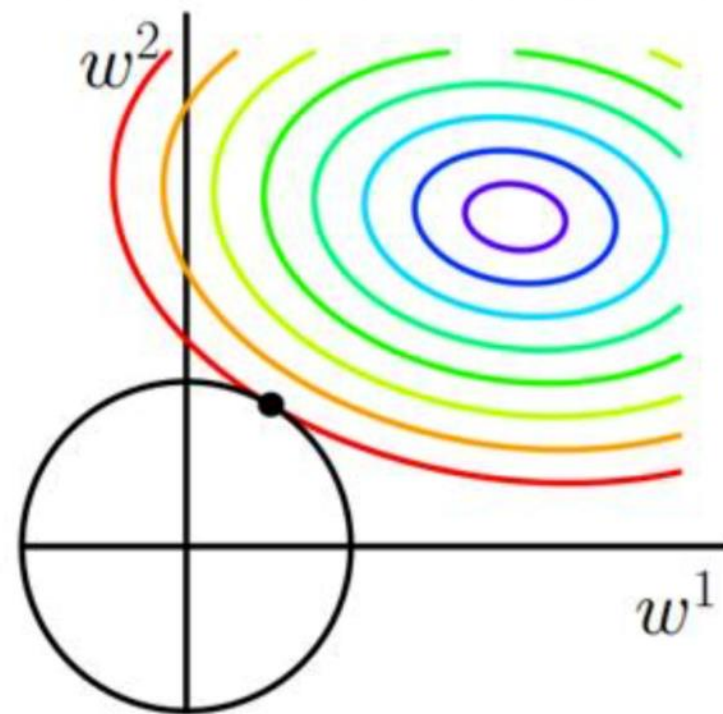
對於線性回歸模型，使用L1正則化的模型建叫做Lasso回歸，使用L2正則化的模型叫做Ridge回歸

L1 v.s. L2 正規化

■ L1正規化

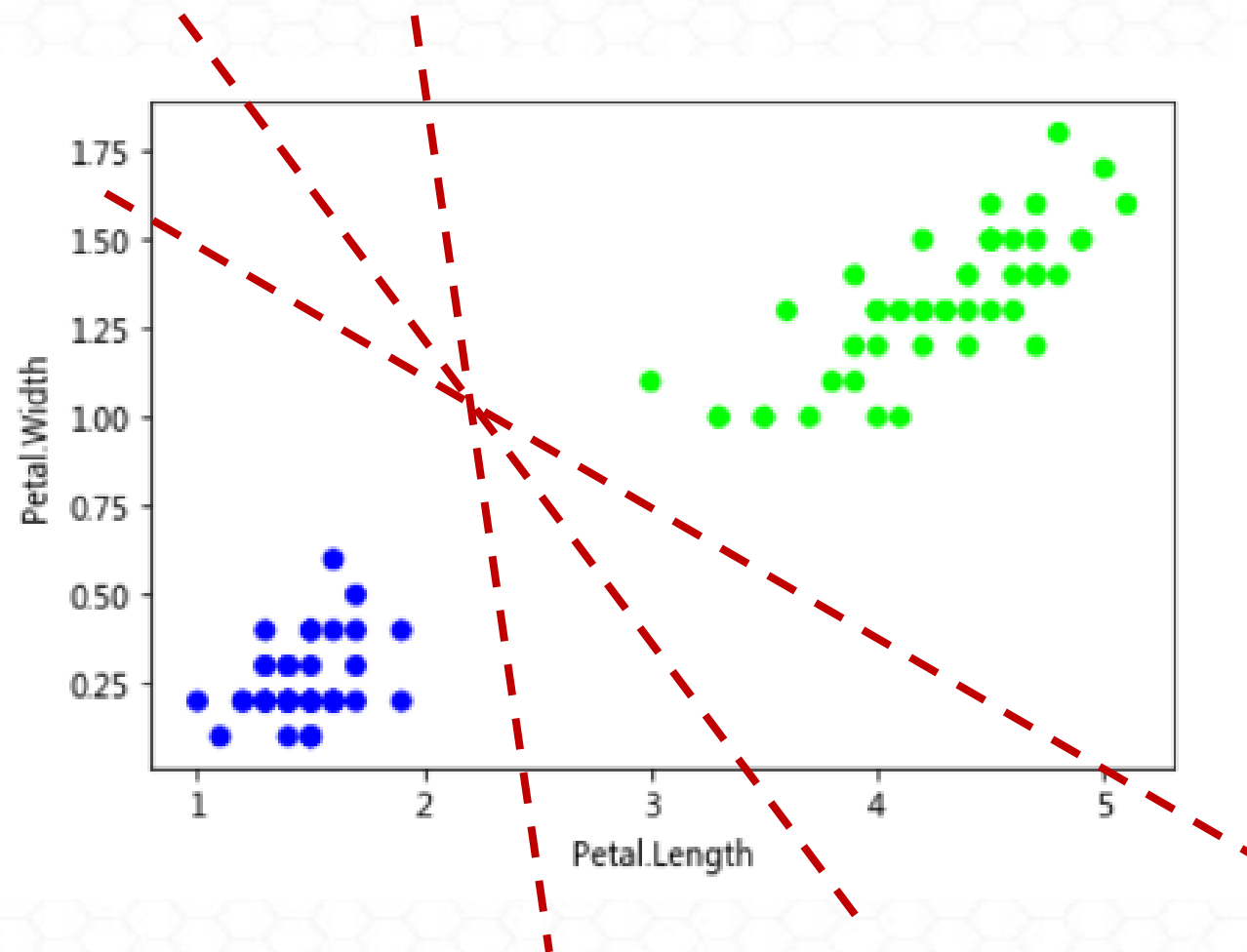


■ L2正規化

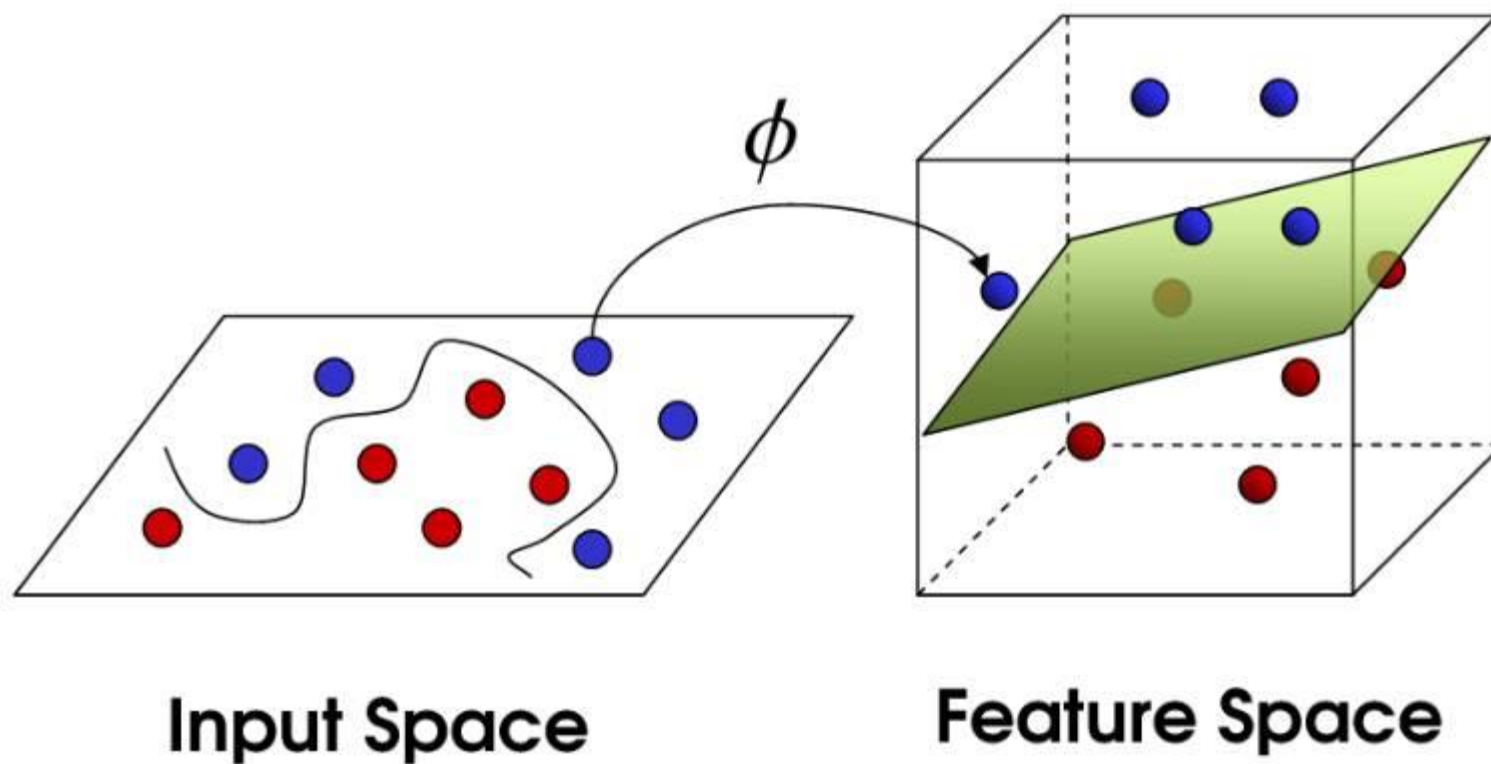


Support Vector Regression

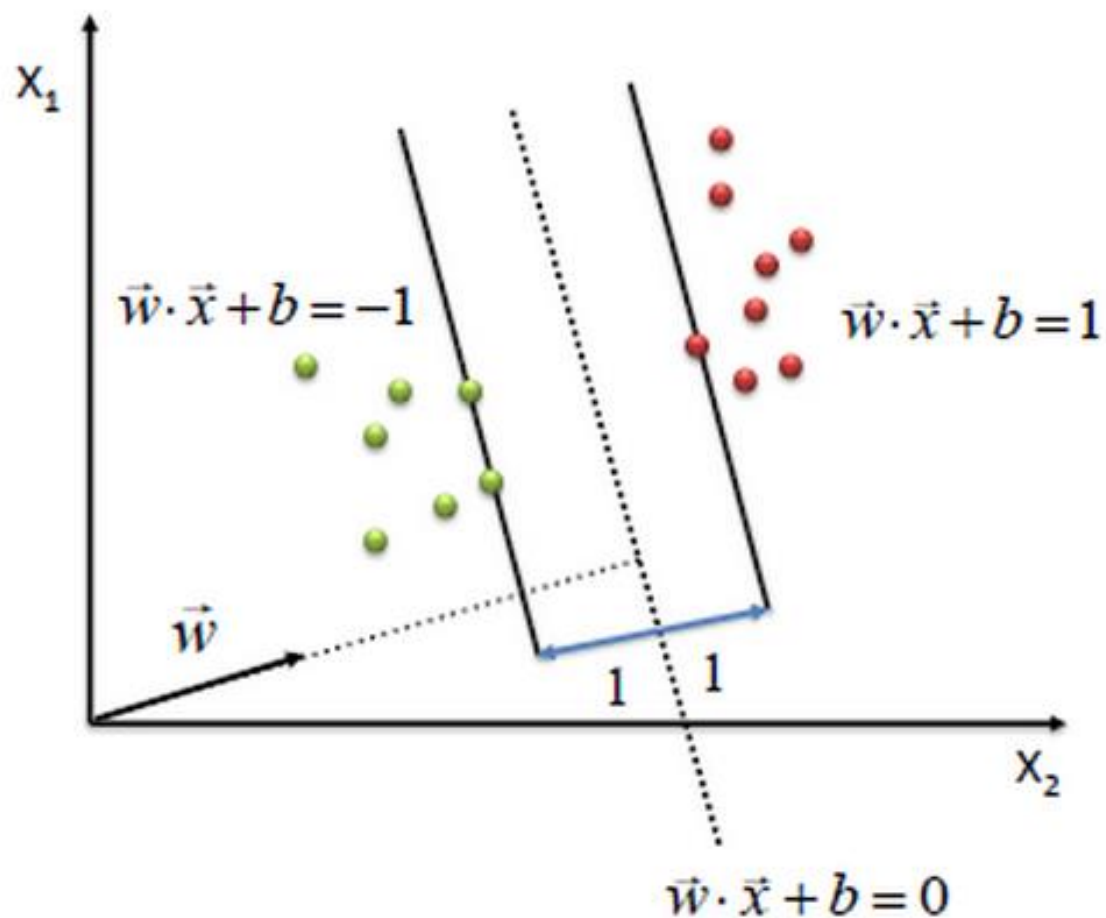
該選哪一條線做切割？



如何解決高維度資料切分問題



支持向量機 (Support Vector Machine)



$$\max \frac{2}{\|\vec{w}\|}$$

s.t.

$$(\vec{w} \cdot \vec{x} + b) \geq 1, \forall \vec{x} \text{ of class 1}$$

$$(\vec{w} \cdot \vec{x} + b) \leq -1, \forall \vec{x} \text{ of class 2}$$

建立支持向量機 (1)

```
from sklearn.datasets import load_iris
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
iris = load_iris()
```

```
X = iris.data[0:100,[2,3]]
y = iris.target[0:100]
```

```
clf1 = SVC(kernel="linear")
clf1.fit(X, y)
```

```
clf2 = LogisticRegression()
clf2.fit(X, y)
```


建立支持向量機 (2)

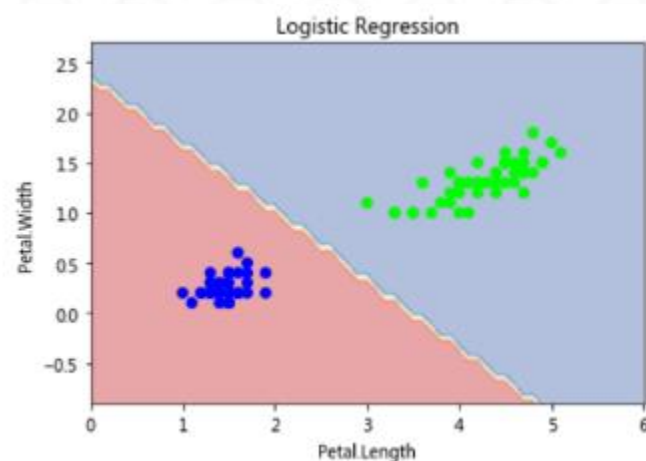
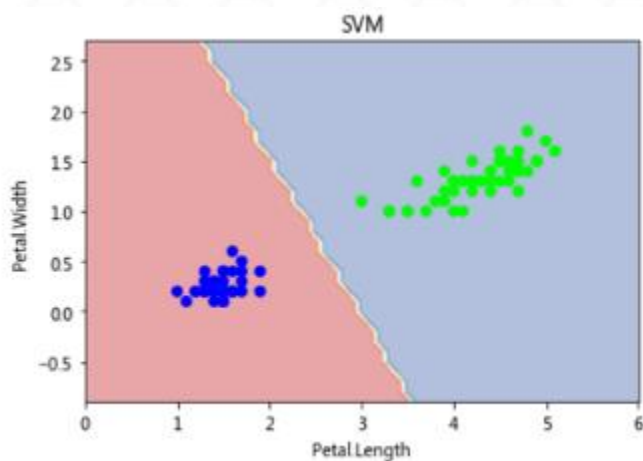
```
def plot_estimator(estimator, X, y):  
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1  
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1  
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),  
                          np.arange(y_min, y_max, 0.1))  
  
    Z = estimator.predict(np.c_[xx.ravel(), yy.ravel()])  
    Z = Z.reshape(xx.shape)  
  
    plt.plot()  
    plt.contourf(xx, yy, Z, alpha=0.4, cmap = plt.cm.RdYlBu)  
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap = plt.cm.brg)  
    plt.xlabel('Petal.Length')  
    plt.ylabel('Petal.Width')  
    plt.show()
```

建立一個繪圖函數

SVM v.s. Logistic Regression

`plot_estimator(clf1, X, y)`

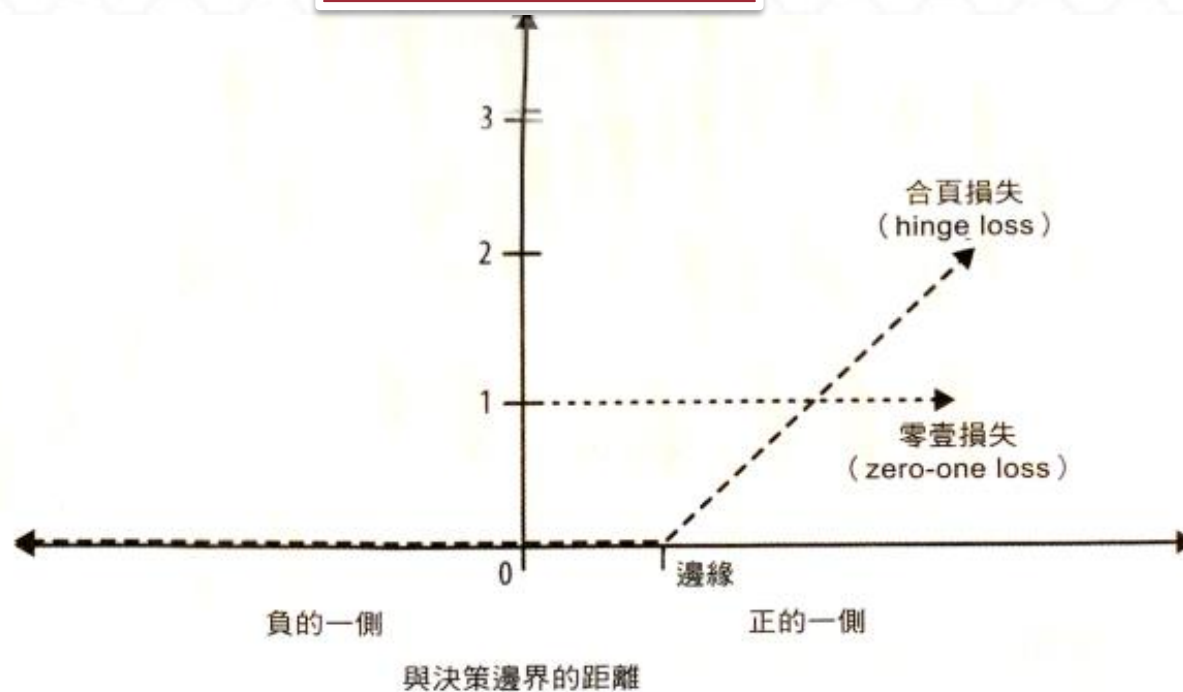
`plot_estimator(clf2, X, y)`



調整Kernel Function 可以做到非線性適配

評估決策邊界

SVM 只會犯小錯



正則項 (Regularization Term)

- 變更正則項 (Regularization Term) C
- 小的 C 可允許界線被忽略 → 寬邊界 (wide margin)
- 大的 C 讓界線難以被忽略 → 窄邊界 (narrow margin)
- $C = \infty$ 必須遵守界線規則，不得越界(hard margin)

設定正則項 (Regularization Term)

```
data = np.array([[ -1,2,0],[ -2,3,0],[ -2,5,0],[ -3,-4,0],[ -0.1,2,0],[0.2,1,1],[0,1,1],[1,2,1],[1,1,1],[ -0.4,0.5,1],[2,5,1]])
```

```
X = data[:, :2]
```

```
Y = data[:,2]
```

```
# Large Margin
```

```
clf = SVC(C=1.0, kernel='linear')
```

```
clf.fit(X, Y)
```

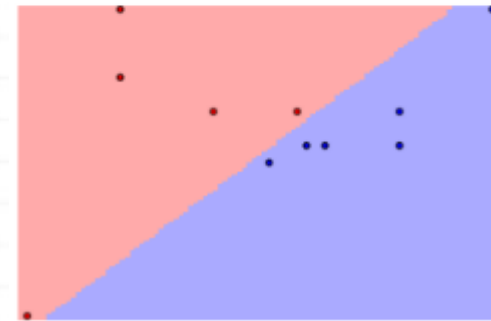
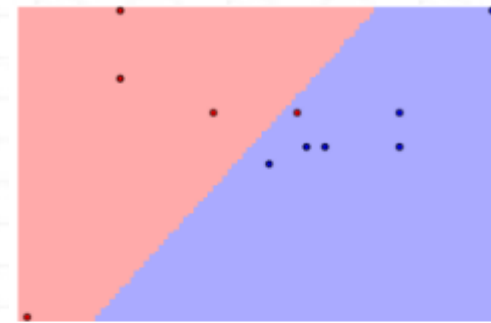
```
plot_estimator(clf,X,Y)
```

```
# Narrow Margin
```

```
clf = SVC(C=100000, kernel='linear')
```

```
clf.fit(X, Y)
```

```
plot_estimator(clf,X,Y)
```



讀取數據

```
from itertools import product  
import numpy as np  
import matplotlib.pyplot as plt
```

```
from sklearn.datasets import load_iris  
from sklearn.svm import SVC
```

```
iris = load_iris()  
X = iris.data[:, [2, 3]]  
y = iris.target
```

比較不同Kernel

```
clf1 = SVC(kernel="rbf")
```

```
clf1.fit(X, y)
```

```
clf2 = SVC(kernel="poly")
```

```
clf2.fit(X, y)
```

```
clf3 = SVC(kernel="linear")
```

```
clf3.fit(X, y)
```

繪製決策邊界

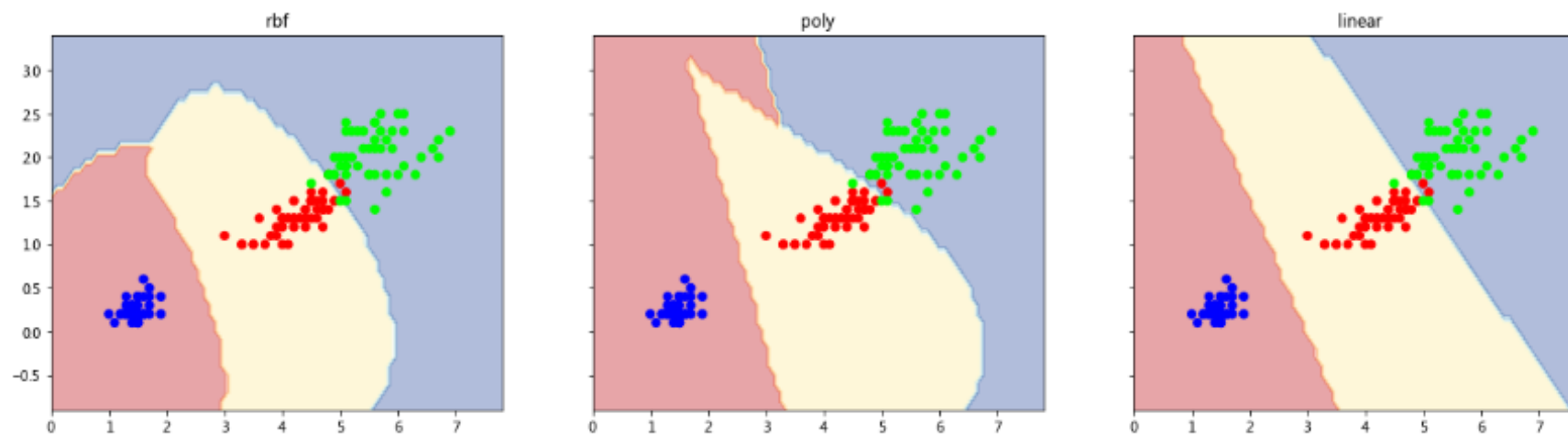
```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

f, axarr = plt.subplots(1, 3, sharex='col', sharey='row', figsize=(20, 5))

for idx, clf, title in zip([0,1,2],[clf1, clf2, clf3], ['rbf', 'poly', 'linear']):
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    axarr[idx].contourf(xx, yy, Z, alpha=0.4, cmap = plt.cm.RdYlBu)
    axarr[idx].scatter(X[:, 0], X[:, 1], c=y, cmap = plt.cm.brg)
    axarr[idx].set_title(title)
```

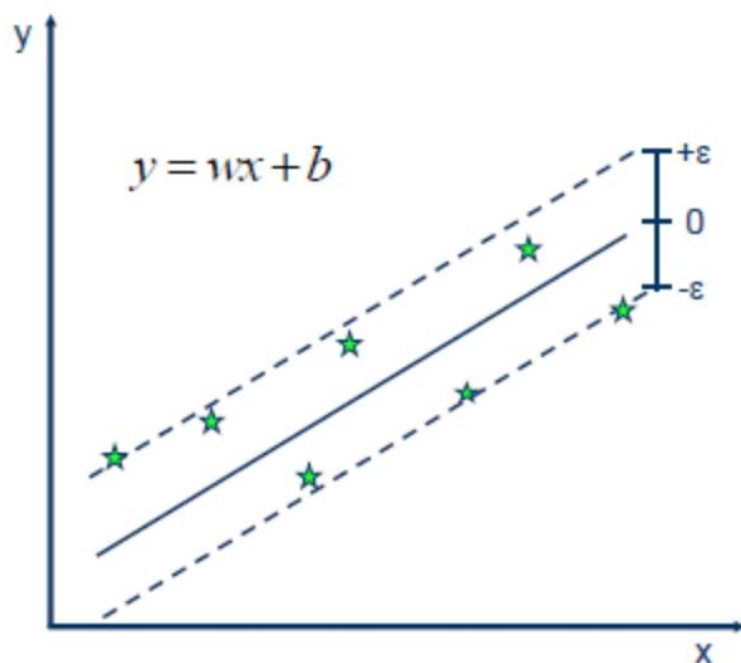

比較不同Kernel Function



Support Vector Regression

■ SVR是本來SVM的延伸型態，能夠處理連續的預測問題

▣ 確定誤差不會超過邊界 (Epsilon)



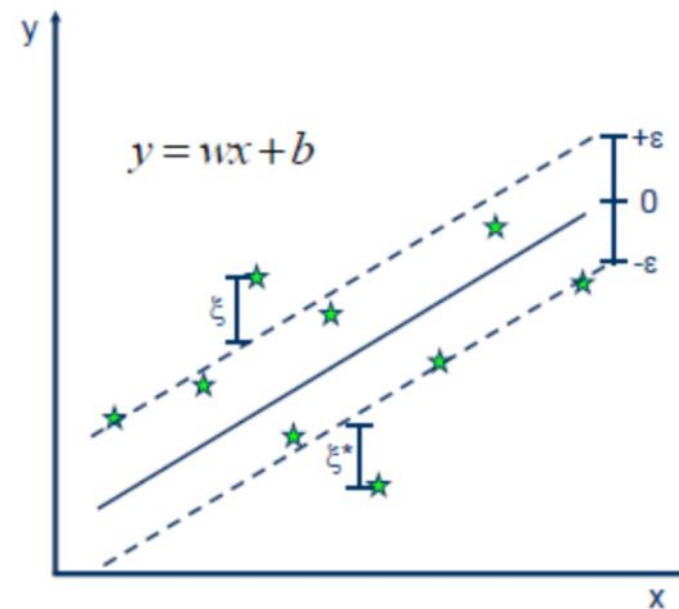
• Solution:

$$\min \frac{1}{2} \|w\|^2$$

• Constraints:

$$y_i - wx_i - b \leq \epsilon$$

$$wx_i + b - y_i \leq \epsilon$$



• Minimize:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^N (\xi_i + \xi_i^*)$$

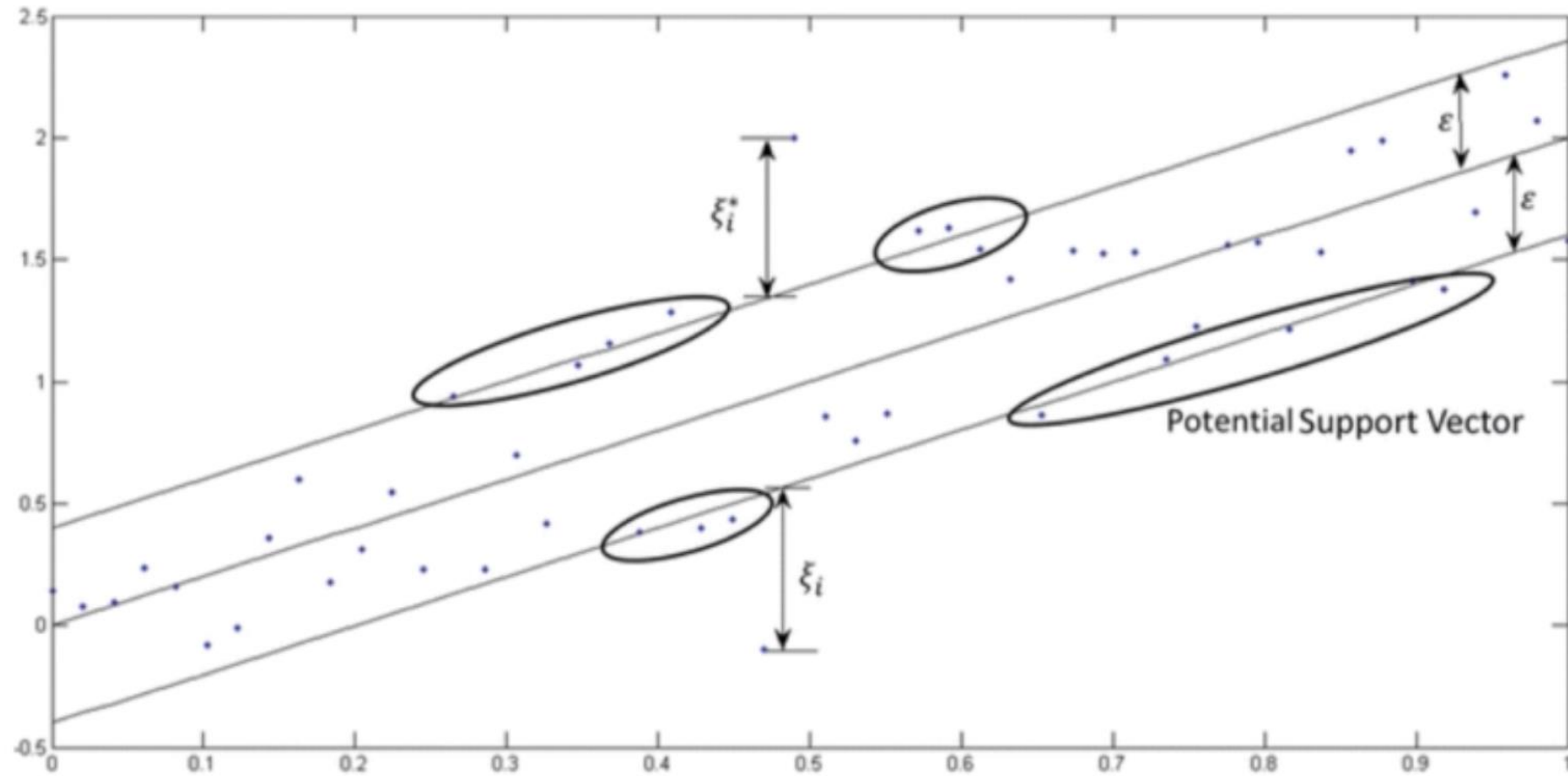
• Constraints:

$$y_i - wx_i - b \leq \epsilon + \xi_i$$

$$wx_i + b - y_i \leq \epsilon + \xi_i^*$$

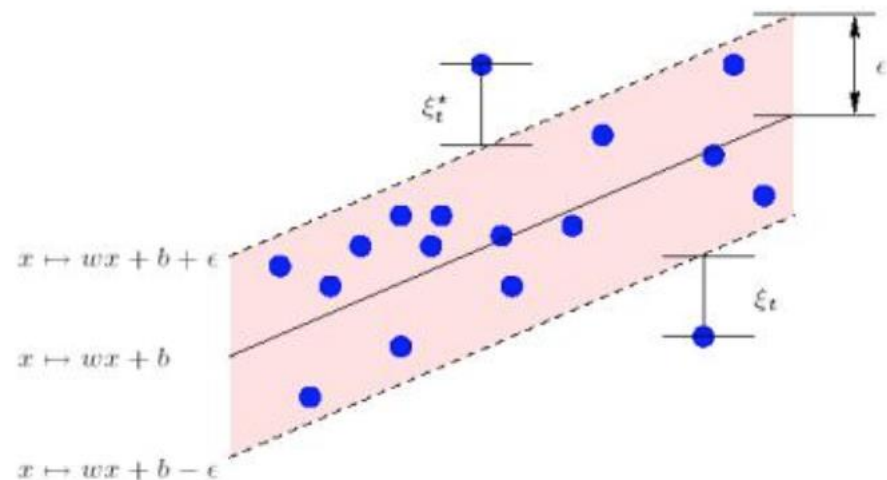
$$\xi_i, \xi_i^* \geq 0$$

Support Vector Regression



Epsilon

- SVR的損失函數中，使用的是epsilon intensive hinge loss。
- Epsilon(ϵ)的概念是給予一個margin of tolerance，在「margin of tolerance」內的資料點會被忽視，它們的殘差(error)一點幫助也沒有，換句話說：它們對訓練SVR一點幫助也沒有。



Epsilon

- Epsilon越大，代表容忍區塊越大，越多資料會被忽視，造成模型的準確度越低，support vectors的數量減少
- Epsilon越低($\rightarrow 0+$)，所有的資料殘差(error)都會被考慮，卻也容易造成overfitting。

SVR v.s. Linear Regression

■ Linear Regression

- 讓預測跟實際值的誤差越小越好

■ SVR

- 確定誤差不會超過邊界

建構SVR 模型

Fitting SVR to the dataset

```
from sklearn.svm import SVR
```

```
regressor = SVR(kernel = 'rbf')
```

```
regressor.fit(X, y)
```

Predicting a new result

```
y_pred = regressor.predict(6.5)
```

```
y_pred = sc_y.inverse_transform(y_pred)
```

預測結果視覺化

```
# Visualising the SVR results
plt.scatter(X, y, color = 'red')
plt.plot(X, regressor.predict(X), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

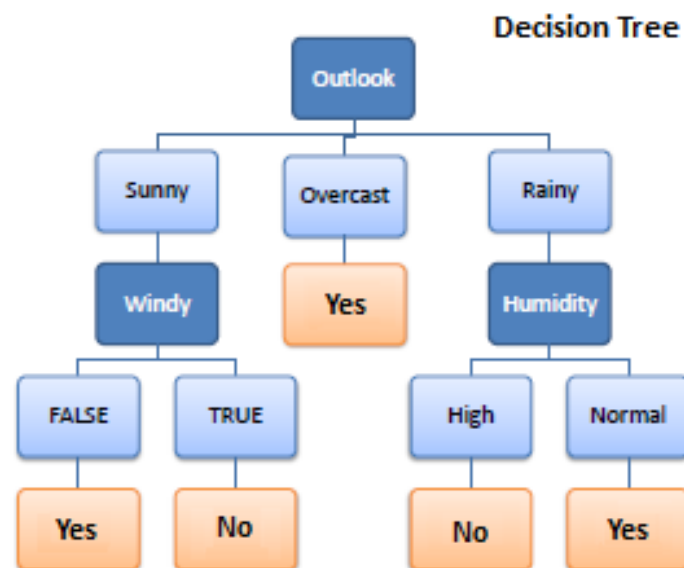

預測結果視覺化

```
# Visualising the SVR results (for higher resolution and smoother curve)
X_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of 0.1 step
because the data is feature scaled
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

Decision Tree Regression

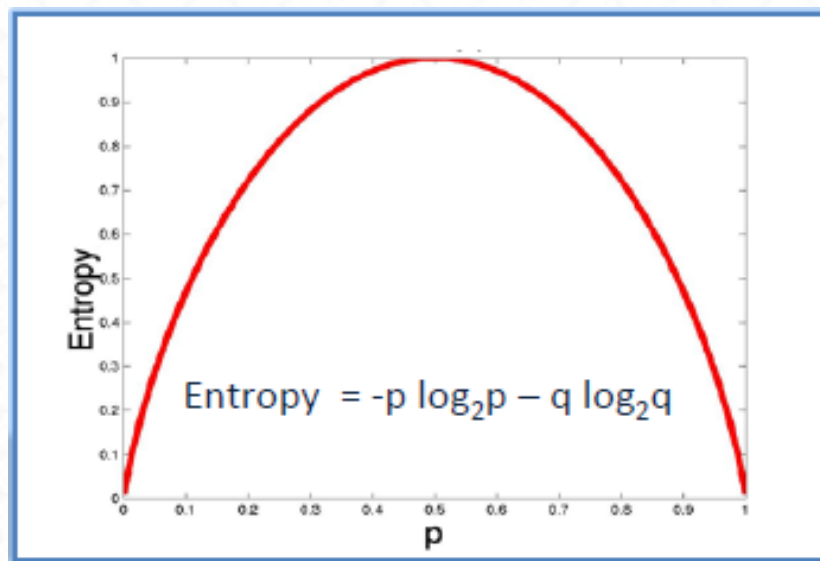
決策樹

Predictors				Target
Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No



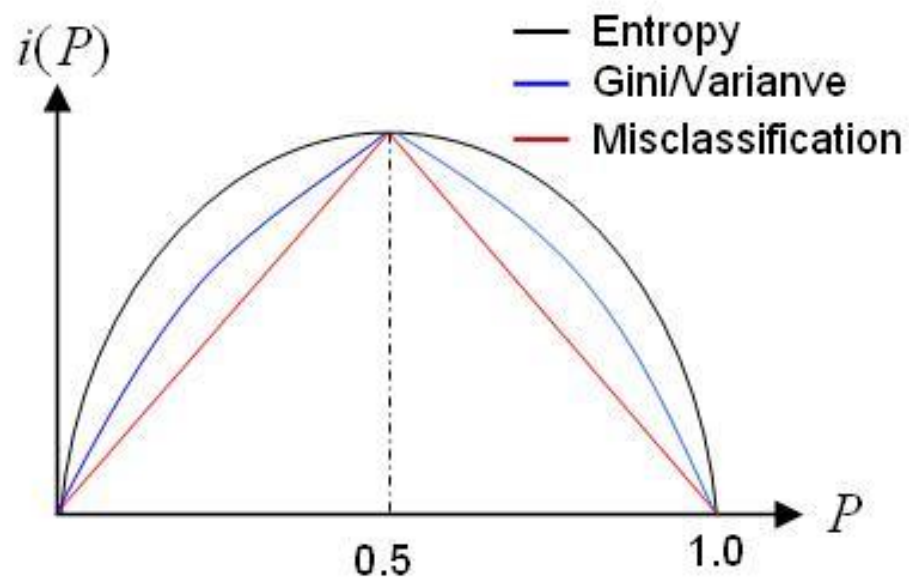
Entropy

- 用於計算一個系統中的失序現象，也就是計算該系統混亂(糾結)的程度



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Gini Impurity



範例:

Prob (晴天) = 0.4

Prob (陰天) = 0.3

Prob (雨天) = 0.3,

$$\text{Gini Index} = 1 - \sum_j p_j^2$$

$$\text{Gini Index} = 1 - (0.4^2 + 0.3^2 + 0.3^2) = 0.660$$

單一變數的計算

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Play Golf	
Yes	No
9	5



Entropy(PlayGolf) = Entropy (5,9)
= Entropy (0.36, 0.64)
= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64)
= 0.94

多變數的計算

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14



$$\begin{aligned} E(\text{PlayGolf}, \text{Outlook}) &= P(\text{Sunny}) * E(3,2) + P(\text{Overcast}) * E(4,0) + P(\text{Rainy}) * E(2,3) \\ &= (5/14) * 0.971 + (4/14) * 0.0 + (5/14) * 0.971 \\ &= 0.693 \end{aligned}$$

Information Gain

- 根據分割(Split)後，所減少的Entropy
- 因此做分割時，會尋找最大的Information Gain

1. 計算E

$$\text{Entropy(PlayGolf)} = \text{Entropy}(5,9)$$

$$= \text{Entropy}(0.36, 0.64)$$

$$= - (0.36 \log_2 0.36) - (0.64 \log_2 0.64)$$

$$= 0.94$$

計算Information Gain

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			


		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

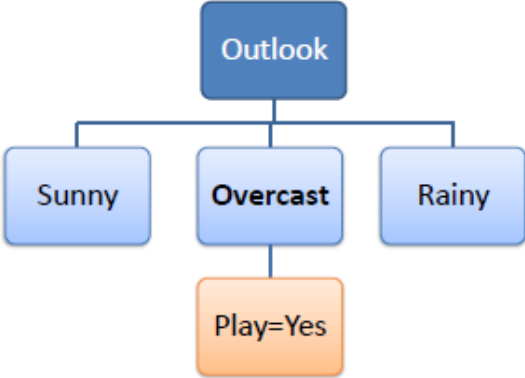
$$\begin{aligned} G(\text{PlayGolf}, \text{Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf}, \text{Outlook}) \\ &= 0.940 - 0.693 = 0.247 \end{aligned}$$

選擇有最大Information Gain的屬性

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

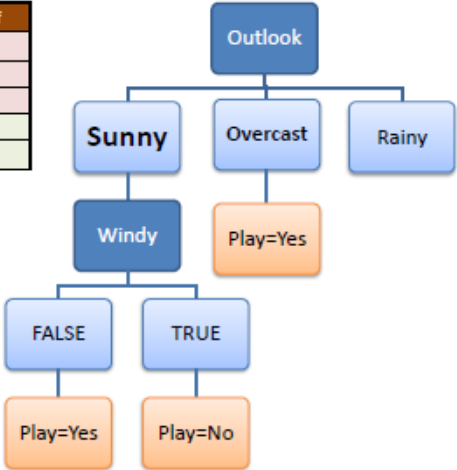
選擇子節點與分割節點

Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes
Hot	High	FALSE	Yes



Entropy 為 0
則為子節點

Temp	Humidity	Windy	Play Golf
Mild	High	FALSE	Yes
Cool	Normal	FALSE	Yes
Mild	Normal	FALSE	Yes
Cool	Normal	TRUE	No
Mild	High	TRUE	No



Entropy 非 0
則為分割節點

決策樹如同IF...ELSE

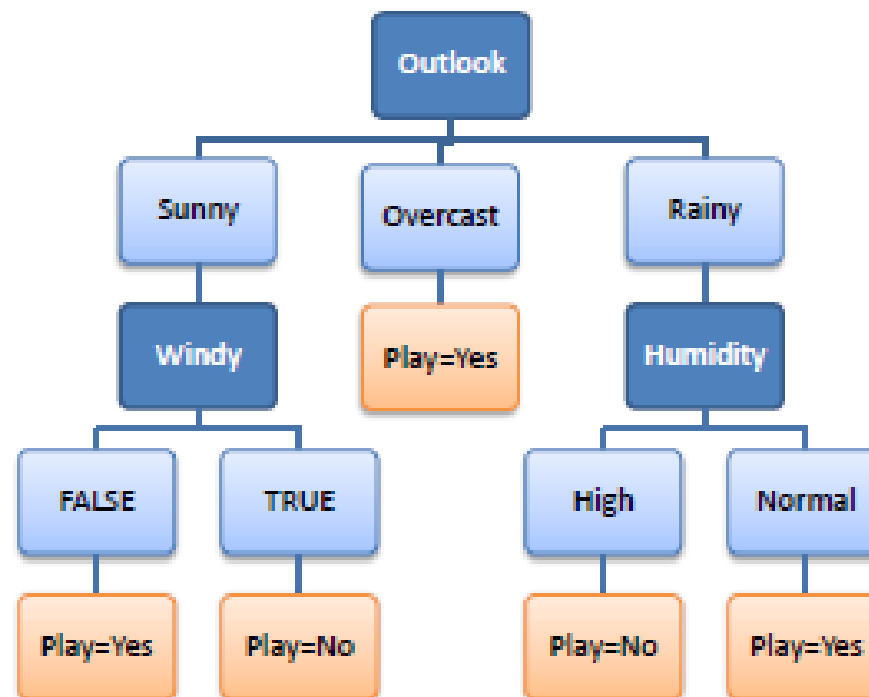
R_1 : IF (Outlook=Sunny) AND
(Windy=FALSE) THEN Play=Yes

R_2 : IF (Outlook=Sunny) AND
(Windy=TRUE) THEN Play=No

R_3 : IF (Outlook=Overcast) THEN
Play=Yes

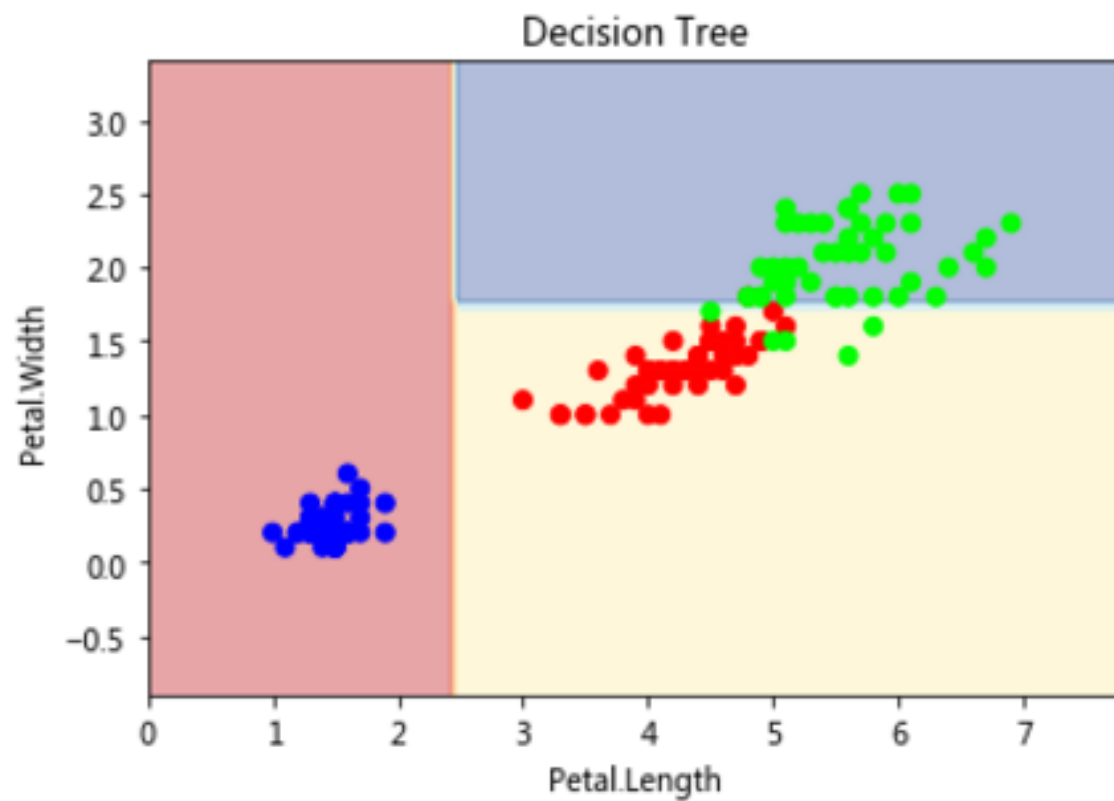
R_4 : IF (Outlook=Rainy) AND
(Humidity=High) THEN Play=No

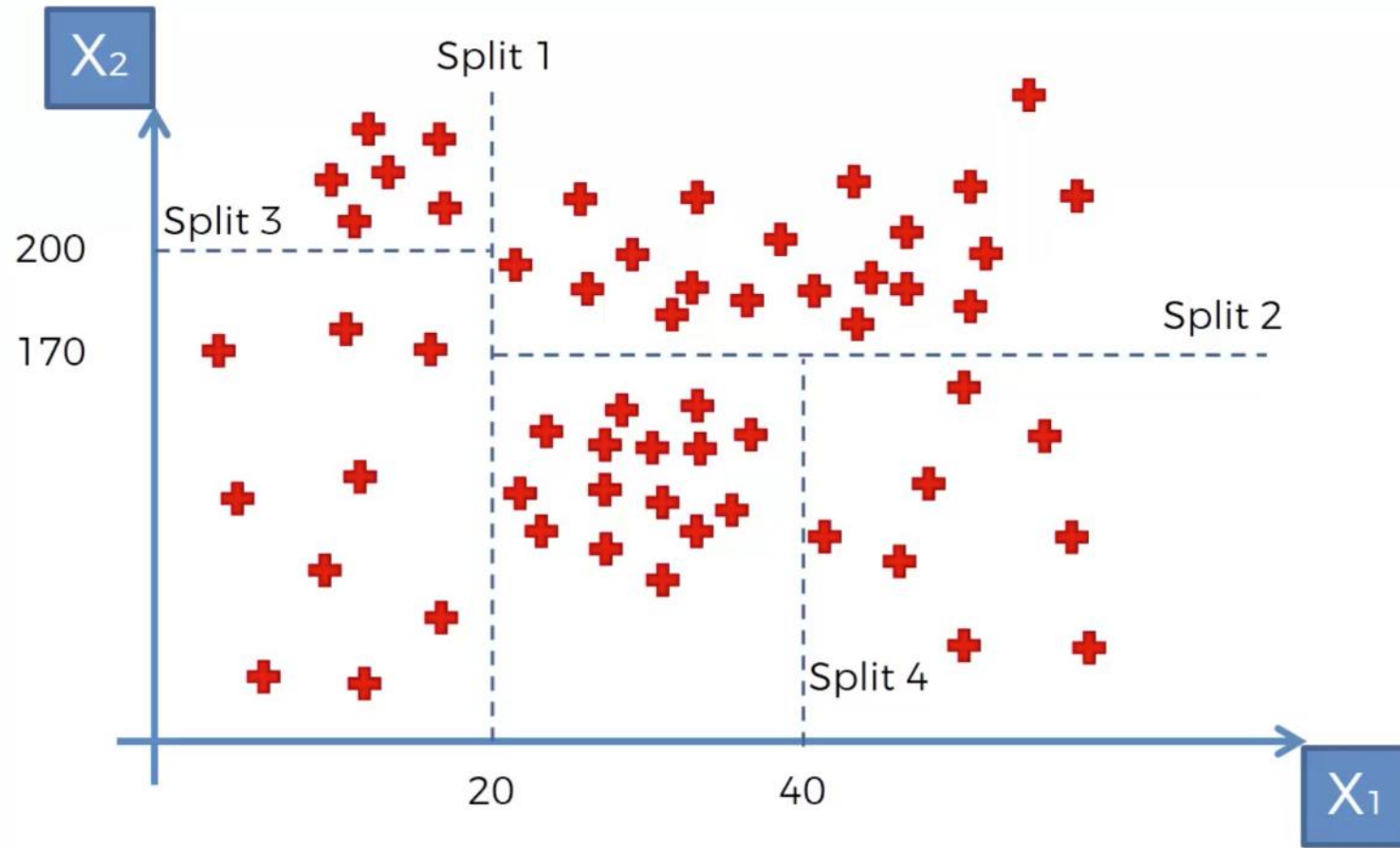
R_5 : IF (Outlook=Rain) AND
(Humidity=Normal) THEN
Play=Yes

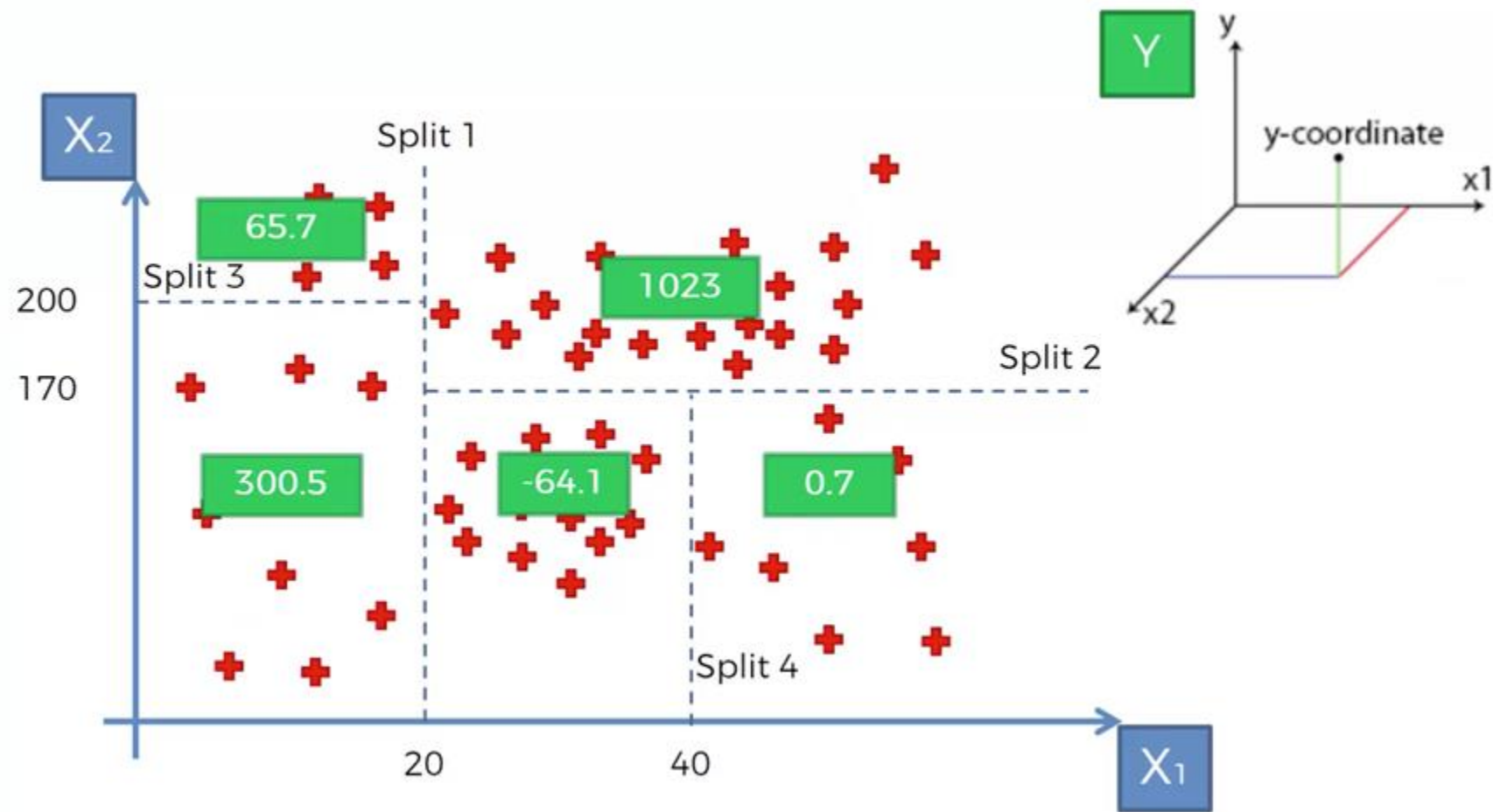


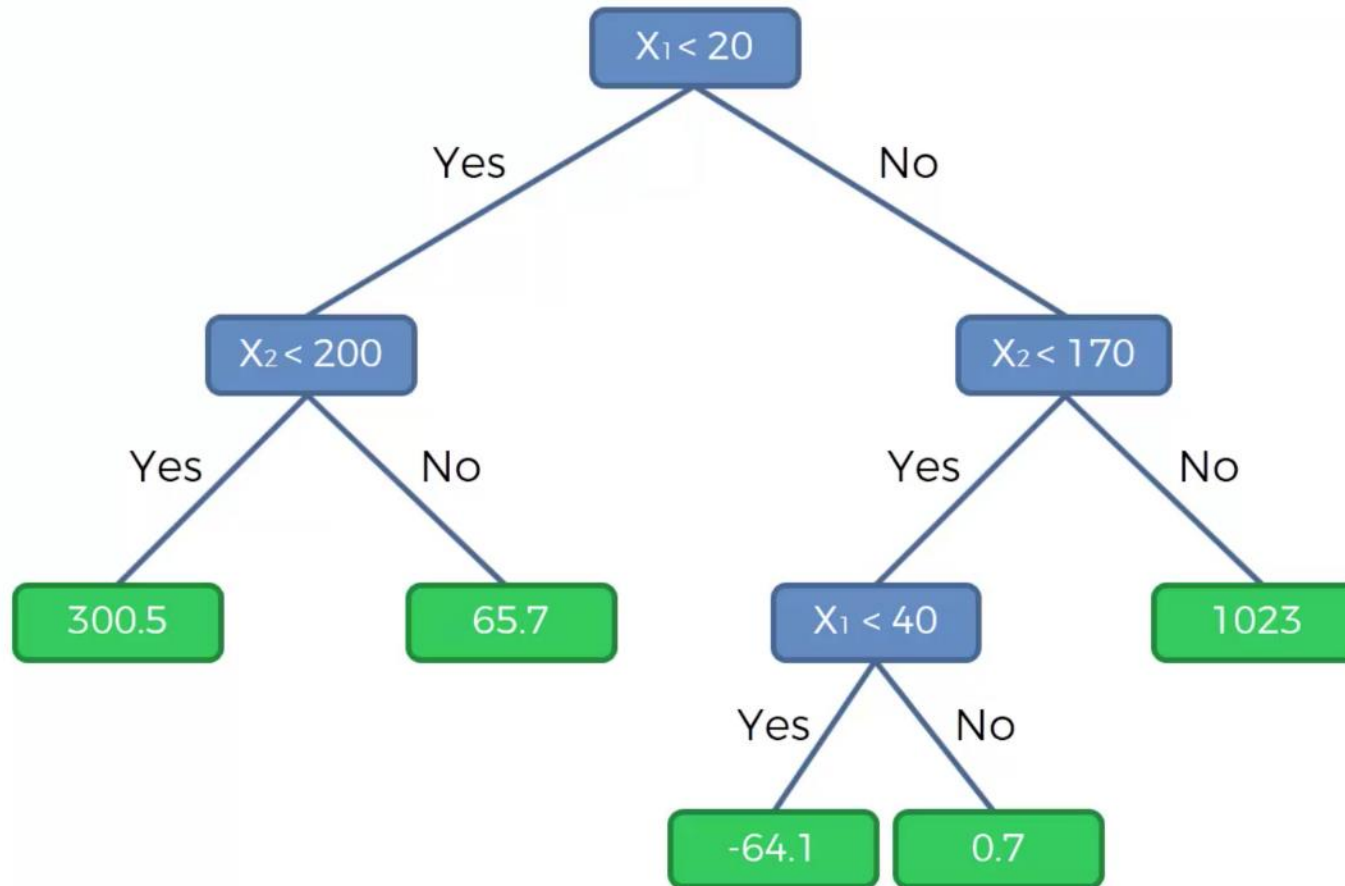
決策樹模型

- 每次根據一個變數取決分隔









讀取資料

Importing the libraries

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import pandas as pd
```

Importing the dataset

```
dataset = pd.read_csv('Position_Salaries.csv')
```

```
X = dataset.iloc[:, 1:2].values
```

```
y = dataset.iloc[:, 2].values
```

資料預處理

Splitting the dataset into the Training set and Test set

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

Feature Scaling

```
from sklearn.preprocessing import StandardScaler  
sc_X = StandardScaler()  
sc_y = StandardScaler()  
X = sc_X.fit_transform(X)  
y = sc_y.fit_transform(y)
```

建立決策樹模型

```
# Fitting Decision Tree Regression to the dataset
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)

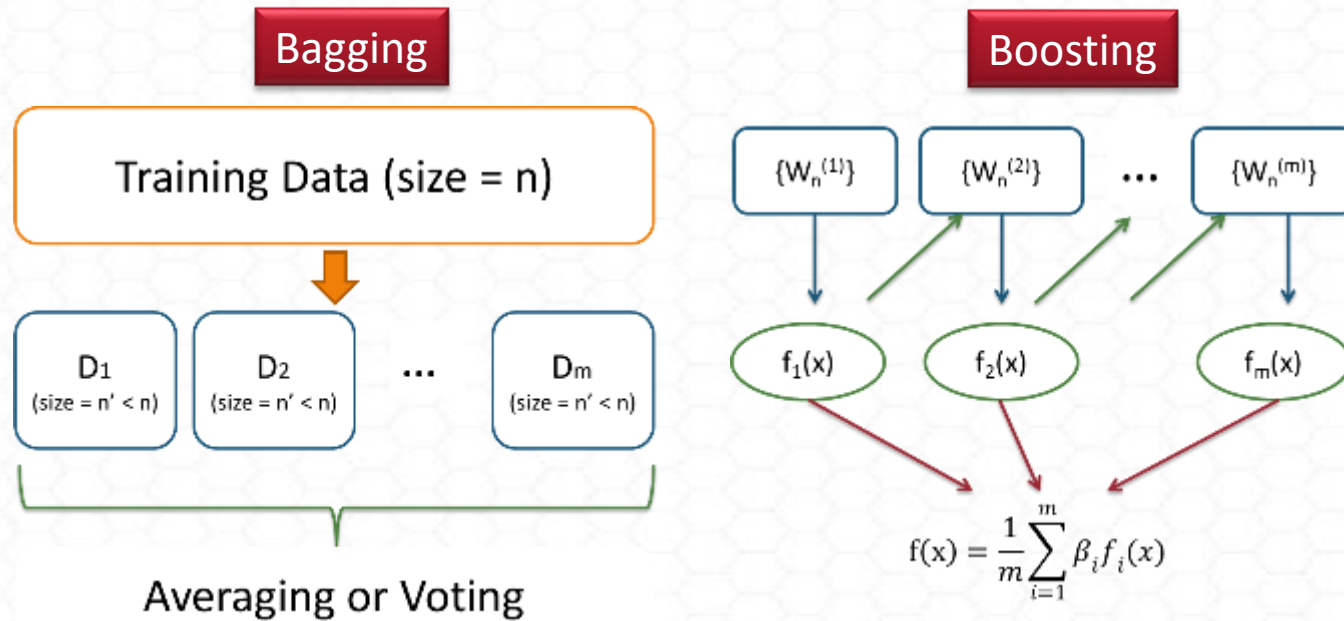
# Predicting a new result
y_pred = regressor.predict(6.5)
```

資料視覺化

```
# Visualising the Decision Tree Regression results (higher resolution)
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Decision Tree Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

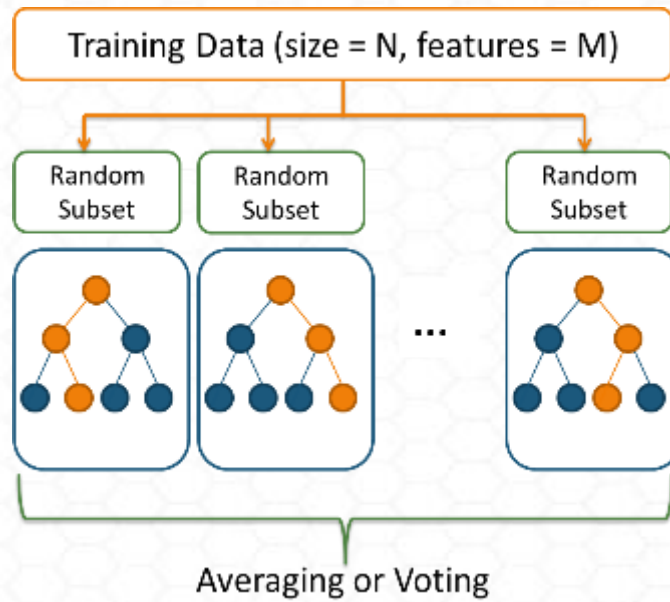

Random Forest Regression

集成機器學習 (Ensemble Learning)



隨機森林 (Random Forest)

■ N 多少樹, M 多少個特徵



建構Random Forest 模型

Fitting Random Forest Regression to the dataset

```
from sklearn.ensemble import RandomForestRegressor
```

```
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
```

```
regressor.fit(X, y)
```

Predicting a new result

```
y_pred = regressor.predict(6.5)
```


預測結果視覺化

```
# Visualising the Random Forest Regression results (higher resolution)
X_grid = np.arange(min(X), max(X), 0.01)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (Random Forest Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

模型綜合比較

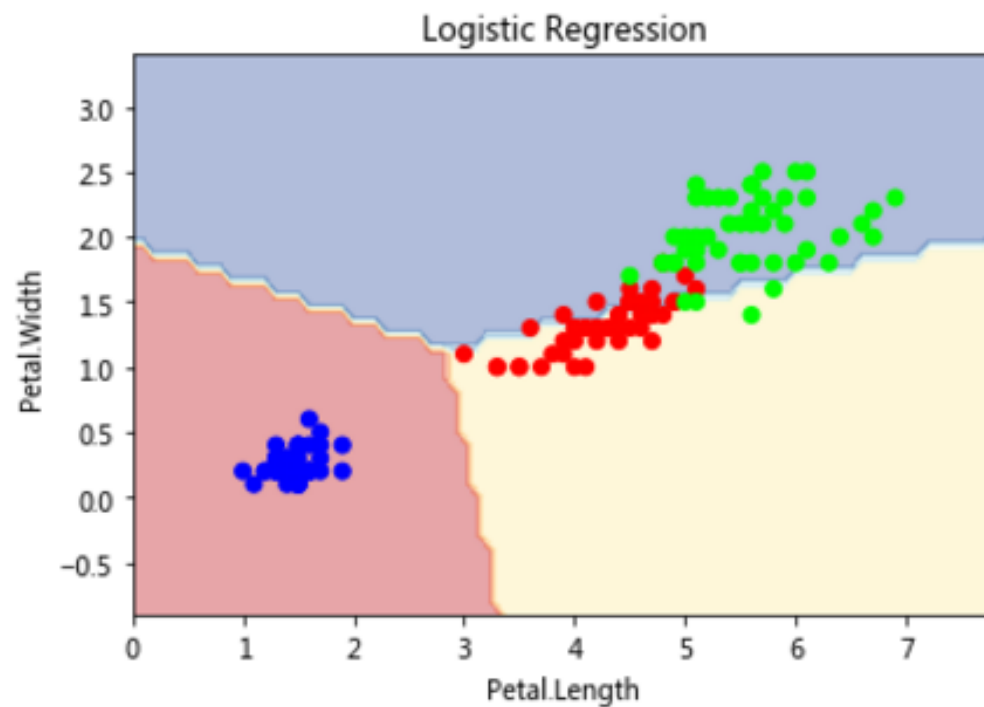
Regression Model	Pros	Cons
Linear Regression	Works on any size of dataset, gives informations about relevance of features	The Linear Regression Assumptions
Polynomial Regression	Works on any size of dataset, works very well on non linear problems	Need to choose the right polynomial degree for a good bias/variance tradeoff
SVR	Easily adaptable, works very well on non linear problems, not biased by outliers	Compulsory to apply feature scaling, not well known, more difficult to understand
Decision Tree Regression	Interpretability, no need for feature scaling, works on both linear / nonlinear problems	Poor results on too small datasets, overfitting can easily occur
Random Forest Regression	Powerful and accurate, good performance on many problems, including non linear	No interpretability, overfitting can easily occur, need to choose the number of trees

Logistic Regression

線性分類法

■ 線性判別

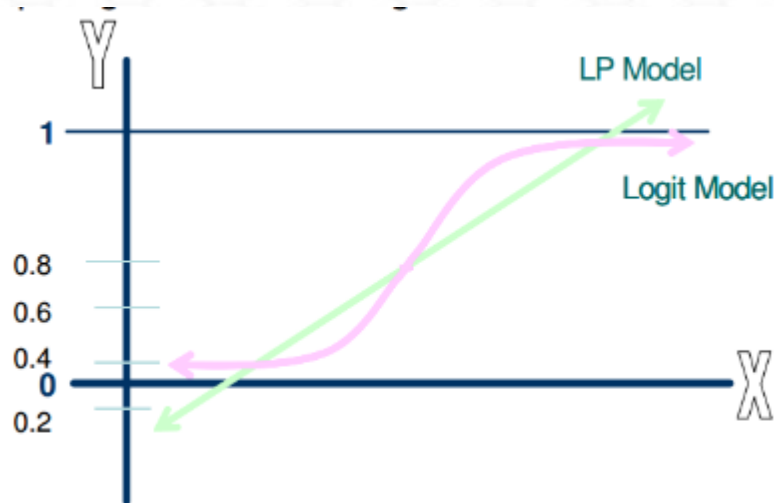
□ 如何切斜的刀



邏輯回歸分析 (Logistic Regression)

■ 從對連續依變數的預測轉變為二元的結果(是/否)

- ▣ 客戶是否流失?
- ▣ 客戶是否買單?
- ▣ 腫瘤為良性還惡性?



如果是線性回歸
會不會X值越大會得到
>100% 的預測結果?

邏輯回歸分析 (Logistic Regression)

■ 定義

Logistic Regression

$$\logit(y) = \ln(odds) = b_0 + b_1 X_1 + \varepsilon$$

■ Odds

□ Odds

= Probability of event for success (PE)/ failure

= PE/(1-PE)

■ 推導

$$e^{\ln(odds)} = odds = e^{(b_0 + b_1 X_1 + \varepsilon_i)}$$

$$PE = odds / (1 + Odds) = e^{(b_0 + b_1 X_1 + \varepsilon_i)} * \frac{1}{1 + e^{(b_0 + b_1 X_1 + \varepsilon_i)}}$$

單純代表獲勝/失敗的機率

建立邏輯回歸分析模型

```
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
iris = load_iris()
clf = LogisticRegression()
clf.fit(iris.data, iris.target)

clf.predict(iris.data)
```

建立決策邊界 (1)

```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression

iris = load_iris()
X = iris.data[:, [2, 3]]
y = iris.target

clf = LogisticRegression()
clf.fit(X, y)
```


建立決策邊界 (2)

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
```

```
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
```

```
plt.plot()
plt.contourf(xx, yy, Z, alpha=0.4, cmap = plt.cm.RdYlBu)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap = plt.cm.brg)
plt.title('Logistic Regression')
plt.xlabel('Petal.Length')
plt.ylabel('Petal.Width')
plt.show()
```

點估計與信賴區間

平均數的抽樣分佈

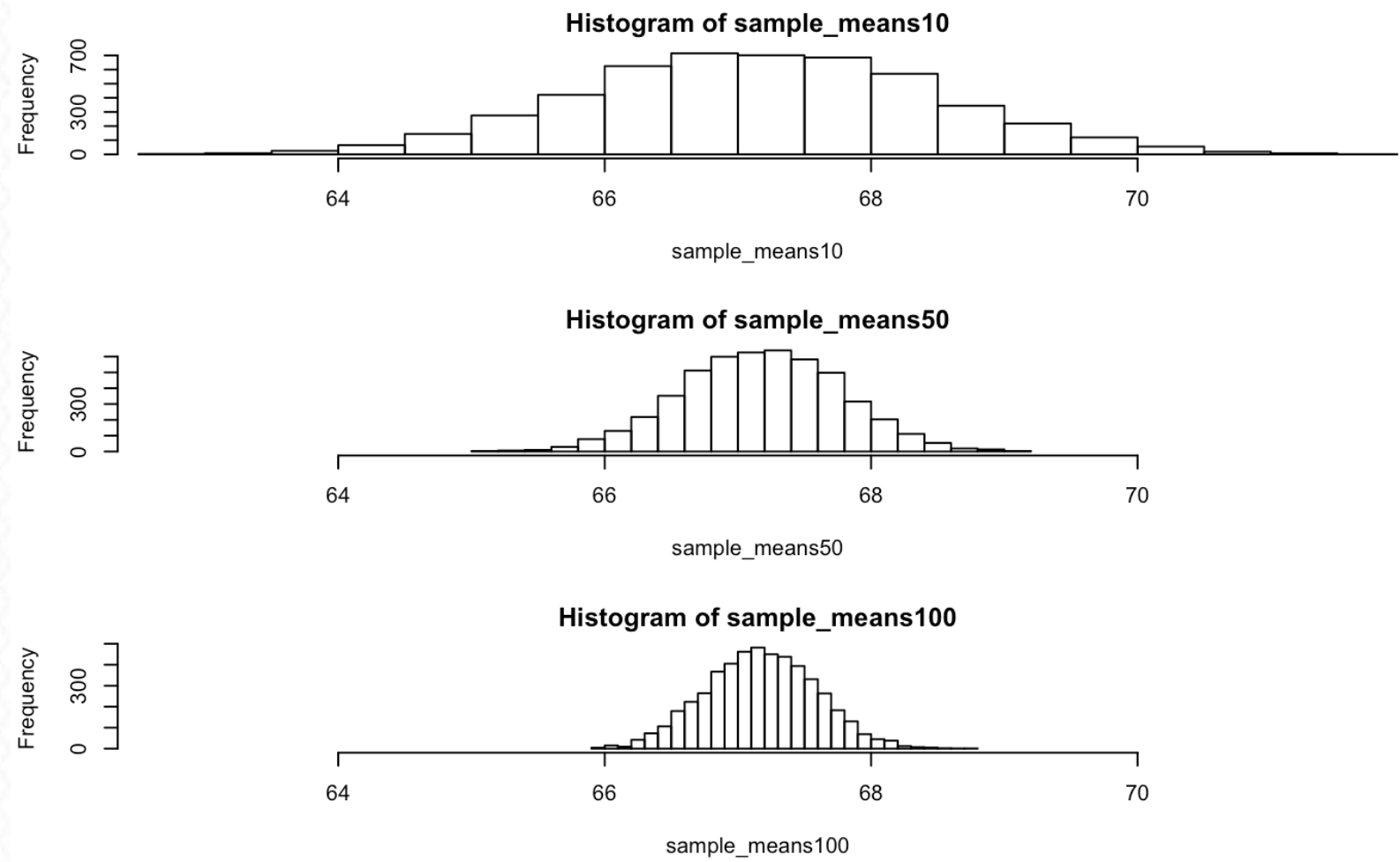
- 生產手機的廠商，希望不用檢查每一隻手機，就知道平均手機的耗電量
 - 隨機取出 n 支手機，度量耗電量
- 抽樣分佈
 - 對特定樣本數 N ，算術平均數的機率分佈
 - 對任意樣本數，取樣的算術平均數亦為常態分佈
 - 中央極限定理

取樣分佈(Sampling Distribution)

■ 中央極限定理 Central Limit Theorem

- 從任何一個母群(算術平均數為 μ ，標準差為 σ)中取大小為 N 之樣本，當 N 足夠大時，取樣的算術平均數會接近常態分佈
- 要知道平均數的分佈，只要知道母體平均數與標準差即可
- 不同種的分佈（常態、均勻或布瓦松分配）只要平均數和標準差相同，平均數的分佈都為常態分佈

不同取樣值產生的直方圖



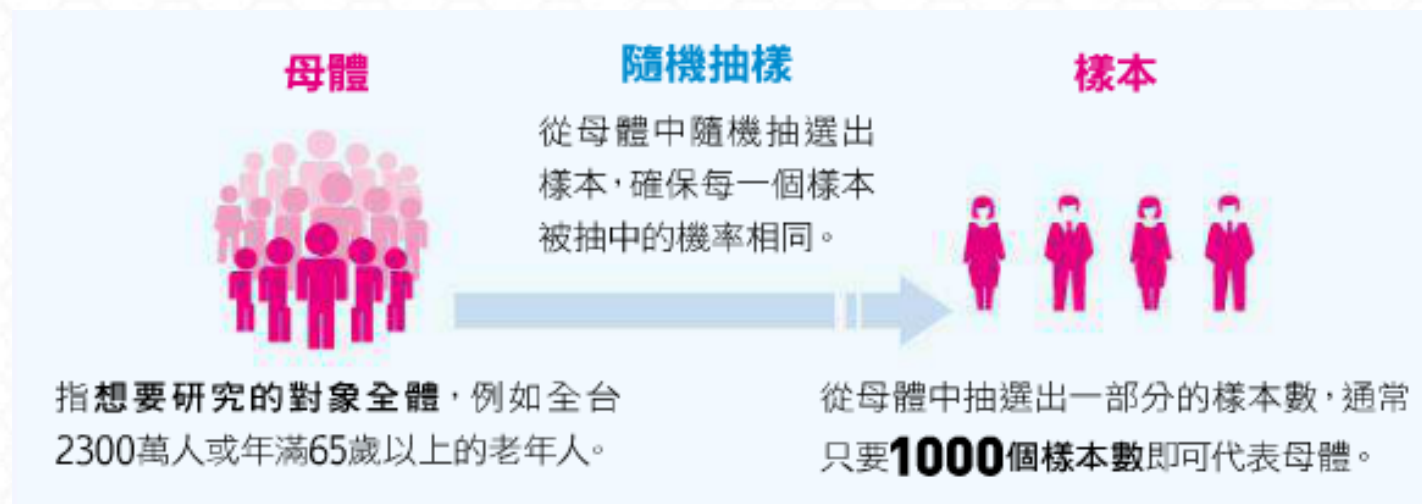
信賴區間(Confidence Interval)

- 為想了解所有民眾的身高分布，通常會從兩個基本資訊著手
 - 一般民眾大概身高多高？
 - 人與人之間的身高變異數有多大？

但如果人口很多, 沒有辦法一一調查呢？

抽樣調查

- 誤差：推論會產生誤差，要得到比較小的誤差，樣本數就比較大
- 信心水準：有多大的信心可以用樣本推論母體，通常設定在95或99%



95%信心水準

- 有95%的機率，會產生一個包含 p 在內的區間 \Rightarrow 95%的時候， p 會落在以觀測值 \hat{p} 為中心的區間
- 假設從母體中隨機抽出1,000名消費者，如果發現80%消費者對A產品滿意在95%信心水準下，誤差範圍為正負3%
 - 代表針對該產品重複進行100次調查，有95次消費者對A產品的滿意比例介於77% ~ 83%

標準誤差 Standard Error

- 標準誤差即樣本統計量的標準差
 - 樣本的平均數和總體真實平均數的偏離程度
- 每次抽樣都能得到一個樣本均值
 - 如果你抽樣的樣本數 N 很大，那麼相對你的樣本均值和總體均值的偏離程度就比較小(如果你的 N 接近無窮大那麼偏離程度趨近0)
 - 而反之，如果樣本數 N 很小，它和真正總體的均值的偏離程度是相對較大

Z-Score

- 是藉由從單一（原始）分數中減去母體的平均值，再依照母體（母集合）的標準差分割成不同的差距

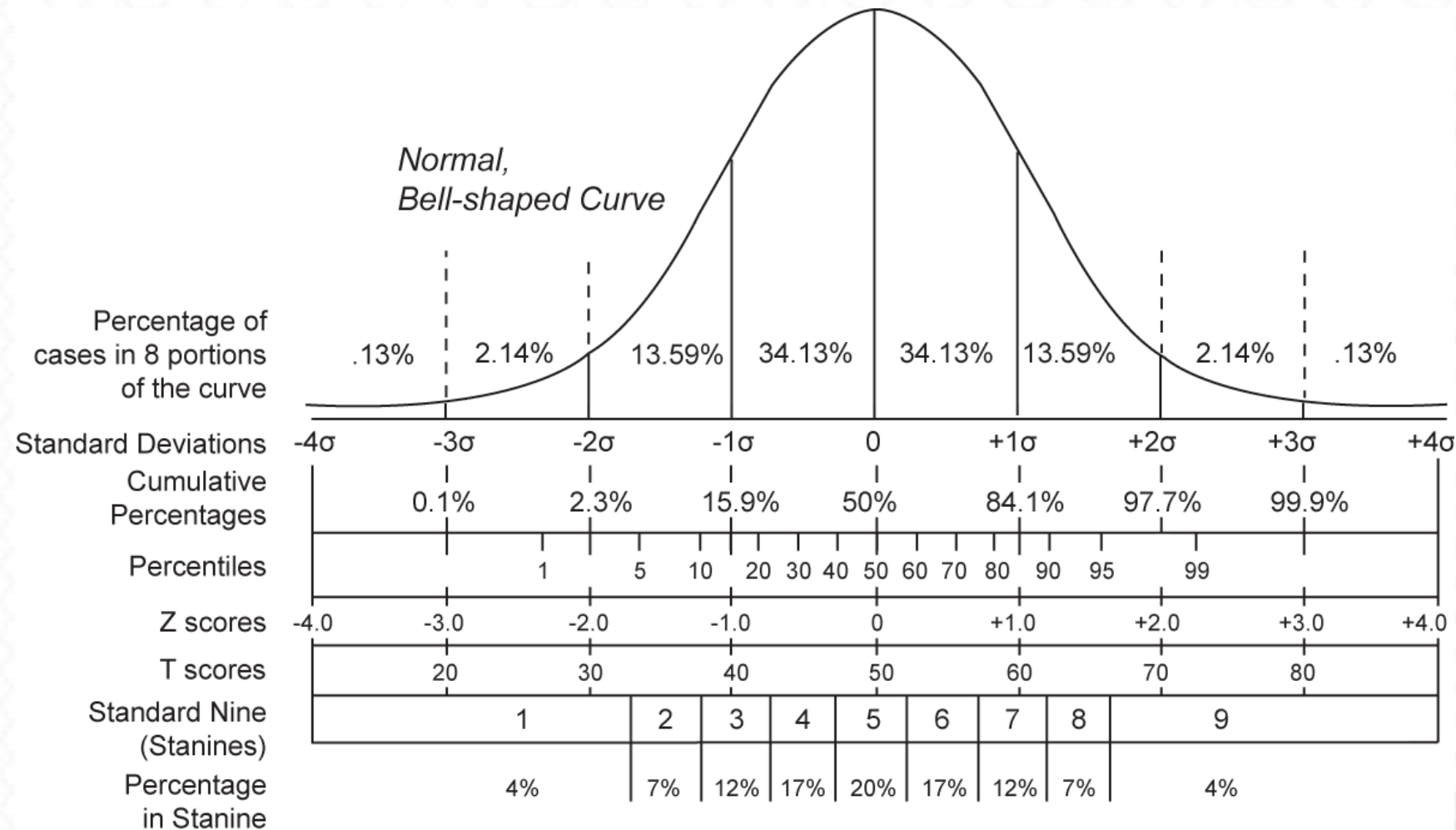
$$z = \frac{x - \mu}{\sigma}$$

其中 $\sigma \neq 0$ 。

其中

- x 是需要被標準化的原始分數
- μ 是母體的平均值
- σ 是母體的標準差

常態分布 與 Z Scores



信賴區間(Confidence Interval)

- 要正確估計母體參數是不可能的，但是可以假設母體參數應該落在一定的區間，稱為信賴區間(confidence interval)
- 而產生信賴區間需要信心水準(confidence level)，或者是誤差(margin of error)
- 點估計加減誤差便是區間估計
 - 信賴區間=點估計 \pm 誤差
 - 誤差=critical value \times sde
 - 而critical value(z值)來自於 $\alpha=1 - (\text{信賴區間}/100)$
 - z值對應 α ， $\alpha/2$ 分屬於z值分佈的兩端

The background features a light blue hexagonal grid pattern. Overlaid on this is a large, faint, circular graphic composed of several concentric rings. These rings are segmented, resembling a stylized spiral or a series of overlapping paths. The colors of the rings transition from a pale blue at the outer edge to a slightly darker, more vibrant blue towards the center. The overall effect is a modern, geometric, and somewhat futuristic aesthetic.

THANK YOU