AI VIET NAM – Course 2024

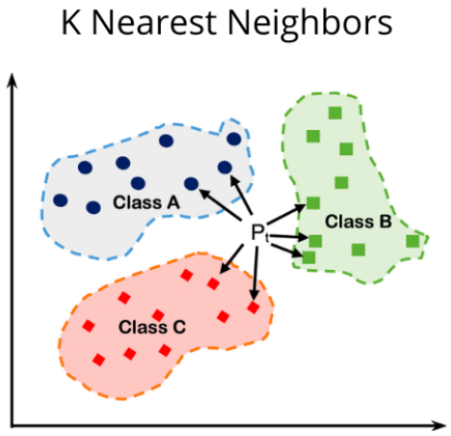
K-Nearest Neighbors (KNN)

Hoàng-Nguyên Vũ

Ngày 10 tháng 2 năm 2024

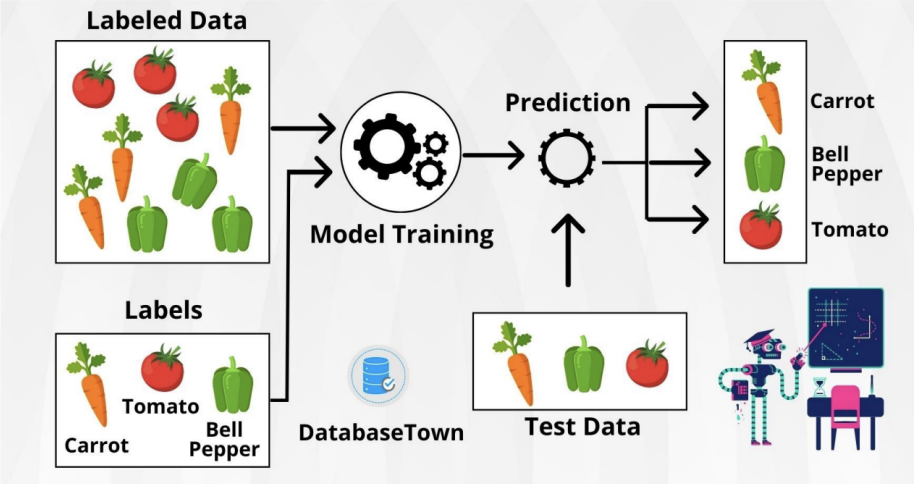
1. **Tóm lược:**

   - **K-nearest neighbor** là một trong những thuật toán **supervised-learning** đơn giản nhất trong Machine Learning. Khi training, thuật toán này gần như không học gì từ dữ liệu training (hay còn được biết tới tên gọi là **lazy learning**), mọi tính toán được thực hiện khi mô hình cần dự đoán kết quả của dữ liệu mới cần dự đoán. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là **Classification và Regression**.



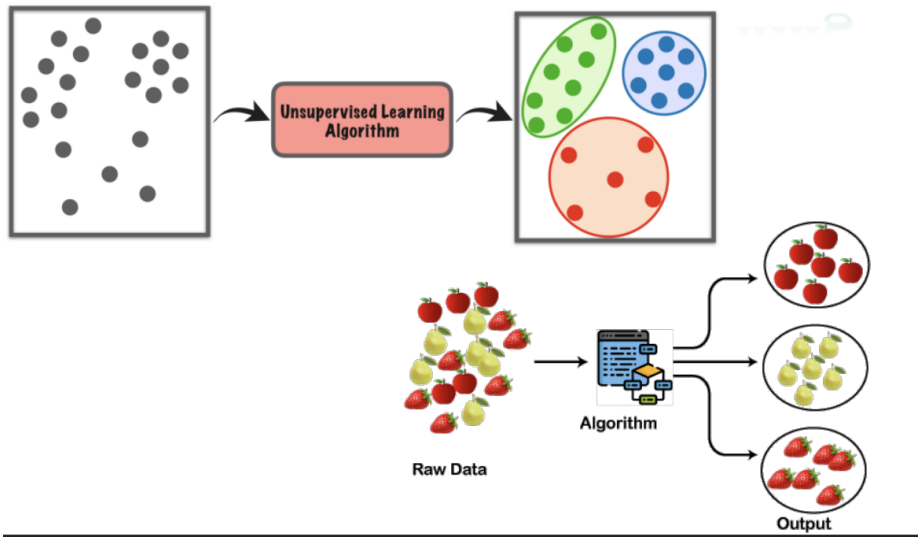
2. **Tổng quan về Machine Learning (ML) :**

   - Nhìn chung, hiện nay có 4 nhóm phổ biến trong Machine Learning bao gồm: Supervise Learning, Unsupervise Learning, Semi-Supervise Learning và Reinforcement Learning
     - **Supervise Learning:**
       * Là mô hình dự đoán kết quả (output) dựa trên tập dữ liệu huấn luyện (training data) được gán nhãn (label). Các bài toán phổ biến của Supervise Learning bao gồm: **Regression** (áp dụng cho dự đoán giá trị continuous) và **Classification** (áp dụng cho bài toán phân loại)



– **Unsupervise Learning:**

∗ Khác với **Supervise Learning** mô hình Unsupervise dự đoán kết quả (output) dựa trên tập dữ liệu huấn luyện (training data) được **không** gán nhãn (label). Các bài toán phổ biến của Unsupervise Learning bao gồm: **Clustering** Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. và **Association** Ví dụ: khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

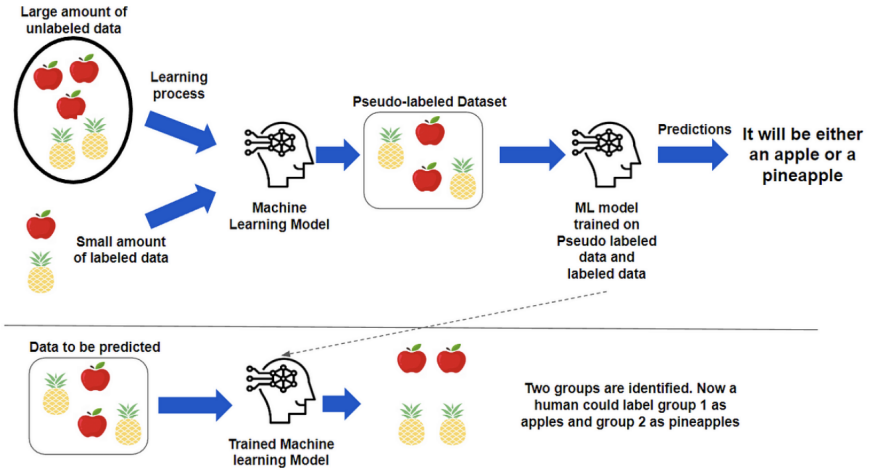


– **Semi-Supervise Learning:**

∗ Semi-supervsied learning là một phương pháp sử dụng **kết hợp** cả dữ liệu **có nhãn** và dữ liệu **không nhãn** trong quá trình huấn luyện mô hình học máy. Ví dụ: Các bài toán khi chúng ta có một lượng lớn dữ liệu X nhưng chỉ một phần nhỏ trong chúng được gán nhãn, phần còn lại đều chưa được gán nhãn, lúc này ta sẽ áp dụng kỹ thuật này ở 2 Step:

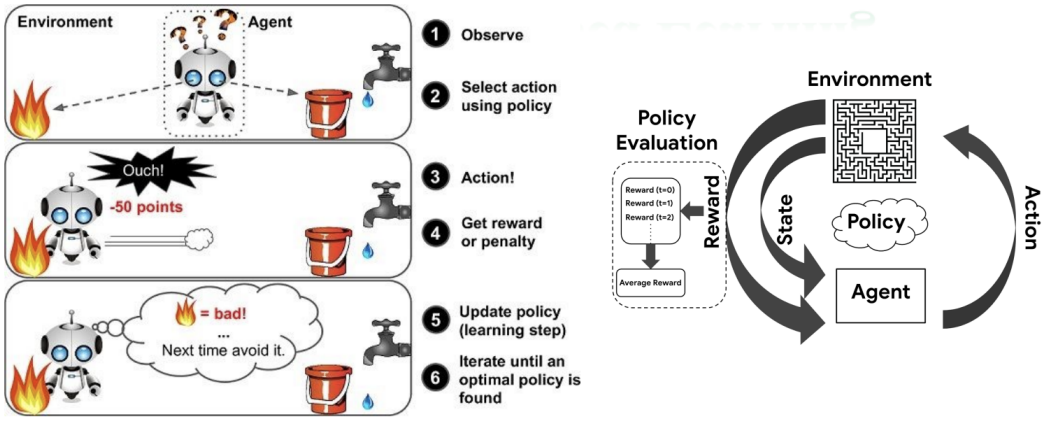
· **Step 1:** Tạo mô hình ML với tập data đã được label với tập data chưa label để tạo ra kết quả output là **Pseudo Label Dataset**

· **Step 2:** Sử dụng Pseudo Label Dataset ở bước trên xây dựng mô hình ML khác và thực hiện dự đoán



– **Reinforcement Learning:**

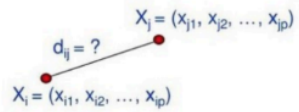
∗ Là mô hình giúp cho một hệ thống tự động xác định hành vi dựa trên hoàn cảnh để đạt được lợi ích cao nhất (maximizing the performance). Hiện tại, Reinforcement learning chủ yếu được áp dụng vào Lý Thuyết Trò Chơi (Game Theory), bao gồm State và Action kèm với Reward cho mỗi action tương ứng để mô hình có thể cập nhật lại một cách tự động.



3. **K-Nearest Neightboor (KNN) :**

- Như đã trình bày ở trên, KNN là là một trong những thuật toán **supervised-learning** đơn giản nhất trong Machine Learning, có thể áp dụng được vào cả hai loại của bài toán Supervised learning là **Classification và Regression**.
- Các bước để thực hiện thuật giải KNN như sau:

- **Bước 1:** Chọn K lân cận để thực hiện bước voting
- **Bước 2:** Tính khoảng cách của data input với các data trong có trong tập data train, có các cách tính khoảng cách như: Minkowski, Euclid, Manhattan, ... tùy mục đích sử dụng mà chúng ta sử dụng cách tính khoảng cách, thông dụng nhất là cách tính Euclid



- **Minkowski distance**

$$d(i,j) = \sqrt[q]{|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \ldots + |x_{ip} - x_{jp}|^q}$$

1st dimension, 2nd dimension, pth dimension

- **Euclidean distance**

q = 2

$$d(i,j) = \sqrt{|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \ldots + |x_{ip} - x_{jp}|^2}$$

- **Manhattan distance**

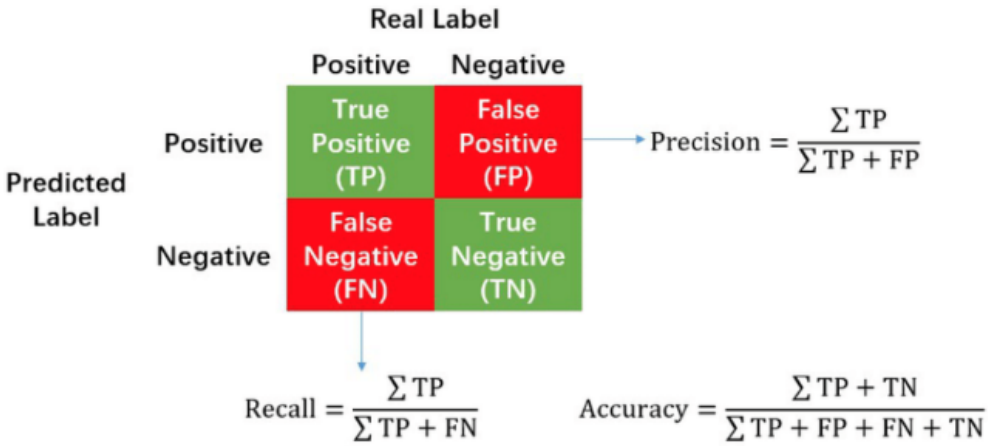
q = 1

$$d(i,j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \ldots + |x_{ip} - x_{jp}|$$

- **Bước 3:** Sau khi tính khoảng cách từ data input tới toàn bộ data trong tập training, chọn ra K lân cận với khoảng cách ngắn nhất, với K được chọn ở bước số 1
- **Bước 4:** Thực hiện Voting, kết quả sẽ theo label có tỉ lệ voting cao nhất

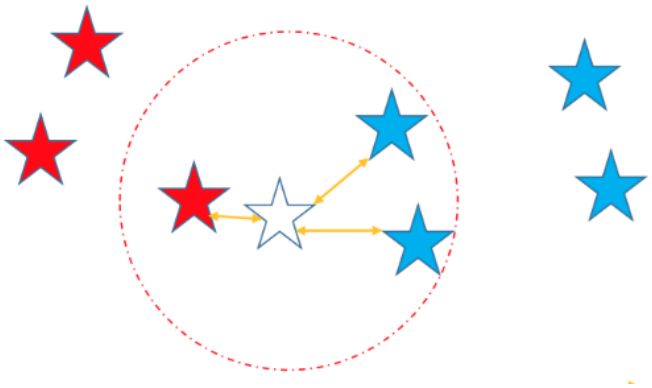
4. **Làm thế nào để lựa chọn K phù hợp ?**

- **Làm thế nào để xác định mô hình hiệu quả ?**
  - Trên thực tế, nếu chúng ta chỉ quan tâm đến độ chính xác của mô hình (accurancy) càng cao thì càng tốt thì quan điểm này chưa chính xác.
  - **Ví dụ:** Bài toán nhận dạng ung thư phổi, giả sử mô hình Machine Learning sau khi chúng ta xây dựng xong đạt được độ hiệu quả là 99% trên tập data kiểm thử, điều này đồng nghĩa với 100 bức ảnh thì mô hình dự đoán đúng 99 tấm ảnh và 1 ảnh dự đoán sai. Trường hợp khác, mình bạn học viên khác xây dựng hàm rất đơn giản, cứ tấm ảnh nào đưa vào bạn kết luận không mắc bệnh. Vậy nếu như khách hàng kiểm thử với data của họ chỉ với 0.1% trong số đó là mắc bệnh vậy thì mô hình bạn học viên không dùng Machine Learning hoạt động hiệu quả hơn với độ chính xác 99.9% so với mô hình Machine Learning đạt 99%. Vậy thì điều này không hợp lý, nên chúng ta cần có thước đo để xác định mô hình có đang hiệu quả hay không, cụ thể là **Confusion Matrix**



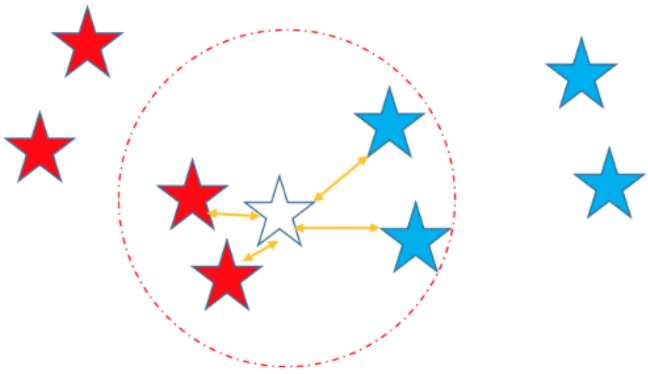
- Precision: Trong tất cả các bệnh nhân mà mô hình dự đoán bị bệnh, tỉ lệ bệnh thật sự là bao nhiêu ?
- Recall: Trong tất cả các bệnh nhân thật sự bị cancer, chương trình dự đoán đúng được bao nhiêu trường hợp ?

- **Giá trị K như thế nào là phù hợp ?**
  - **Nếu K là số lẻ :** Khi K là số lẻ khi thực hiện voting chúng ta luôn thu về kết quả voting luôn tồn tại số voting cao hơn nên kết quả sẽ không khiến chúng ta gặp khó khăn trong việc kết luận.



- Ví dụ trên dễ nhận thấy khi K = 3, kết quả dự đoán sẽ là màu xanh vì số sao xanh là 2 lớn hơn sao đỏ

  - **Nếu K là số chẵn :** Khi K là số chẵn khi thực hiện voting chúng ta sẽ có thể thu về kết quả voting mà số voting cho mỗi phân loại đều bằng nhau. Lúc này ảnh hưởng đến kết quả dự đoán không hiệu quả.

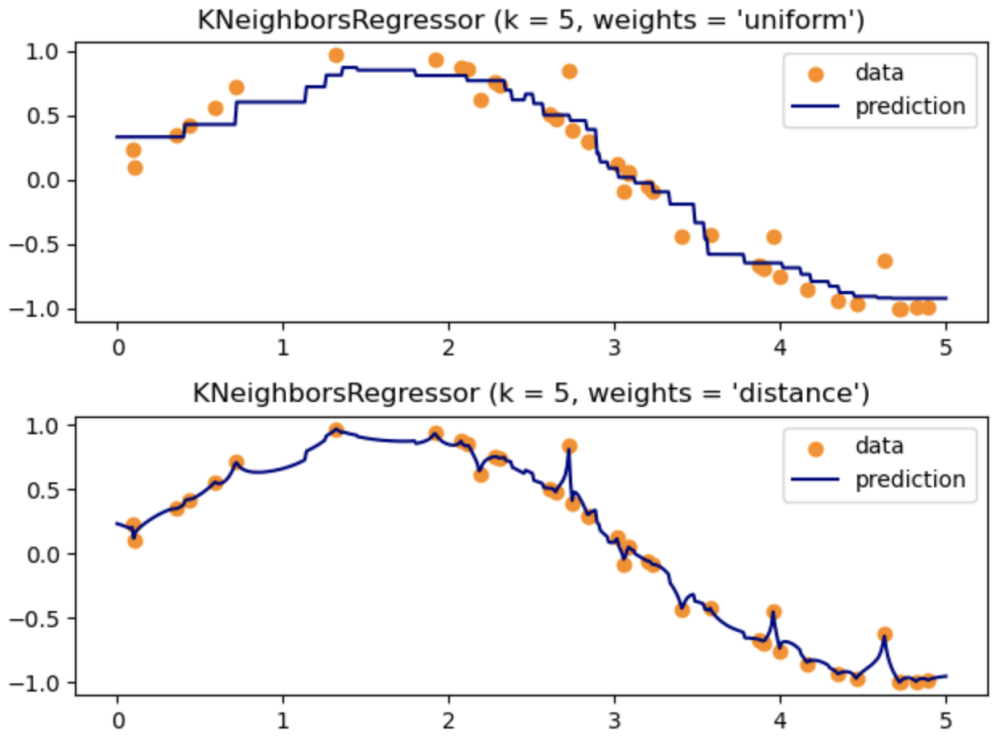


- Ví dụ trên dễ nhận thấy khi K = 4, kết quả dự đoán sẽ là màu xanh hoặc vì số sao xanh và sao đỏ bằng nhau nên mô hình dự đoán bên nào cũng đúng. Dẫn đến kết quả không hiệu quả.
- Để khắc phục, chúng ta sẽ xét thêm trọng số (Weight), có thể hiểu khi trường hợp trên xảy ra, mô hình sẽ xét trọng số của các số phiếu voting, trọng số ở bài này được tính bằng: uniform weight, distance weight, và customize weight. Nếu không định nghĩa mặc định mô hình cho rằng mọi phiếu bầu đề có trọng số bằng nhau ứng với uniform weight. Còn khi chúng ta xét distance weight, mô hình sẽ căn cứ phân loại có distance

ngắn nhất và đưa ra kết luận, cụ thể trong ví dụ ngôi sao đỏ sẽ là kết quả cuối cùng, vì khoảng cách từ input đến sao đỏ ngắn hơn sao xanh

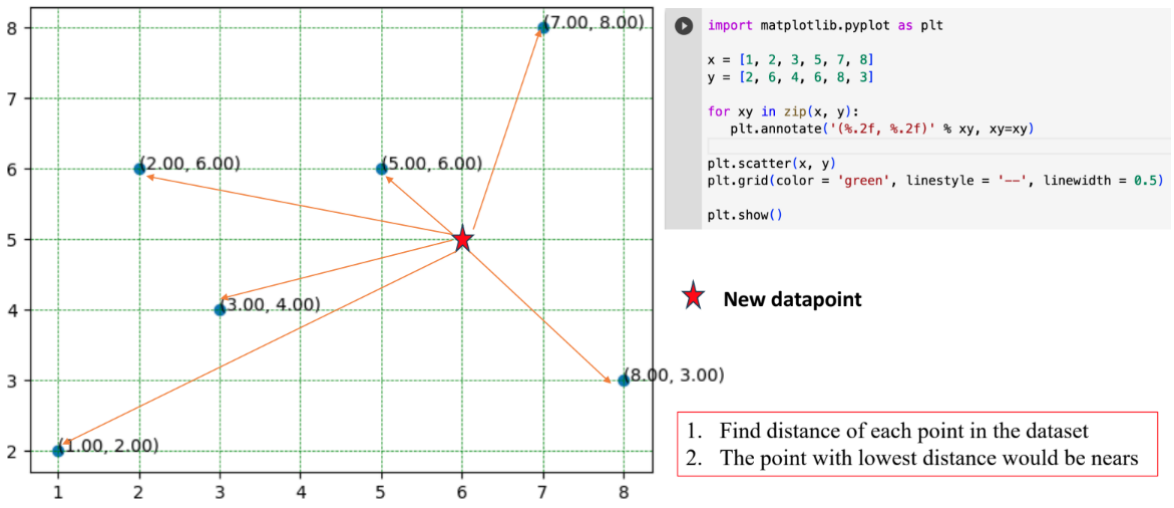
5. **KNN trong bài toán regression?**

- Đối với bài toán regression, chúng ta cũng làm giống với bài toán phân loại nhưng điểm khác biệt là sau khi chọn K lân cận thì chúng ta thực hiện bước tính trung bình để cho ra kết quả dự đoán

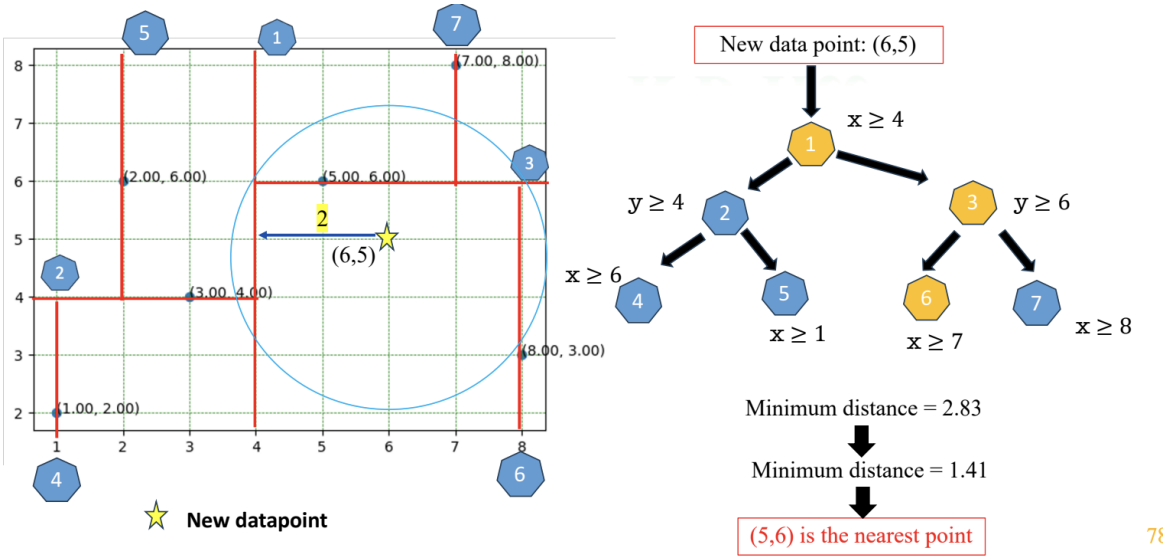


6. **Các giải thuật trong KNN**

- **Brute Force:**
  - Ý tưởng của giải thuật brute force hay thường gọi là quét cạn, đơn giản là chúng ta sẽ tính khoảng cách từ input data đến tất cả các data trong tập training. Dễ thấy, nếu tập training của chúng ta càng lớn, thì việc tính toán sẽ mất rất nhiều thời gian dẫn đến giải thuật này không hiệu quả với lượng data lớn.

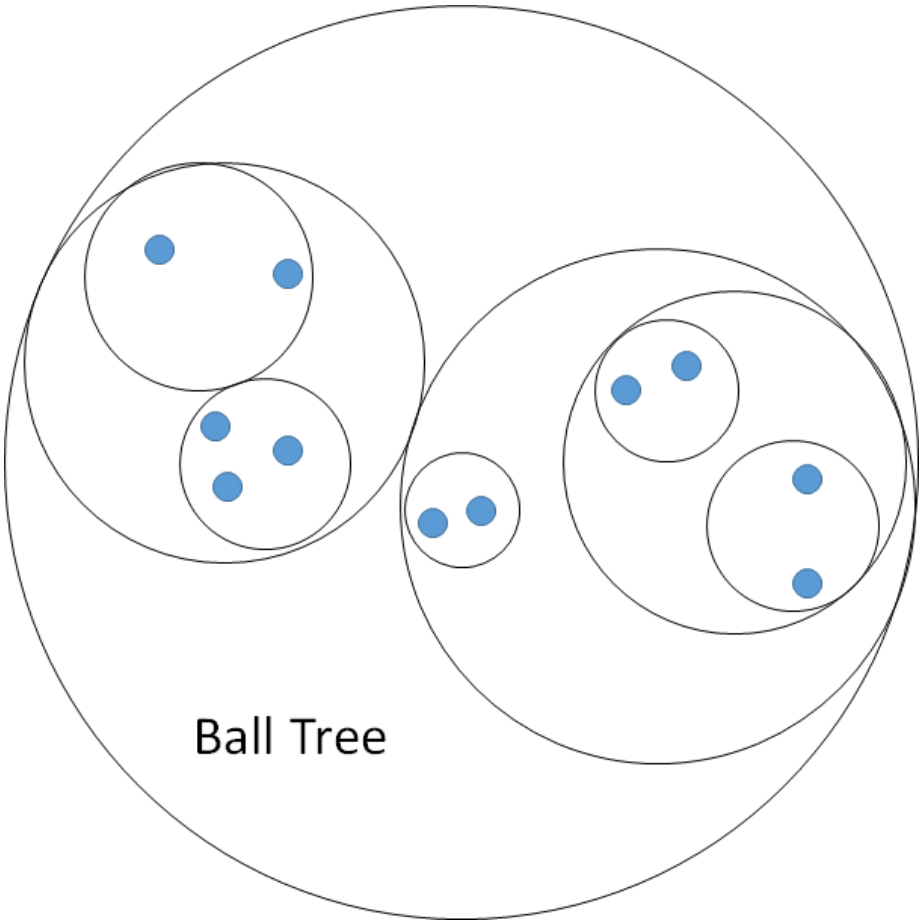


- **K-D Tree:**
  - Để khắc phục hạn chế của giải thuật brute force, K-D Tree dựa trên ý tưởng của giải thuật về cây quyết định (Decision Tree), để cải thiện hiệu năng trong việc tính toán như sau:
    - **Bước 1:** Chọn ngẫu nhiên 1 feature, ví dụ hình trên, chúng ta lấy trục feature Ox
    - **Bước 2:** Tính giá trị trung vị (median) của feature X, ở đây ta thu được điểm A
    - **Bước 3:** Khi chọn được điểm A, chúng ta có thể dễ dàng chia tập data thành {B, C, D} và {E, F}
    - **Bước 4:** Tiếp tục làm tương tự với các tập data sau khi phân nhánh nhỏ hơn, ta sẽ thu về cây quyết định như ví dụ
  - Khi đã có cây quyết định, chúng ta có thể dễ dàng đưa ra kết luận dựa vào các nút gốc và nút lá



- **Ball Tree:**
  - Cũng giống với K-D Tree nhưng đối với Ball Tree, hướng tiếp cận của tác giả giải thuật sẽ hơi khác, cụ thể tác giả thiết kế giải thuật như sau:

* **Bước 1:** Lựa chọn 1 điểm bất kỳ trong training data
* **Bước 2:** Tìm điểm xa nhất đối với điểm vừa xác định
* **Bước 3:** Tìm điểm xa nhất đối với điểm tìm được ở bước 2
* **Bước 4:** Nối 2 đường thẳng giữa 2 điểm ở bước 2 và 3
* **Bước 5:** Chiếu tất cả các điểm data còn lại lên đường ở bước 4
* **Bước 6:** Xác định giá trị trung vị, lúc này chúng ta có thể dễ dàng chia tập data thành các phần nhỏ hơn
* **Bước 7:** Tìm các điểm centroid đơn giản bằng cách tính trung bình giữa các điểm data trong từng nhóm data sau khi phân nhỏ
* **Bước 8:** Lặp lại cho đến khi kết thúc tập dữ liệu train



– Sau khi xây dựng được các bước trên, chúng ta vẫn thu về được 1 cây quyết định dựa theo các điểm centroid, lúc này khi thực hiện dự đoán, các bước tương tự giống với K-D Tree

- Hết -