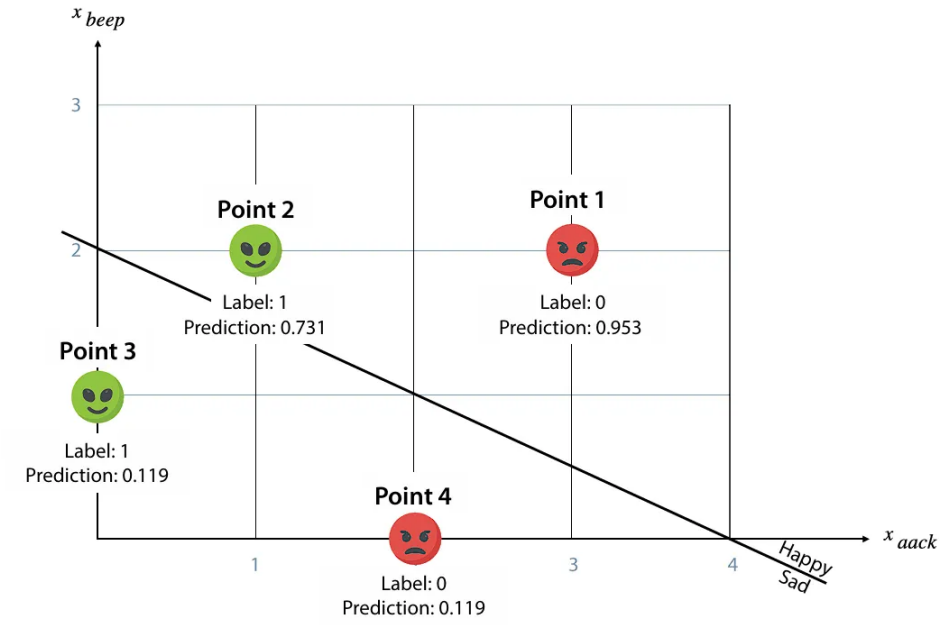
Classification

> Thư viện sử dung và tham khảo: 1. Supervised learning — scikit-learn 1.5.2 documentation

Logicstic Classifier



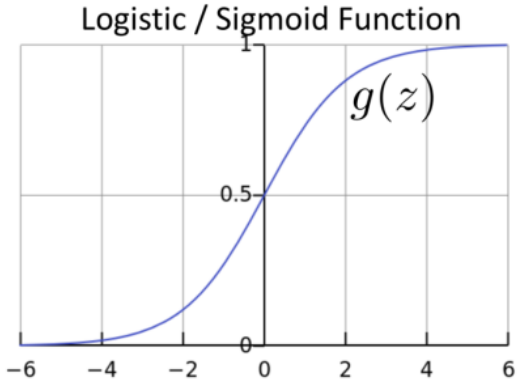
Model này sử dụng một hàm bọc các kết quả của Linear Regression để các kết quả đầu ra sẽ là dạng probability gọi lại hàm sigmoid hay logicstic function:

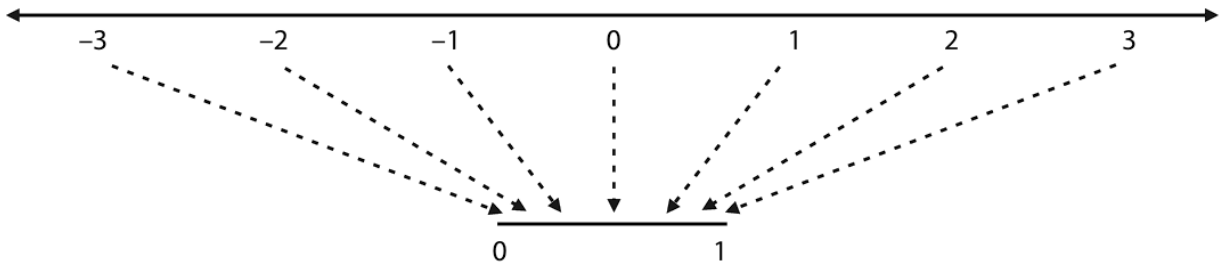
- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}}}$$





- Loss Function (Cost Funcion)

Do hàm h(x) làm hàm non convex (MSE) nên ta sử dụng log cho Cost function

$$\text{cost}\left(h_\theta(x), y\right) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

Điều này giúp tối ưu hóa quá trình tìm kiếm các tham số trong không gian tham số. Hàm Cost function này có tính chất convex nên có thể sử dụng các phương pháp tối ưu như gradient descent để tìm minimum global.

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$

$$\text{where } \frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

- Tránh overfitting

Overfitting xảy ra khi mô hình học quá kỹ các chi tiết và nhiễu trong tập huấn luyện, dẫn đến hiệu suất kém trên tập dữ liệu kiểm tra. Regularization thêm một thuật ngữ phạt vào hàm mất mát, giúp ngăn chặn mô hình trở nên quá phức tạp. Có thể sử dụng:

1. *L1 Regularization (Lasso):*
2. *L2 Regularization (Ridge):*

- Multi class Classification

One vs All (One vs rest) → Soft max regression

$$p_i = \frac{e^{a_i}}{e^{a_1} + e^{a_2} + \cdots + e^{a_n}}.$$

- Tuning Param — Xem thêm tại 1. Supervised learning — scikit-learn 1.5.2 documentation

```
param_grid = {
    # Tham số điều chỉnh độ mạnh của regularization
    'C': [0.01, 0.1, 1, 10, 100, 1000],
    # Loại regularization: L1 (Lasso), L2 (Ridge), ElasticNet
    'penalty': ['l1', 'l2', 'elasticnet', 'none'],  # 'none' kh
    # Thuật toán tối ưu hóa
    'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga']
    'max_iter': [100, 200, 500, 1000],
    'tol': [1e-4, 1e-3, 1e-5],
    # Điều chỉnh trọng số các lớp, hữu ích khi dữ liệu mất cân b
    'class_weight': [None, 'balanced'],  # 'balanced' tự động đ
    # Quyết định có sử dụng intercept (hệ số b0) hay không
    'fit_intercept': [True, False],  # Xem có tính intercept hay
    # Các tham số ElasticNet (nếu penalty='elasticnet')
    'l1_ratio': [0.1, 0.5, 0.7, 1.0],  # Kết hợp giữa L1 và L2,
    # Chọn phương pháp chuẩn hóa dữ liệu
    'intercept_scaling': [1, 5, 10],  # Tham số cho solver 'lib
}
```

Các ứng dụng:

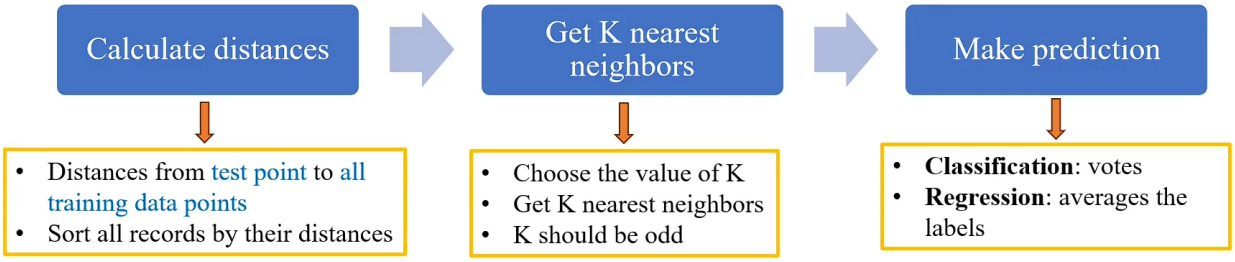
- Dự đoán bệnh
- Customer Behavior Prediction
- Financial Signal Analysis
- Social Science Analysis
- Muticlass

K Nearist Neighbors

Là thuật toán supervise learning đơn giản. Khi training thì thuật toán không học gì từ dữ liệu training mà nhớ một cách máy móc toàn bộ dữ liệu đó. → Lazy Learning

Có thể sử dụng cho classification cũng như regression. KNN còn gọi là thuật toán instance -based.

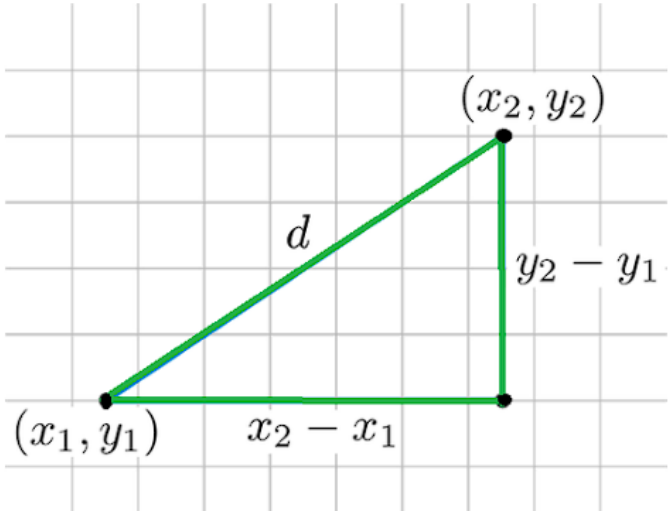
Ý tưởng thuật toán:



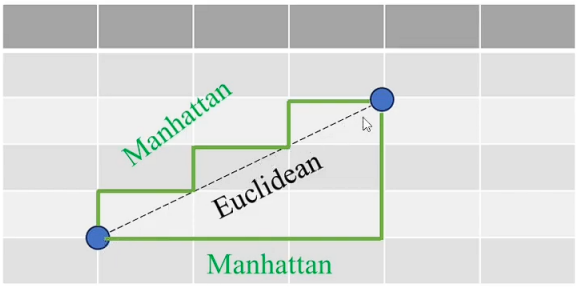
Hàm khoảng cách được được áp dụng :

- Euclidean Distance

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$
$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$



- Mahatan and Minkowski Distance



Manhattan distance

$$d(a,b) = \sum_{i=1}^{N} |a_i - b_i|$$

Minkowski distance

$$d(a,b) = \left(\sum_{i=1}^{N} (a_i - b_i)^p \right)^{\frac{1}{p}}$$

Trong kỹ thuật **major voting**, bản thân mỗi trong bảy điểm gần nhất được coi là có vai trò như nhau và giá trị lá phiếu của mỗi điểm này là như nhau. Tuy nhiên, cách này có thể dẫn đến sự không công bằng, vì những điểm gần hơn cần có trọng số cao hơn. Vì vậy, ta sẽ định trọng số khác nhau cho mỗi trong bảy điểm gần nhất này.

Cách định trọng số phải thỏa mãn điều kiện rằng một điểm càng gần điểm kiểm thử thì trọng số của nó càng cao. Cách đơn giản nhất là lấy **nghịch đảo của khoảng cách** này. Trong trường hợp dữ liệu kiểm thử trùng với một điểm dữ liệu trong tập huấn luyện (tức là khoảng cách bằng 0), ta lấy luôn đầu ra của điểm dữ liệu trong tập huấn luyện đó

Ngoài ra KNN còn được sử dụng cho bài toán Regression

- Kết quả được tính bởi công thức:

$$\frac{w_1 y_1 + w_2 y_2 + \cdots + w_K w_K}{w_1 + w_2 + \cdots + w_K}$$

Default Set up on Sklearn

```
knn = KNeighborsClassifier(
    n_neighbors=5,
    weights='uniform',
```

```
    algorithm='auto',
    leaf_size=30,
    p=2,
    metric='minkowski',
    n_jobs=-1
)
```

Tạo param grid để tuning parameter:

```
param_grid = {
    'n_neighbors': [3, 5, 7, 10],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan', 'minkowski'],
    'p': [1, 2]
}
```

⇒ Sử dụng các kĩ thuật suật Search mà Sklearn cung cấp : GridSearch, RandomSearch,...

Sử dụng Random Search tổ hợp số lượng params quá lớn.

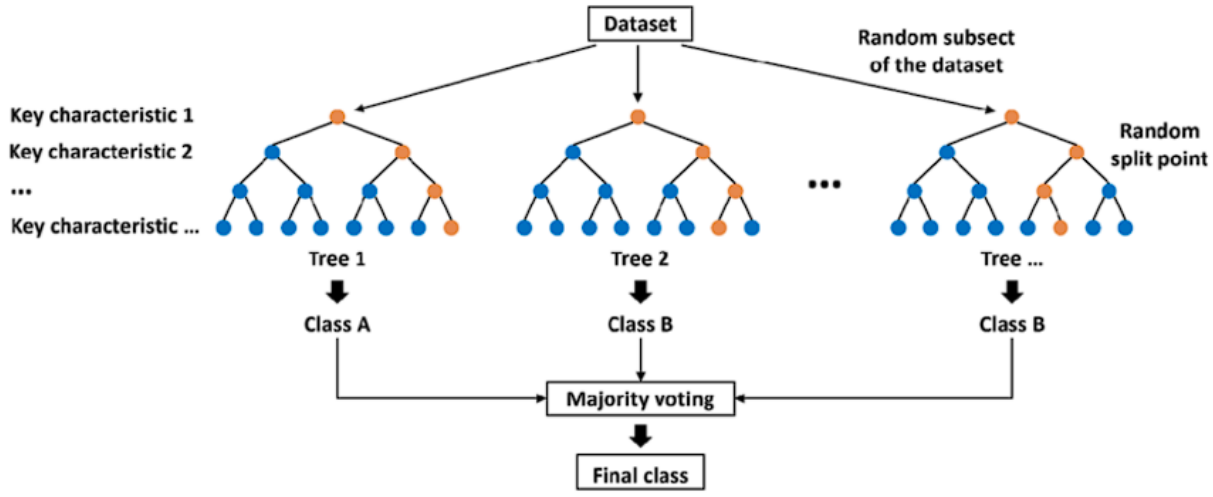
Ứng dụng:

Cũng như các thuật toán Classification KNN có một số ứng dụng trong trong bài toán phân loại như:

- Customer Segmentation
- Phân loại bệnh học
- Fraud Detection
- Handwriting Recognition

Extra Trees Classifier (Extremely Randomized Trees)

ExtraTreesClassifier — scikit-learn 1.5.2 documentation



Extra Trees Classifier là model tập hợp nhiều cây quyết định (Decision Trees) tương tự như Random Forest, Extra Trees tạo ra một tập hợp nhiều cây quyết định (ensemble) và dự đoán dựa trên
majority voting (với bài toán phân loại) hoặc **trung bình** (với bài toán hồi quy).

(Xem lại thêm Decision Tree và Random Forest) → Report của Nam Nguyễn

Extra Trees Classifier thêm một bước ngẫu nhiên hóa mạnh hơn trong quá trình xây dựng cây so với Random Forest:

1. **Chọn ngẫu nhiên giá trị ngưỡng chia tách (split threshold):**
    - Ở mỗi nút của cây, thay vì tìm ngưỡng tốt nhất để chia dữ liệu (như Gini hoặc Entropy), Extra Trees chọn **một ngưỡng ngẫu nhiên**.
    - Việc này tạo ra các cây thô sơ hơn nhưng rất khác nhau, giúp giảm phương sai.
2. **Không sử dụng kỹ thuật bagging (không bootstrap):**
    - Mặc định, Extra Trees sử dụng toàn bộ dữ liệu huấn luyện thay vì lấy mẫu bootstrap như Random Forest.
    - Điều này giúp cây tận dụng tối đa dữ liệu để xây dựng.

Tiêu chí	Random Forest	Extra Trees Classifier
1. Cách xây dựng cây (Tree Construction)	- Chọn ngưỡng chia tách (split threshold) **tối ưu nhất** dựa trên độ bất thuần (Gini, Entropy).	- Chọn ngưỡng chia tách **ngẫu nhiên**, không cần tìm ngưỡng tốt nhất.
2. Ngẫu nhiên hóa ngưỡng chia tách	Ít ngẫu nhiên hơn, tối ưu hóa để chia nút hiệu quả nhất.	Tăng ngẫu nhiên hóa bằng cách chọn ngưỡng chia bất kỳ trong khoảng giá trị đặc trưng.
3. Bootstrap (Lấy mẫu dữ liệu)	Mặc định sử dụng **bootstrap sampling** để lấy mẫu dữ liệu trước khi xây dựng cây.	Mặc định không sử dụng bootstrap; dùng toàn bộ dữ liệu để xây dựng mỗi cây.
4. Tính phức tạp của cây	Cây phức tạp hơn do ngưỡng chia tách được chọn tối ưu.	Cây đơn giản hơn do ngưỡng chia tách được chọn ngẫu nhiên.
5. Độ đa dạng giữa các cây	Ít đa dạng hơn do ngưỡng chia được tối ưu hóa.	Đa dạng hơn do ngưỡng chia được chọn ngẫu nhiên.
6. Độ chính xác trên dữ liệu sạch	Thường có độ chính xác cao hơn nếu dữ liệu ít nhiễu.	Có thể hơi kém hơn Random Forest trên dữ liệu sạch.
7. Khả năng kháng nhiễu	Kháng nhiễu tốt nhưng không bằng Extra Trees nếu dữ liệu nhiều nhiễu.	Kháng nhiễu tốt hơn nhờ tính ngẫu nhiên cao trong cách xây dựng cây.
8. Tốc độ xây dựng cây	Chậm hơn, do phải tính toán ngưỡng chia tách tối ưu.	Nhanh hơn, vì không cần tìm ngưỡng chia tách tối ưu.
9. Tài nguyên tính toán	Yêu cầu tài nguyên cao hơn (do tính ngưỡng tối ưu).	Ít tài nguyên hơn (do ngẫu nhiên hóa ngưỡng chia tách).
10. Độ chính xác tổng quát (Generalization)	Hiệu suất tổng quát tốt trên dữ liệu ít nhiễu hoặc nhiều mẫu.	Hiệu suất tốt hơn trên dữ liệu nhiều nhiễu hoặc đặc trưng không quan trọng.
11. Tùy chọn bootstrap	Có thể bật hoặc tắt bootstrap bằng tham số `bootstrap=True/False`.	Mặc định không dùng bootstrap, nhưng có thể bật nếu cần.

Tạo param grid

```
param_grid = {
    'n_estimators': [100, 200, 300],
```

```
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None],
    'bootstrap': [True, False],  // khong can thiet cung dc
    'class_weight': [None, 'balanced'],
    'n_jobs': [-1]
}
```

Các ứng dụng:

- Phân loại văn bản
- Anomaly Detection
- **Cải thiện các mô hình học máy khác**
- ....