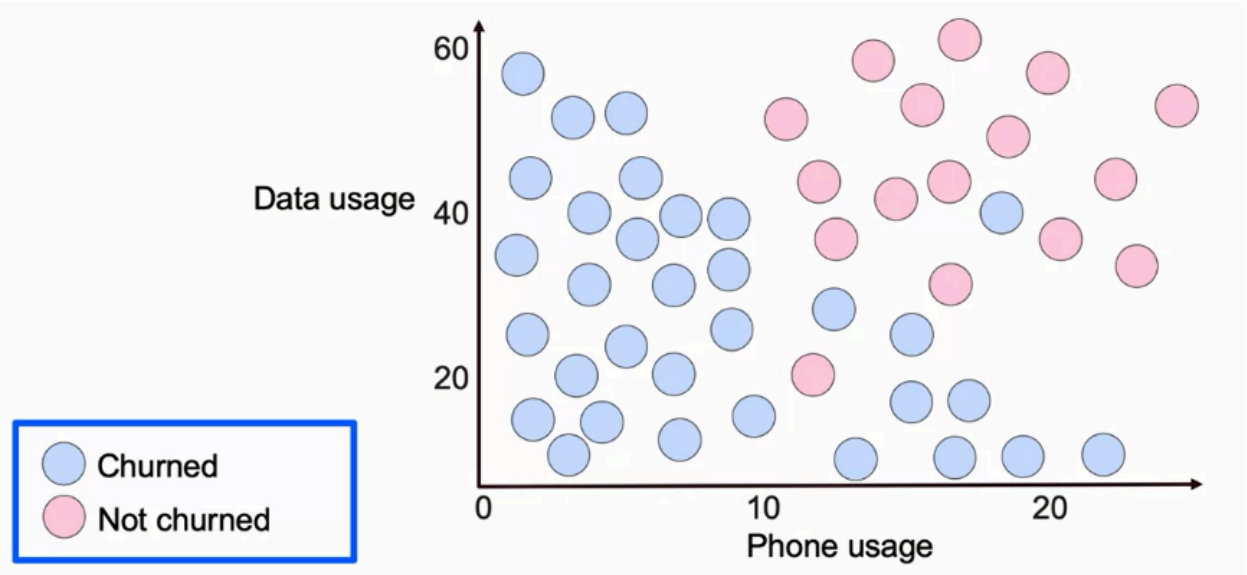
Course3_Module2

K Nearest Neighbors

1) Giới thiệu nhanh về KNN

- **KNN (K Nearest Neighbors)** là thuật toán học có giám sát (supervised), **không tham số** (*non-parametric*): không giả định phân bố xác suất hay dạng hàm nền.
- **Lazy learner / Memory-based**: không huấn luyện mô hình tham số; thay vào đó lưu trữ toàn bộ dữ liệu huấn luyện. Khi dự đoán, mới "tra cứu lân cận".
- Ứng dụng: **Phân lớp** (phổ biến hơn) và **Hồi quy**.

> Trực giác: Mẫu mới sẽ có nhãn (hoặc giá trị) giống đa số các điểm gần nó nhất trong không gian đặc trưng.



---

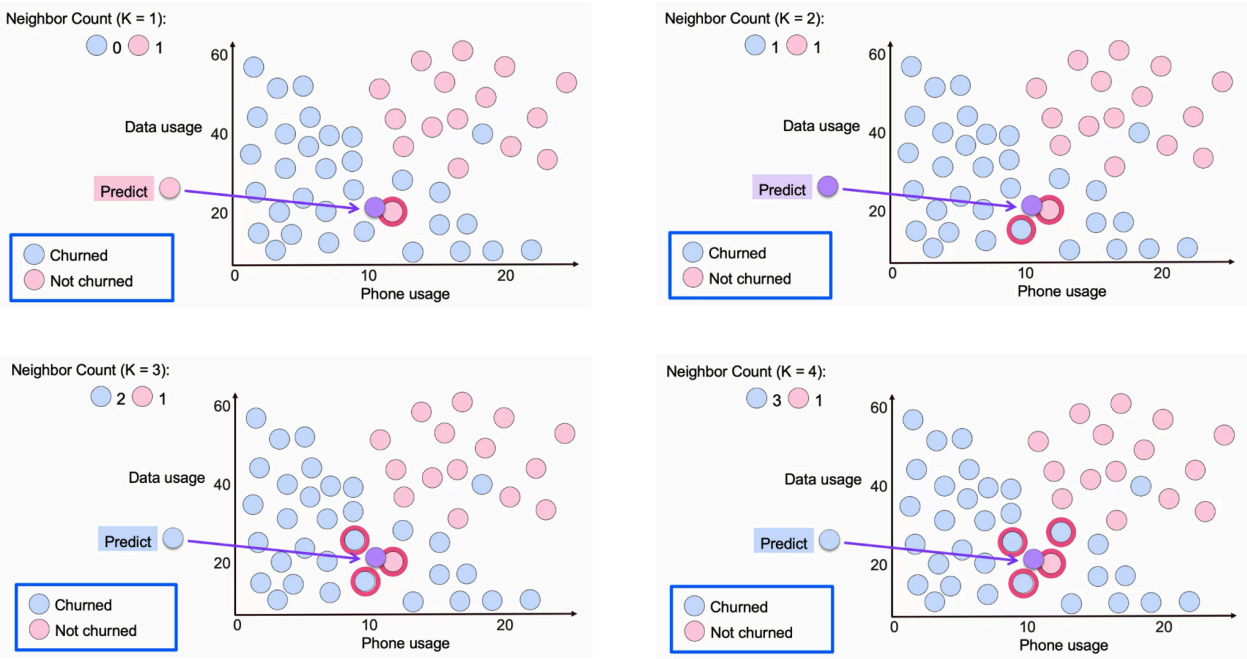
2) Cách KNN hoạt động (Classification)

Bước 1. Chọn **K** (số láng giềng gần nhất).

Bước 2. Với mỗi điểm kiểm tra $\mathbf{x}$

1. Tính khoảng cách giữa $\mathbf{x}$ và **mọi** điểm huấn luyện.
2. Sắp xếp theo khoảng cách tăng dần.
3. Lấy **K** điểm gần nhất.
4. **Bỏ phiếu đa số** (majority vote) → nhãn dự đoán.

> Ràng buộc thực tế: Cần xử lý điểm hoà (tie) và trọng số theo khoảng cách (xem mục 5).



---

3) KNN cho Regression

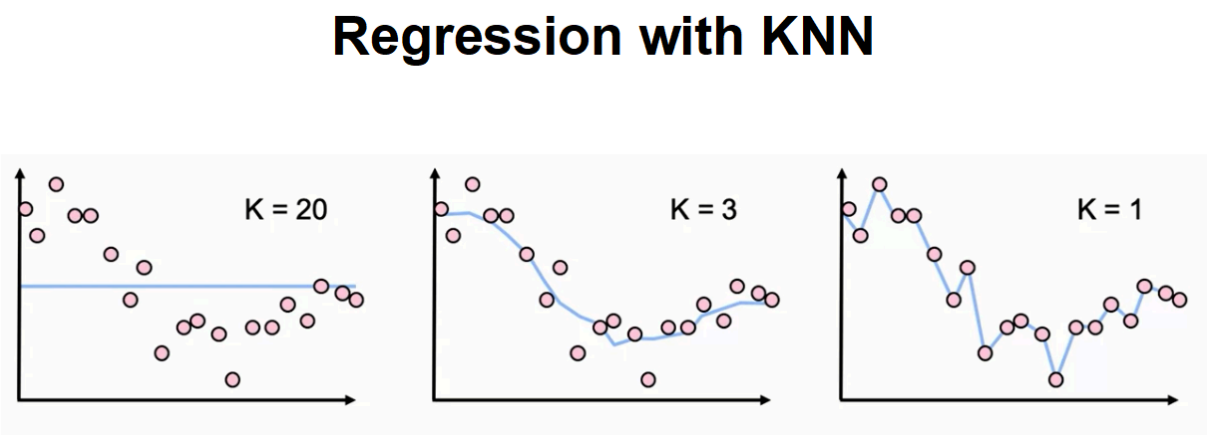
- Dự đoán giá trị **trung bình** của (K) lân cận:

$$\hat{y}(\mathbf{x}) = \frac{1}{K} \sum_{i \in \mathcal{N}_K(\mathbf{x})} y_i$$

- Hoặc **trung bình có trọng số theo khoảng cách**:

$\hat{y}(\mathbf{x}) = \frac{\sum_{i \in \mathcal{N}K(\mathbf{x})} w_i, y_i}{\sum i \in \mathcal{N}_K(\mathbf{x}) w_i}$ với $w_i = \frac{1}{d(\mathbf{x}, \mathbf{x}_i) + \varepsilon} (\varepsilon > 0)$

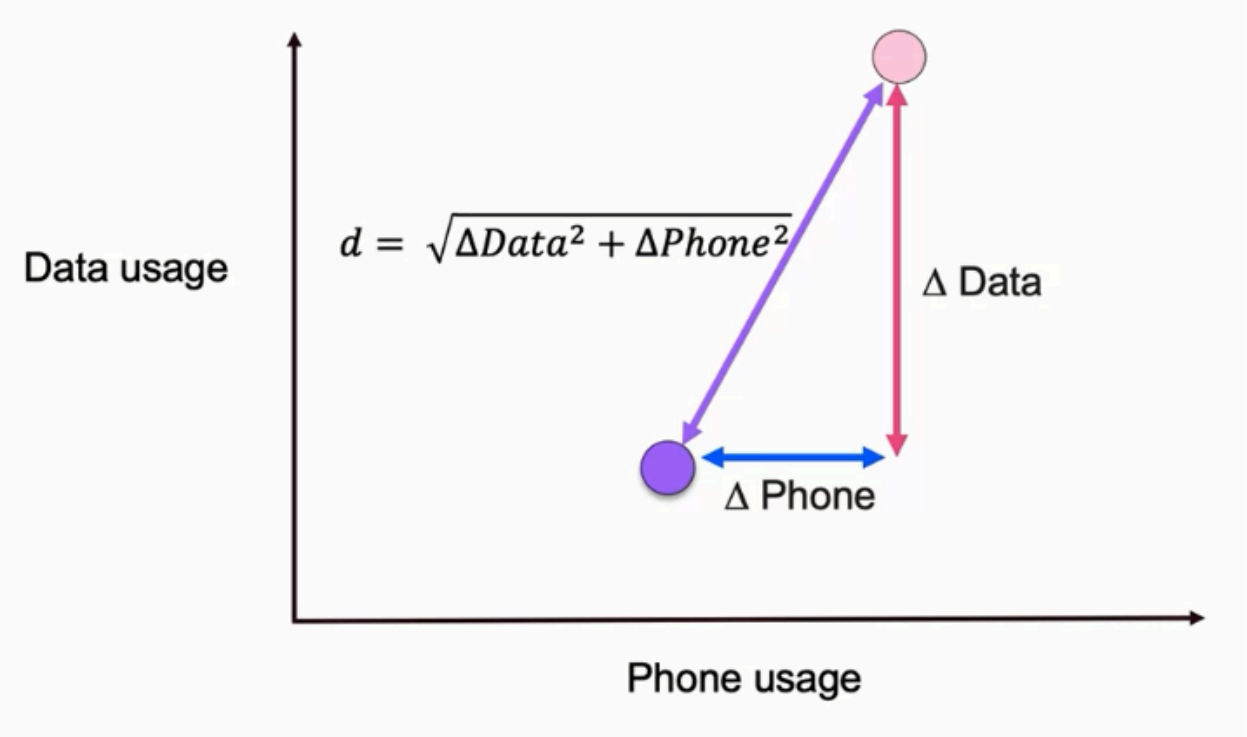
Bước-bước (Regression) giống Classification, khác ở **bước 4**: thay bỏ phiếu bằng công thức trung bình (hoặc trung bình có trọng số).



---

4) Đo khoảng cách — công thức & giải thích từng bước

4.1 Euclidean Distance (L2)



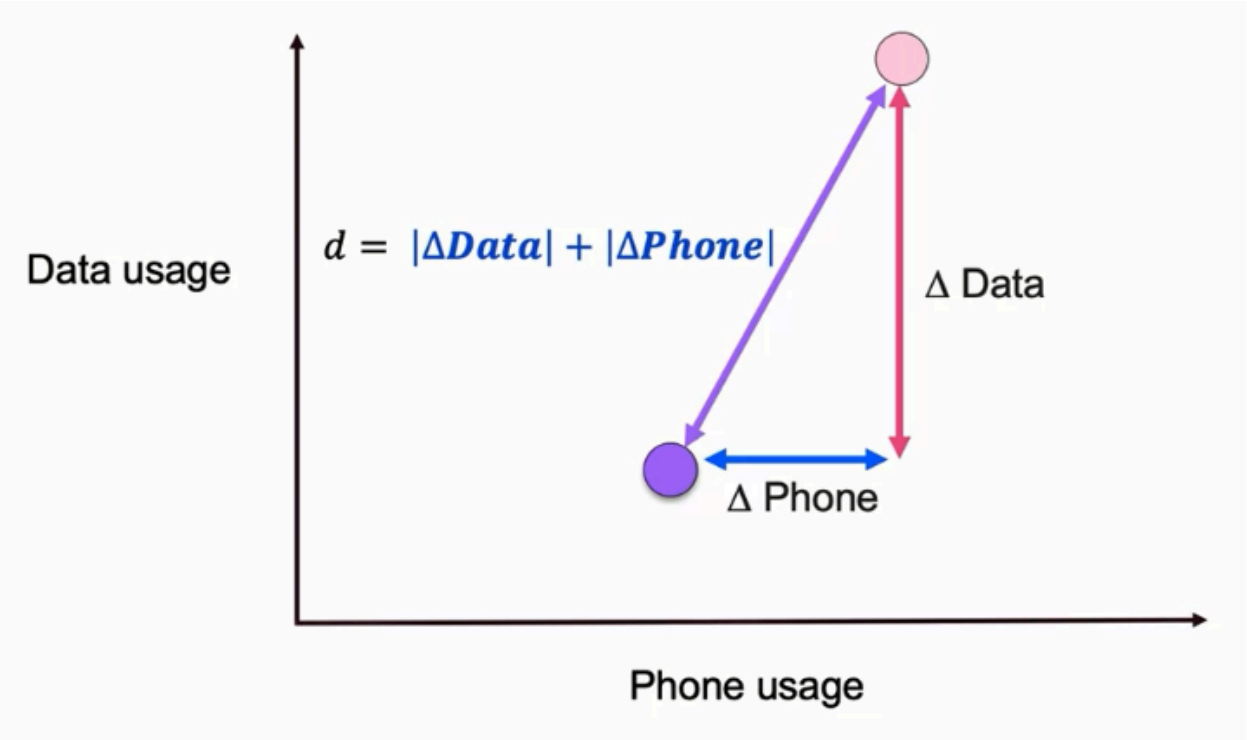
Công thức:

$$d(\mathbf{x}, \mathbf{z}) = \sqrt{\sum_{j=1}^{p} (x_j - z_j)^2}$$

Từng bước:

1. Lấy hiệu từng thuộc tính: $\Delta_j = x_j - z_j$
2. Bình phương: Δ_j^2.
3. Cộng lại: $S = \sum_j \Delta_j^2$.
4. Lấy căn bậc hai: $d = \sqrt{S}$.

4.3 Manhattan Distance (L1)



$$d(\mathbf{x}, \mathbf{z}) = \sum_{j=1}^{p} |x_j - z_j|$$

Từng bước:

1. Tính hiệu từng thuộc tính: $\Delta_j = x_j - z_j$.

2. Lấy trị tuyệt đối: $|\Delta_j|$.
3. Cộng lại trên mọi thuộc tính: $d = \sum_{j=1}^{p} |\Delta_j|$.

> Ghi chú: "đếm khác biệt" là Hamming distance cho dữ liệu rời rạc, không phải Manhattan.

> Lưu ý: Tuỳ bài toán có thể dùng Minkowski, Cosine (cho vector tần suất/văn bản), nhưng phổ thông nhất là Euclidean và Manhattan.

---

5) Bỏ phiếu & Trọng số theo khoảng cách (Classification)

- **Bỏ phiếu đều**: mỗi láng giềng đóng 1 phiếu cho nhãn của nó.
- **Bỏ phiếu có trọng số**: láng giềng gần hơn có **trọng số lớn hơn**:
  $w_i = \frac{1}{d(\mathbf{x},\mathbf{x}_i)+\varepsilon} \quad \text{hoặc} \quad w_i = e^{-\alpha, d(\mathbf{x},\mathbf{x}_i)}$
- **Xử lý hoà (tie-break)**: ưu tiên lớp có tổng trọng số lớn hơn; hoặc chọn lớp xuất hiện gần hơn nhất.

---

6) Quyền lực của **Feature Scaling** (chuẩn hoá đặc trưng)

KNN **rất nhạy** với thang đo: thuộc tính có biên độ lớn sẽ "lấn át" khoảng cách.

Hai cách phổ biến:

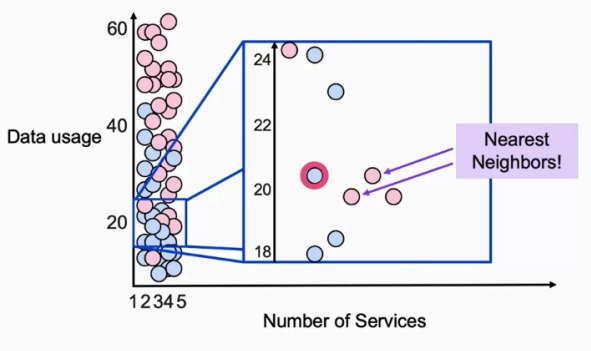
1. **Z-score (Standardization)**

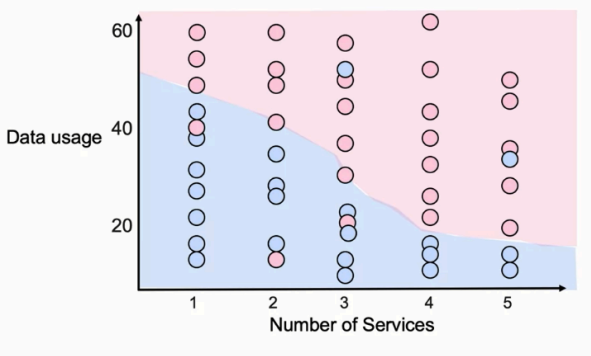
   $x'_j = \frac{x_j - \mu_j}{\sigma_j}$ **Bước-bước:** tính trung bình μ_j, độ lệch chuẩn σ_j trên tập huấn luyện → áp dụng công thức cho train/validation/test bằng **tham số của train**.

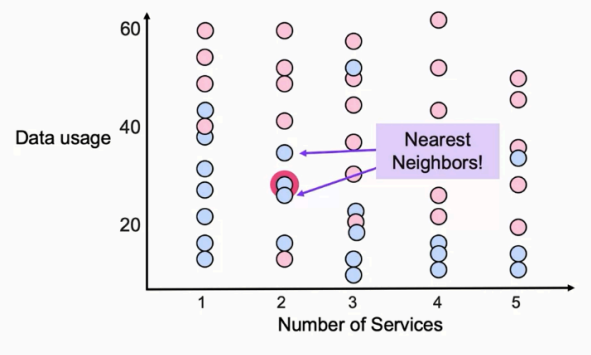
2. **Min–Max (Normalization)**

   $x'_j = \frac{x_j - x_j^{\min}}{x_j^{\max} - x_j^{\min}}$ **Bước-bước:** tìm min/max theo cột trên train → co kéo dữ liệu về [0,1] cho mọi split.

> Thực hành: luôn đặt StandardScaler/MinMaxScaler trước KNN trong Pipeline.







---

7) Đường biên quyết định (Decision Boundary) & chọn K

- **K nhỏ** (ví dụ K=1): biên rất **gấp khúc/nhấp nhô**, nhiễu cao → **variance lớn**, dễ overfit.
- **K lớn**: biên **mượt hơn**, **bias lớn** hơn, có thể underfit.

Chọn K (Elbow Method):

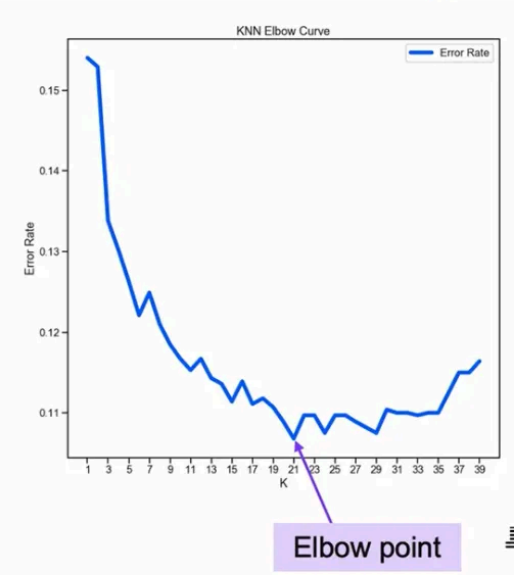
1. Lặp K = 1,2,...,K_max.
2. Đánh giá lỗi (CV/validation) → vẽ **Error vs K**.
3. Chọn "Elbow point" nơi tốc độ giảm lỗi **chững** lại.

> Ngoài ra có thể chọn theo metric ưu tiên (Accuracy/F1/Recall...), trọng số khoảng cách, loại distance, và ràng buộc tốc độ.

K Nearest Neighbors Decision Boundary

Choosing the right value for K

- KNN does not provide a 'correct' K.
- The right value depends on which error metric is most important
- A common approach is to use an 'elbow method approach
- This emphasizes kinks in a curve of the error rate as a function of K
- Beyond this point, the rate of improvement slows or stops



---

8) Độ phức tạp & Mẹo tối ưu

- Dự đoán KNN cần tính khoảng cách tới **N** điểm: chi phí xấp xỉ **O(N·p)** mỗi truy vấn.
- Với dữ liệu lớn: cân nhắc **chỉ mục không gian** (KD-Tree/Ball-Tree trong `sklearn`), **xấp xỉ láng giềng gần (ANN)**, **giảm chiều** (PCA), hoặc **chọn mẫu**.

---

10) Triển khai với scikit-learn (Classification)

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

pipe = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsClassifier())
])
```

```
param_grid = {
    "knn__n_neighbors": [1, 3, 5, 7, 11, 15, 21],
    "knn__weights": ["uniform", "distance"],
    "knn__metric": ["euclidean", "manhattan"]
}

search = GridSearchCV(pipe, param_grid, cv=5, scoring="f1_macro")
search.fit(X_train, y_train)
print(search.best_params_, search.best_score_)
```

Hồi quy

```
from sklearn.neighbors import KNeighborsRegressor
pipe_reg = Pipeline([
    ("scaler", StandardScaler()),
    ("knn", KNeighborsRegressor())
])

param_grid_reg = {
    "knn__n_neighbors": [3, 5, 7, 11, 15],
    "knn__weights": ["uniform", "distance"],
    "knn__metric": ["euclidean", "manhattan"]
}

search_reg = GridSearchCV(pipe_reg, param_grid_reg, cv=5, scoring="neg_ro
ot_mean_squared_error")
search_reg.fit(X_train, y_train)
print(search_reg.best_params_, -search_reg.best_score_)
```

---

11) Ưu/nhược điểm

Ưu điểm

- Dễ hiểu, trực quan; linh hoạt với dữ liệu **phi tuyến**.
- Áp dụng cho cả Classification/Regression.

Nhược điểm

- **Tốn bộ nhớ** (lưu toàn bộ dữ liệu).
- **Dự đoán chậm** với N lớn.
- **Nhạy** với thang đo và thuộc tính **nhiễu/không liên quan**.

---

12) Tóm tắt nhanh

- KNN: **non-parametric**, **lazy**, dựa trên **láng giềng gần nhất**.
- Chìa khoá: **K**, **khoảng cách**, **scaling**, **weighting**.
- Dùng **Elbow** + **CV** để chọn K; luôn **scale** trước KNN.
- Triển khai dễ với `Pipeline` + `GridSearchCV`.