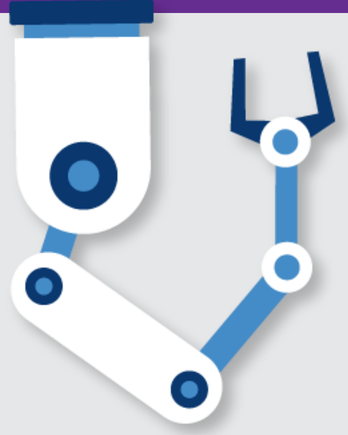
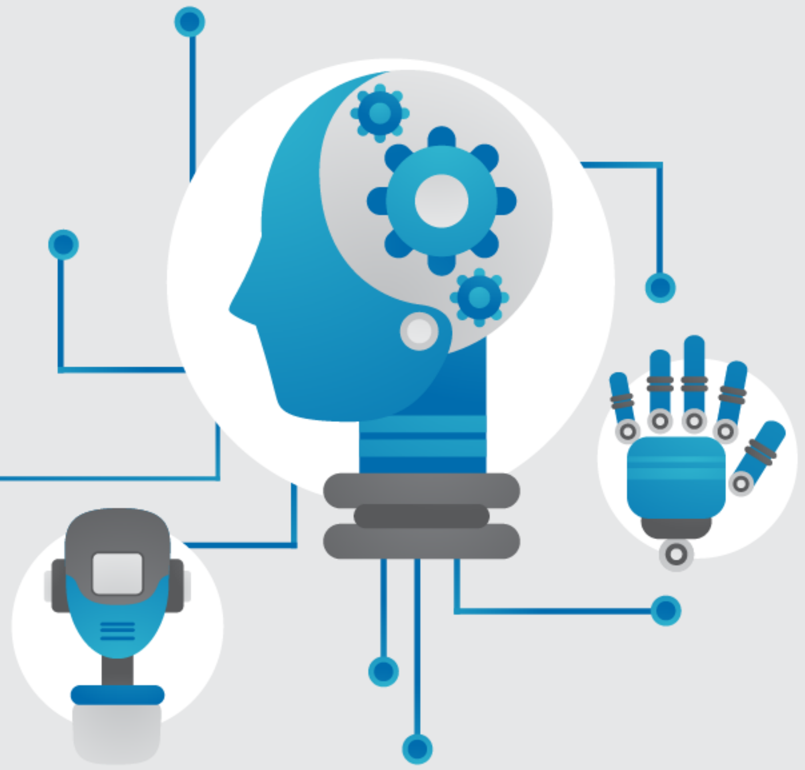


# K-近鄰演算法



# K-近鄰演算法



- **k-近鄰演算法（k-Nearest Neighbor, KNN）**
  最早是由 N.S. Altman 於1992年所提出，是一種用於分類和迴歸的無母數統計方法。

- 在k-NN分類中，一個物件的分類是由其鄰居「多數表決」的結果來決定。k-NN就是由k個最近鄰居的分類結果決定該物件的類別。

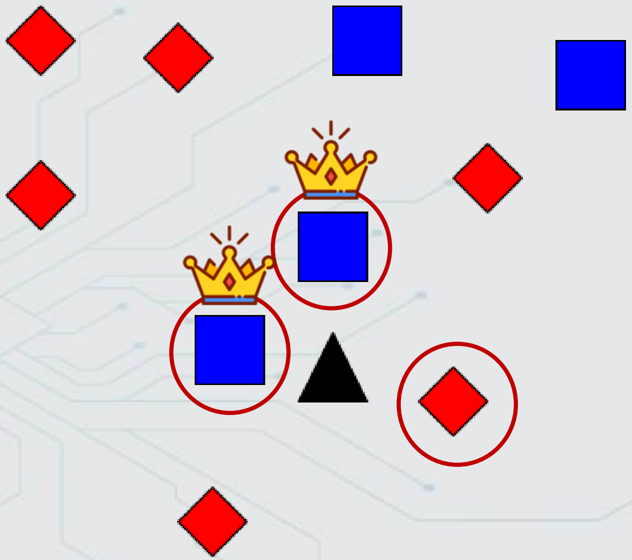
- k的數值影響預測效率

# K-近鄰演算法概念



- 測試樣本為黑色的三角形，如果在分類任務中、K=3，也就是取最近的3個鄰居（紅圈處），並且由這3個鄰居投票表決測試樣本的標籤

➡ 黑色三角形的分類結果為跟**藍色正方形**同一類別



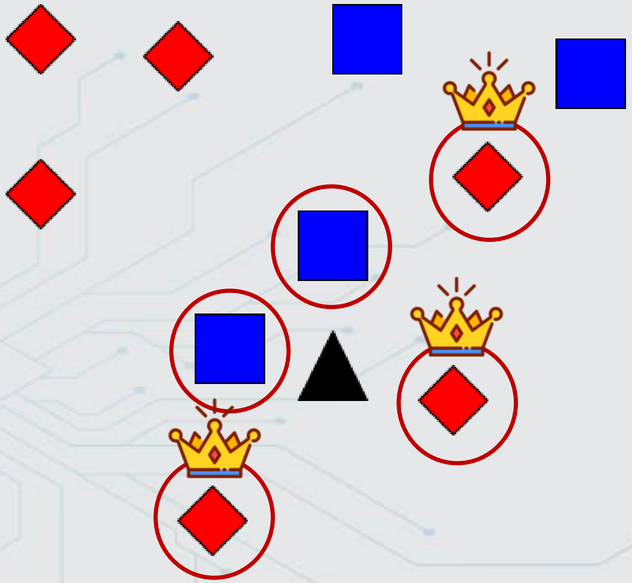
Icon made by Good Ware from www.flaticon.com

# K-近鄰演算法概念



- 如果在分類任務中、K=5，也就是取最近的5個鄰居（紅圈處），並且由這5個鄰居投票表決測試樣本的標籤

➡ 黑色三角形的分類結果為跟**紅色菱形**同一類別



# K-近鄰演算法概念



- 訓練樣本是**多維特徵空間向量**，其中每個訓練樣本帶有一個**類別標籤**。
- 演算法的訓練階段包含訓練樣本的特徵向量和標籤。

# K-近鄰演算法概念



1.決定k值

2.計算每個測試樣本和所有訓練樣本之間的距離

3.選擇k個近鄰集合

4.統計此k近鄰所屬分類

5.根據分類統計最多數的類別為此測試樣本的類別

6.重複步驟2~5直到所有樣本都分類完畢

# 距離函數



- 距離函數決定K-近鄰演算法中如何計算兩點間的距離
- 常見的距離函數

歐幾里得距離（Euclidean Distance）

曼哈頓距離（Manhattan Distance）

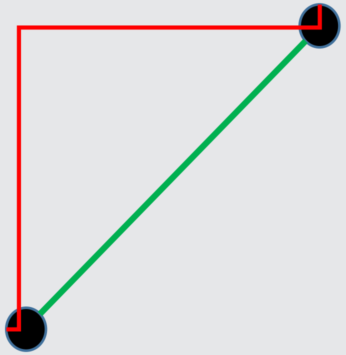
明可夫斯基距離（Minkowski Distance）

# 常見的距離函數



- 假設 $x=(x_1,x_2,\ldots,x_n)$, $y = (y_1,y_2,\ldots y_n)$為兩個n維向量

- 歐幾里得距離（Euclidean Distance）
  - 函數：$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - 圖中**綠色斜線**為歐幾里得距離



- 曼哈頓距離（Manhattan Distance）
  - 函數：$\sum_{i=1}^{n}|x_i - y_i|$
  - 圖中**紅線**為曼哈頓距離

# 常見的距離函數



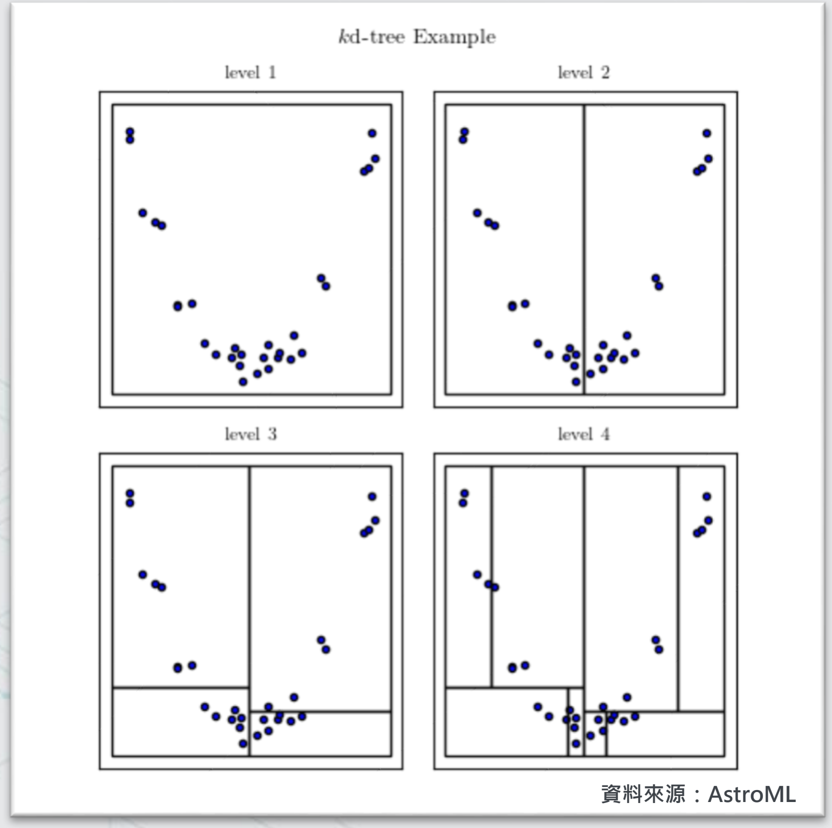
› 明可夫斯基距離（Minkowski Distance）

- 函數：$(\sum_{i=1}^{n}|x_i - y_i|^p)^{\frac{1}{p}}$
- 當p=1時則為曼哈頓距離
- 當p=2即為歐幾里得距離

# 尋找最近k點演算法 - k-d Tree



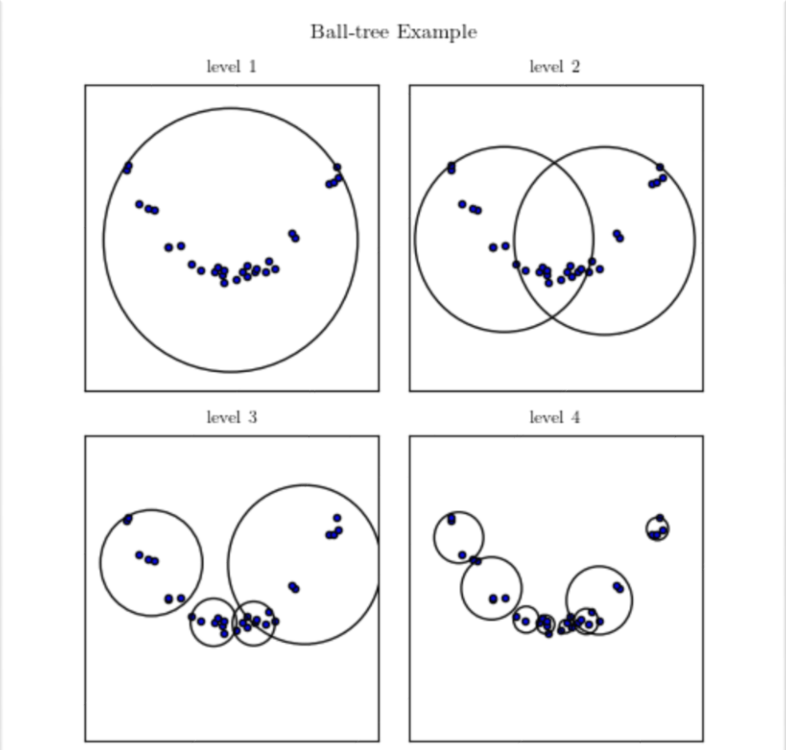
- k-d樹是每個節點都為k維點的二元樹。
  所有非葉子節點可以視作用一個超平面把空間分割成兩個半空間。



# 尋找最近k點演算法 - Ball Tree



- Ball tree將數據遞迴地劃分為由圓心C和半徑r定義的節點，使得節點中的每個點位於由r和C定義的hyper-sphere內。




資料來源：AstroML

# K-近鄰演算法優缺點



優

- 簡單，易於理解，易於實現，無需估計參數，無需訓練

缺

- KNN每一次分類都會重新進行一次全局運算，
  對於樣本數量大的數據集計算量比較大。
- 樣本不平衡時，預測偏差比較大。
- 對於類別數在3個(含)以上，經常會出現投票平局，
  平局要取預設值，這會讓K-NN損失精準度。