ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
***

BÁO CÁO

MACHINE LEARNING MODELS SUPERVISED

Môn học	:	Trí tuệ nhân tạo
Mã học phần	:	INT3401_20
Giảng viên hướng dẫn	:	TS. Trần Hồng Việt
Sinh viên thực hiện	:	Nguyễn Ngọc Thái - 21021367
		Nguyễn Đức Trí - 21020596
		Vương Ngọc Thiện - 21021372
		Nguyễn Gia Thịnh - 21021373
		Nguyễn Phú Trọng - 21021377

Hà Nội, Năm 2024

Mục lục

I. Mở đầu:

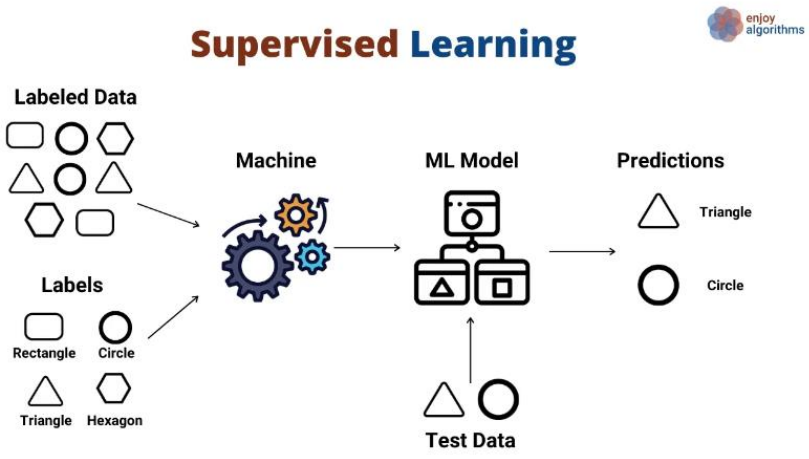
Trong thời đại ngày nay, trí tuệ nhân tạo (AI) đang trở thành một phần quan trọng trong cuộc sống của chúng ta. Một lĩnh vực quan trọng trong AI là học máy (Machine Learning), nơi mà máy tính học từ dữ liệu để đưa ra quyết định hoặc dự đoán. Bài tập lớn này của chúng em sẽ tập trung vào các mô hình học máy giám sát không tham số, cụ thể là K-Nearest Neighbors (KNN) và Decision Tress.

Báo cáo này sẽ giải thích cách hai mô hình này hoạt động, những ưu điểm và nhược điểm của chúng, cũng như các ứng dụng thực tế. Qua đó, chúng em mong muốn giúp mọi người hiểu rõ hơn về cách mà học máy giám sát không tham số có thể được áp dụng trong cuộc sống hàng ngày.

II. Giới thiệu về Machine Learning Models Supervised(Non-parametric)

I. Khái niệm

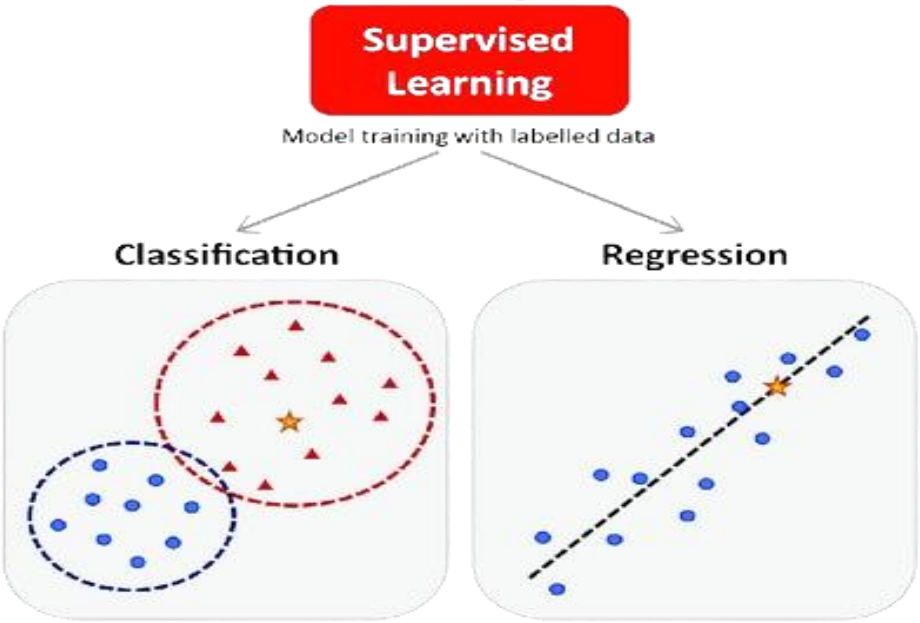
Supervised Learning (Học có giám sát) là một loại học máy sử dụng các tập dữ liệu đã biết trước các đầu ra để huấn luyện các thuật toán dự đoán kết quả và nhận biết các mô hình. Trong quá trình này, mô hình sẽ học cách ánh xạ từ dữ liệu đầu vào đến dữ liệu đầu ra dựa trên các cặp (đầu vào, đầu ra) đã biết từ trước.



II. Phân loại

Supervised Learning được phân thành hai loại thuật toán:

- **Classfication:** Thuật toán phân loại được sử dụng để nhóm dữ liệu bằng cách dự đoán biến đầu ra theo hạng mục dựa trên dữ liệu đầu vào, biến đầu ra của bài toán có các giá trị rời rạc (ví dụ: đồ vật có màu "đỏ" hay "xanh")
- **Regression:** Thuật toán hồi quy được sử dụng để dự đoán một giá trị thực hoặc liên tục, nơi mà thuật toán phát hiện ra mối quan hệ giữa hai hoặc nhiều biến, biến đầu ra của bài toán có các giá trị liên tục (ví dụ: đơn vị USD)



Supervised Learning còn được chia nhỏ thành:

- K-nearest neighbors
- Decision trees
- Naive Bayes Classifier

III. K-Nearest Neighbors:

A. Khái niệm

KNN là một trong những thuật toán phân loại và hồi quy đơn giản và phổ biến nhất được sử dụng trong học máy hiện nay. Nó hoạt động dựa trên giả định rằng các điểm tương tự nhau có thể được tìm thấy gần nhau.

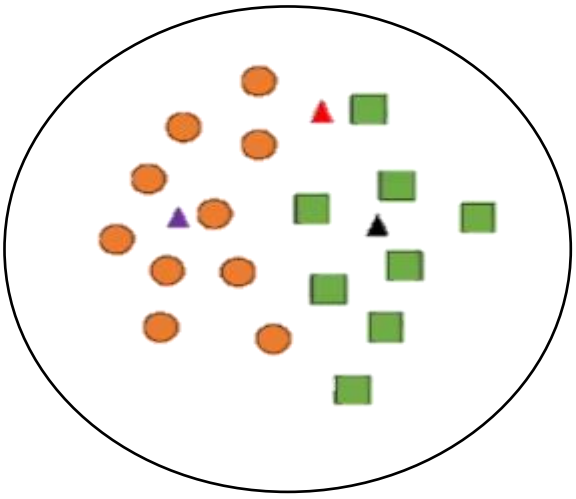
KNN cũng là một phần của các mô hình "lazy learning", có nghĩa là nó chỉ lưu trữ tập dữ liệu huấn luyện thay vì trải qua một giai đoạn huấn luyện. Tất cả các tính toán đều xảy ra khi một phân loại hoặc dự đoán đang được thực hiện.

B. Ý tưởng của KNN

Nếu 1 điểm dữ liệu bất kì (A) nằm trong 1 tập dự liệu (B) hoặc A nằm gần các điểm của B hơn các điểm thuộc tập dữ liệu khác (C) thì ta có thể nói rằng là A thuộc B và A có đầy đủ đặc điểm của B.

Tuy nhiên trong 1 vài trường hợp, yếu tố k sẽ quyết định A thuộc B hay không.

Ví dụ: Cho 1 tập dữ liệu như hình bên dưới bao gồm 2 tập dữ liệu: hình tròn màu cam và hình vuông màu xanh. Trong tập dữ liệu xuất hiện 3 hình tam giác khác màu (tím, đỏ, đen).



C. Phân tích ví dụ

Từ ý tưởng ở trên, ta thấy tam giác màu đen thuộc về tập dữ liệu hình vuông màu xanh. Tam giác màu tìm thuộc về tệp dữ liệu hình tròn màu cam.

Để xác định tam giác màu đỏ thuộc loại dữ liệu nào, ta cần xét đến giá trị của k.

Nếu k = 1 hoặc k = 2 thì tam giác màu đỏ có xu hướng thuộc về tập hình vuông màu xanh, do cả 2 trường hợp thì tam giác đỏ luôn gần với vuông màu xanh hơn là tròn cam.

Nếu k = 3 thì tam giác màu đỏ có xu hướng thuộc về tập tròn cam do có 2 điểm tròn màu cam gần tam giác đỏ hơn là 1 hình vuông xanh.

D. Các bước của học máy trong KNN:

Bước 1: Tính khoảng cách từ điểm dữ liệu mới tới các điểm thuộc tập dữ liệu. Có 3 cách tính thường được sử dụng

Khoảng cách Euclidean: Tính toán khoảng cách giữa hai điểm trong không gian nhiều chiều

$$d(a,b) = \sqrt{\sum_{i=1}^{N}(a_i - b_i)^2}$$

Khoảng cách Manhattan: Tính dựa trên tổng khoảng cách di chuyển của dữ liệu

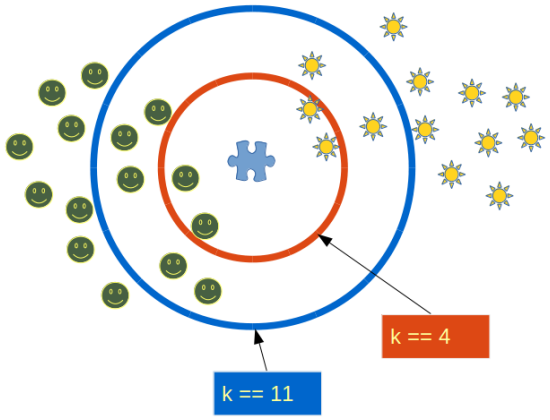
$$d(a,b) = \sum_{i=1}^{N}|a_i - b_i|$$

Khoảng cách Minkowski: Dạng tổng quát của khoảng cách Euclidean và Manhattan

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

Bước 2: Tìm giá trị K phù hợp. Điều này thường được thực hiện bằng cách tính toán khoảng cách từ điểm dữ liệu mới đến tất cả các điểm trong tập dữ liệu huấn luyện, sau đó chọn K điểm có khoảng cách ngắn nhất.

Lưu ý: K nên là số lẻ. Vì khi K là số chẵn, có thể xảy ra tình huống số lượng hàng xóm gần nhất bằng nhau. Khi K là số liệu, tình huống này ít xảy ra hơn vì sẽ có ít nhất 1 số lượng hàng xóm gần nhất nhiều hơn (xem ví dụ ở hình bên dưới)



Bước 3: Sử dụng thông tin từ các láng giềng để đưa ra dự đoán cho điểm dữ liệu mới.

Trong trường hợp phân loại, việc xác định lớp cho một điểm dữ liệu mới được thực hiện thông qua quy tắc "bầu chọn số đông". Cụ thể, lớp mà có số lượng xuất hiện nhiều nhất trong số K láng giềng gần nhất sẽ trở thành lớp dự đoán cho điểm dữ liệu mục tiêu.

Trong trường hợp hồi quy, kết quả sẽ là trung bình (hoặc trọng số) của các giá trị đầu ra của K láng giềng gần nhất. Trọng số thường được chọn dựa trên khoảng cách, với các láng giềng gần hơn có trọng số lớn hơn.

E. Chuẩn hóa dữ liệu

Việc chuẩn hóa dự liệu là một bước quan trọng trong quá trình tiền xử lý dữ liệu và nó có thể ảnh hưởng đến hiệu suất và độ chính xác của nhiều thuật toán học máy, StandardScaler là một công cụ phổ biến để thực hiện việc này, công

cụ này biến đổi dữ liệu sao cho mỗi đặc trưng có giá trị trung bình bằng 0 và độ lệch chuẩn bằng 1. StandardScaler được viết dưới dạng:

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

Trong đó: x là giá trị ban đầu của đặc trưng.

μ là giá trị trung bình của đặc trưng đó trong tập dữ liệu huấn luyện.

σ là độ lệch chuẩn của đặc trưng đó trong tập dữ liệu huấn luyện.

z là giá trị đã được chuẩn hóa

❖ **Chuẩn hóa dữ liệu:**

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

#Chia dữ liệu thành đặc trưng và nhãn
X = mau_du_lieu.iloc[:,0:4] # Đặc trưng
Y = mau_du_lieu.iloc[:,4]   # Nhãn

#Chia dữ liệu thành tập huấn luyện và tập kiểm tra
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=56)

# Khởi tạo và huấn luyện StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)

# Chuẩn hóa dữ liệu huấn luyện và kiểm tra:
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Huấn luyện mô hình KNN
k = 20
knn = KNeighborsClassifier(n_neighbors=k)

# Dự đoán trên tập kiểm tra:
knn.fit(X_train_scaled, Y_train)
Y_pred = knn.predict(X_test_scaled)

print('Y_pred = ',Y_pred)
```

```
Y_pred =  [2 1 1 2 2 2 2 0 0 2 1 0 0 0 2 0 0 0 2 1 0 2 1 1 0 2 2 1 1 1]
```

F. Độ chính xác kết quả của thuật toán

Sau khi mô hình đã được huấn luyện, chúng ta cần đánh giá độ chính xác để xem thuật toán có hoạt động ổn định và chính xác không. Có nhiều phương pháp có thể được sử dụng để kiểm tra độ chính xác của kết quả thuật toán, trong đó bao gồm:

Hàm accuracy_score(Y_test, Y_pred)

Ma trận nhầm lẫn (confusion matrix)

Receiver Operating Characteristic - Area Under Curve(ROC-AUC)

Hàm accuracy_score (Y_test, Y_pred) thuộc thư viện ‘’sklearn.metrics’’.

Trong đó:

Y_test: Là mảng chứa nhãn thực tế của dữ liệu kiểm tra (test set).

Y_pred: Là mảng chứa nhãn được dự đoán bởi mô hình cho dữ liệu kiểm tra.

Hàm so sánh từng phần tử trong Y_test với tương ứng trong Y_pred để đánh giá xem mô hình dự đoán đúng bao nhiêu phần trăm so với nhãn thực tế.

Độ chính xác của hàm được tính bằng tỉ lệ giữa số lượng dự đoán đúng và tổng số lượng mẫu trong tập kiểm tra

$$Accuracy = \frac{Số\ lượng\ dự\ đoán\ đúng}{Tổng\ số\ lượng\ mẫu}$$

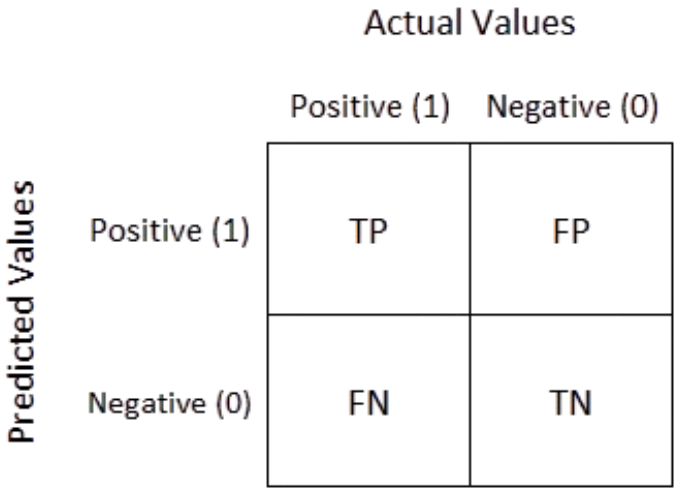
G. Ma trận nhầm lẫn (Confusion Matrix):

Ma trận nhầm lẫn có dạng như hình bên. Trong đó:

- TP (True Positon): Số lượng mẫu được dự đoán đúng là positive (dự đoán đúng nhãn positive).
- FN (False Nagetives): Số lượng mẫu được dự đoán là negative mặc dù thực tế là positive (dự đoán sai nhãn negative cho các mẫu positive).

TP (True Positon): Số lượng mẫu được dự đoán là positive mặc dù thực tế là negative (dự đoán sai nhãn positive cho các mẫu negative).

FN (False Nagetives): Số lượng mẫu được dự đoán đúng là negative (dự đoán đúng nhãn negative).



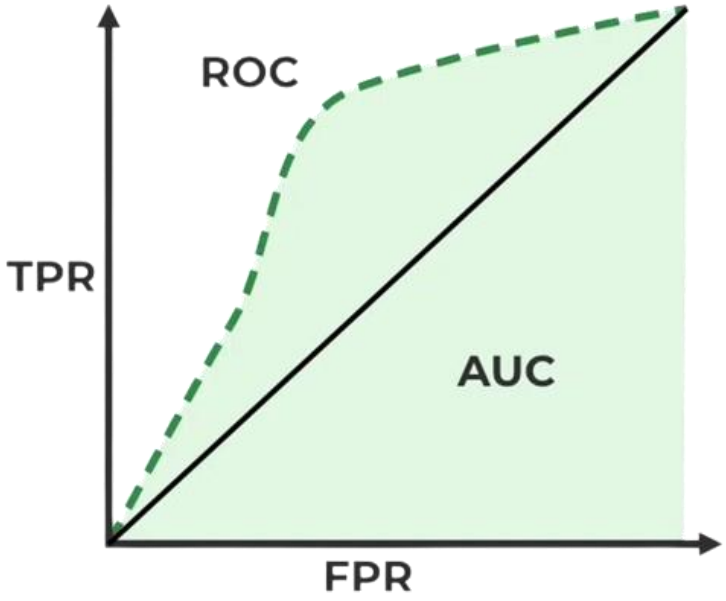
Độ chính xác (accuracy) của một confusion matrix được tính bằng tỉ lệ giữa tổng số dự đoán đúng và tổng số lượng mẫu trong tập dữ liệu.

$$Accuracy = \frac{True\ positives(TP) + True\ Negatives(TN)}{Tổng\ số\ lượng\ mẫu}$$

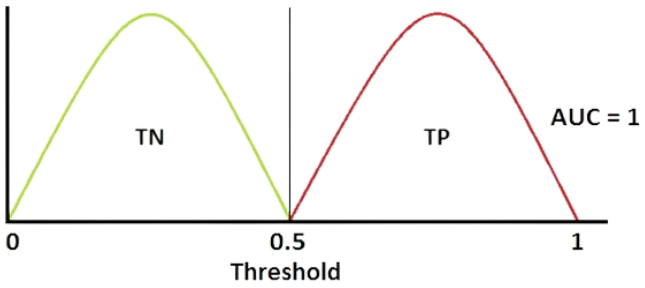
H. Receiver Operating Characteristic – Are Under Curve(ROC-AUC):

ROC là một đồ thị biểu diễn hiệu suất của một mô hình phân loại trên toàn bộ ngưỡng quyết định (thresholds). Đồ thị ROC biểu diễn tỷ lệ True Positive Rate (TPR) trên trục tung và False Positive Rate (FPR) trên trục hoành. AUC là diện tích dưới đường cong ROC.

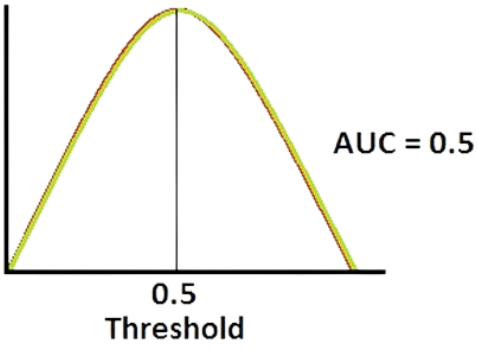
ROC - AUC là tỷ lệ diện tích dưới đường cong ROC và diện tích hình chữ nhật có cạnh là 1. Giá trị ROC-AUC thường nằm trong khoảng [0; 1].



Khi một giá trị ROC-AUC gần 1 cho thấy mô hình có khả năng phân loại tốt hơn. Mô hình phân loại chính xác hơn khi 2 đường cong không chồng lên nhau.



Khi ROC-AUC bằng 0.5, điều đó có nghĩa là mô hình không có khả năng phân loại tốt hơn ngẫu nhiên.



Vấn đề thường gặp trong thuật toán KNN:

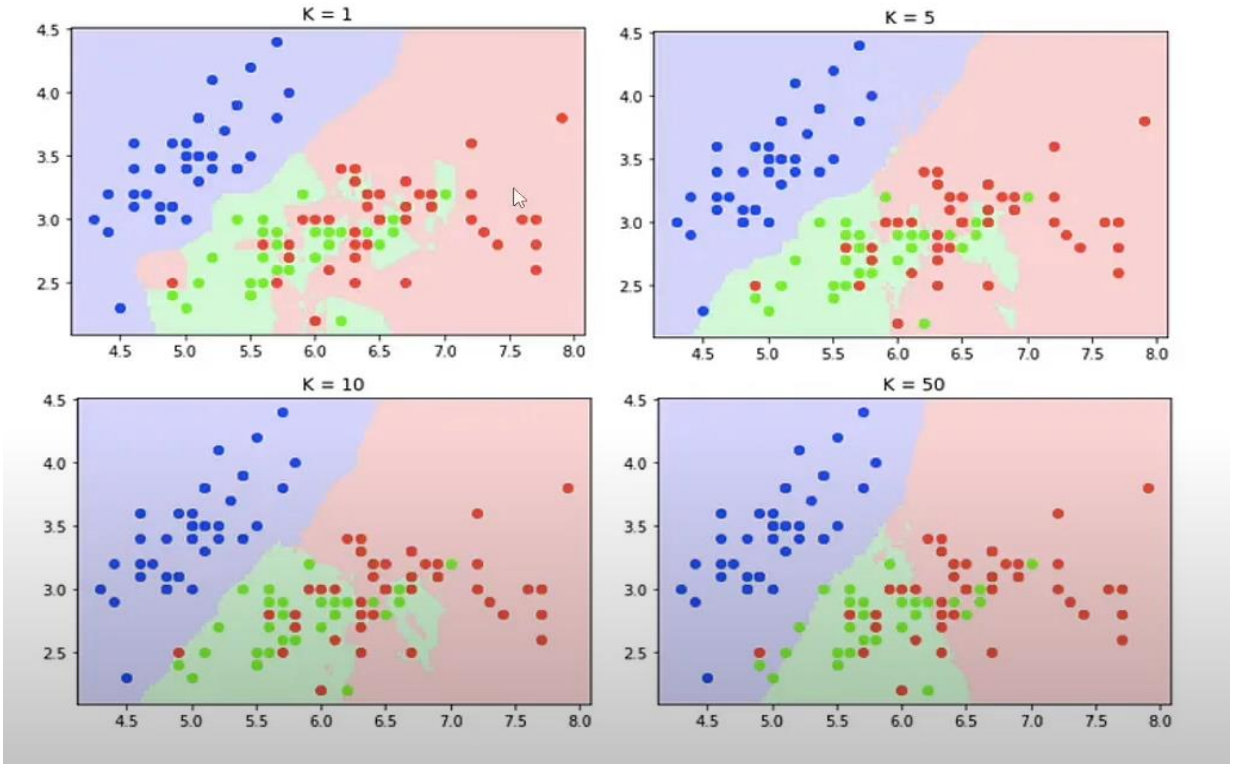
Trong machine learning nói chung và KNN nói riêng, luôn có 2 vấn đề thường xuất hiện, đó là overfitting và underfitting:

Overfitting (quá mức khớp): Khi một mô hình máy học quá tinh chỉnh để phù hợp với dữ liệu huấn luyện, đến mức nó bắt đầu "nhớ" cụ thể các điểm dữ liệu trong tập huấn luyện, thay vì học các mẫu. Kết quả là mô hình này có thể hoạt động rất tốt trên dữ liệu huấn luyện, nhưng hiệu suất của nó sẽ kém khi áp dụng vào dữ liệu mới.

Underfitting (thiếu khớp): Trái ngược với overfitting, underfitting xảy ra khi một mô hình không đủ phức tạp để học được cấu trúc của dữ liệu. Mô hình đơn giản quá không thể biểu diễn đúng các mẫu phức tạp trong dữ liệu huấn luyện. Kết quả là mô hình này không hoạt động tốt trên cả dữ liệu huấn luyện và dữ liệu mới.

Việc chọn giá trị k sẽ ảnh hưởng đến tỉ lệ xuất hiện overfitting và underfitting trong chương trình:

- Nếu k càng bé thì khả năng dẫn đến overfitting càng cao. Khi đó model trở nên rắc rối nhưng bù lại là độ chính xác là 100%.
- Ngược lại, k càng lớn thì thuật toán dễ bị underfitting. Khi đó mặc dù model nhìn đơn giản hơn nhưng độ chính xác lại không cao.



Tỉ lệ xuất hiện overfitting và underfitting ảnh hưởng bởi giá trị k

IV. DECISION TREES:

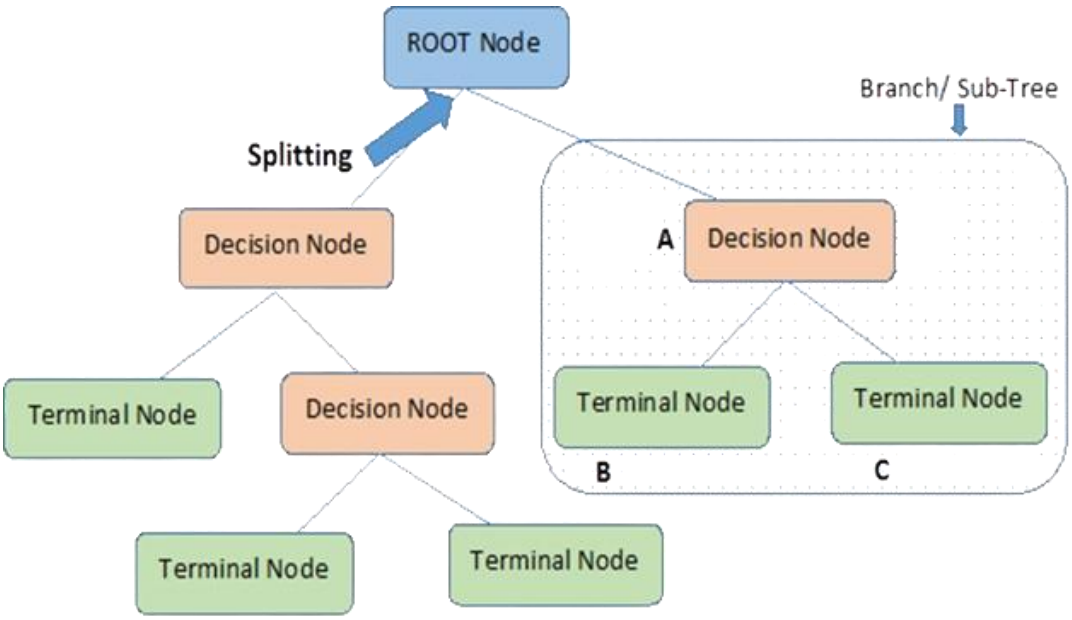
A. Khái niệm:

Decision tree (Cây quyết định) là một thuật toán học máy được giám sát, có thể được sử dụng cho cả các vấn đề hồi quy (dự đoán giá trị liên tục) và phân loại (dự đoán giá trị phân loại). Hơn nữa chúng còn đóng vai trò là nền tảng cho các kĩ thuật nâng cao hơn, chẳng hạn như đóng bao, tăng cường và rừng ngẫu nhiên.

B. Sơ đồ minh họa:

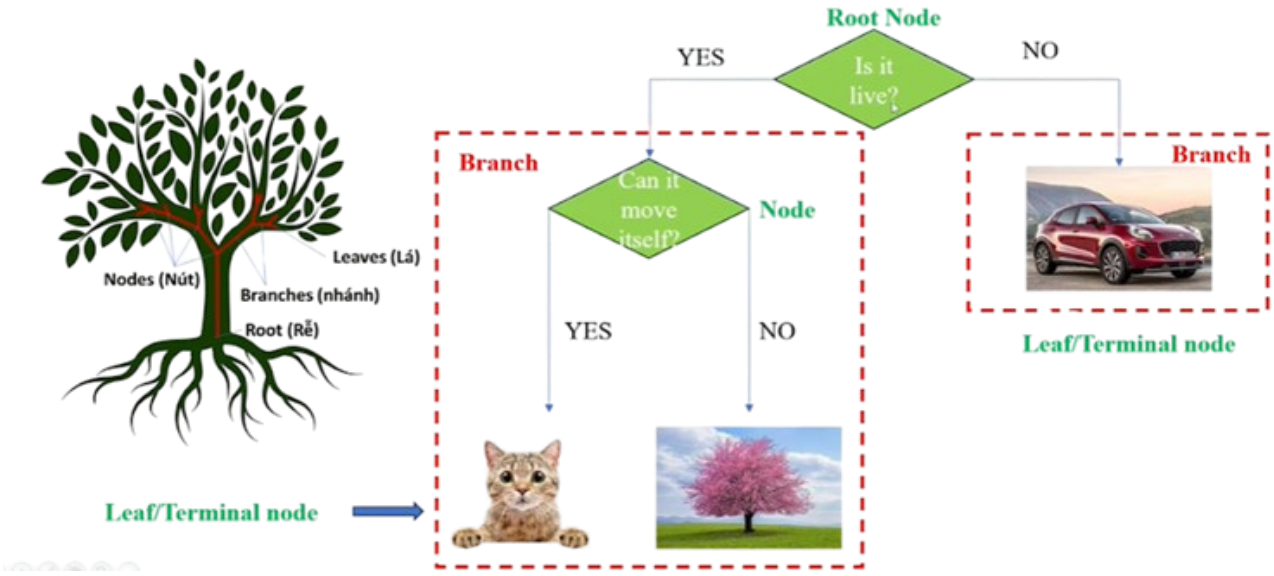
Cây quyết định bắt đầu bằng **nút gốc (ROOT)**, sau đó tách thành hai hoặc nhiều nhóm thống nhất thông qua một phương pháp gọi là phân tách.

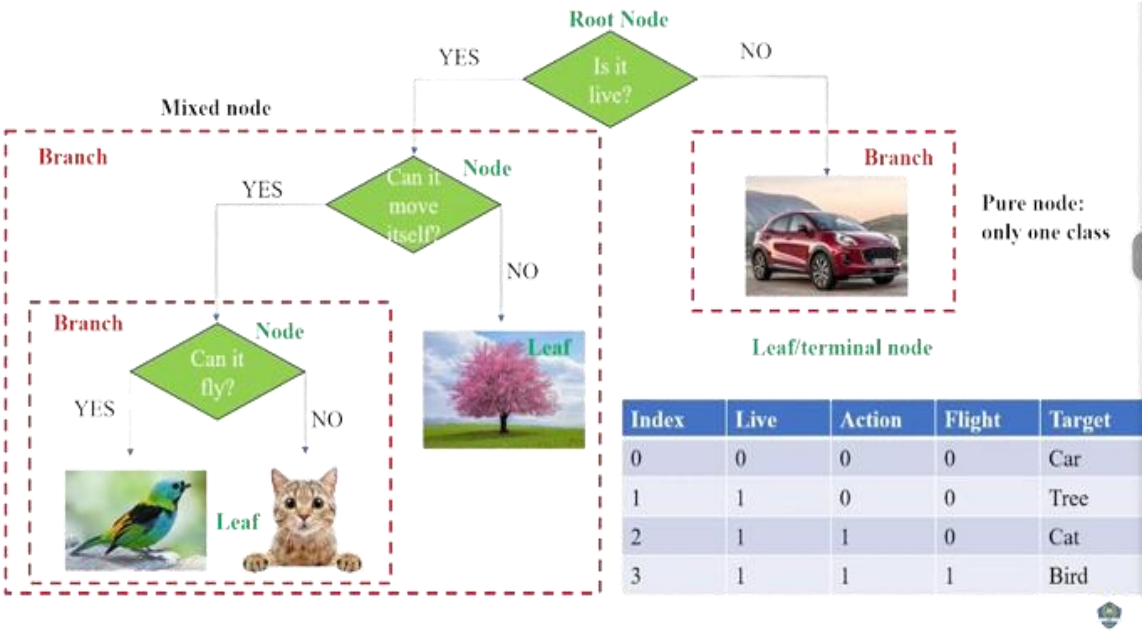
Khi các nút phụ trải qua quá trình phân chia sâu hơn, chúng được xác định là nút quyết định, trong khi các nút không phân chia được gọi **là nút cuối hoặc nút lá**. Một đoạn của cây hoàn chỉnh được gọi là **nhánh.**



Note:- A is parent node of B and C.

C. Ví dụ minh hoạ:

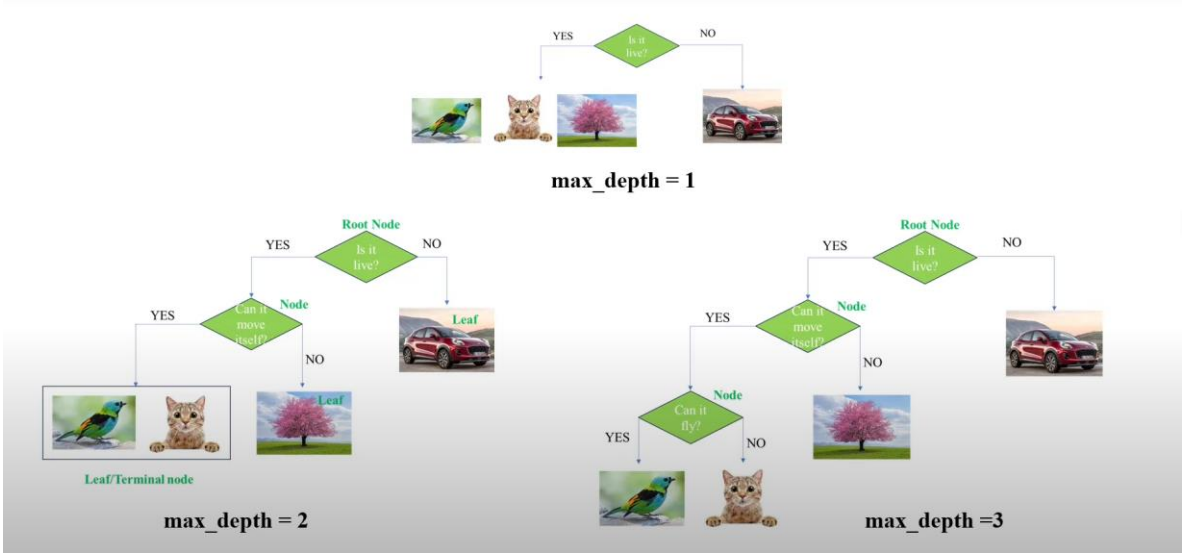




Index	Live	Action	Flight	Target
0	0	0	0	Car
1	1	0	0	Tree
2	1	1	0	Cat
3	1	1	1	Bird

- Node chứa nhiều hơn 1 loại dữ liệu gọi là Mixed Node.
- Node chứa 1 loại dữ loại gọi là Pure Node.

D. Độ Sâu Tối Đa:



Điều khiển độ phức tạp của decision trees:

- max_depth: Độ sâu tối đa của cây.
- max_leaf_node: Tối đa của số nút lá.
- min_samples_leaf: Số mẫu tối thiểu của một lá có thể phân tích.

Overfitting	Underfitting
-Mô hình quá phức tạp -Sai số thấp trên dữ liệu huấn luyện -Sai số cao trên dữ liệu kiểm tra (test set)	-Mô hình quá đơn giản -Sai số cao trên dữ liệu huấn luyện (training set) -Khả năng dự đoán kém

E. Ưu điểm và nhược điểm

1. Ưu điểm

-Đơn giản để hiểu và giải thích.

-Yêu cầu chuẩn bị ít dữ liệu. Các kỹ thuật khác thường yêu cầu chuẩn hóa dữ liệu, cần tạo các biến giả và xóa các giá trị trống.

-Chi phí sử dụng cây (tức là dự đoán dữ liệu) là logarit của số điểm dữ liệu được sử dụng để huấn luyện cây.

-Có khả năng xử lý cả dữ liệu số và phân loại. Tuy nhiên, việc triển khai scikit-learn hiện không hỗ trợ các biến phân loại. Các kỹ thuật khác thường chuyên dùng để phân tích các tập dữ liệu chỉ có một loại biến.

-Có khả năng xử lý các vấn đề đa đầu ra.

-Sử dụng mô hình hộp trắng: Nếu một tình huống nhất định có thể quan sát được trong một mô hình thì việc giải thích cho điều kiện đó có thể dễ dàng được giải thích bằng logic boolean. Ngược lại, trong mô hình hộp đen (ví dụ: trong mạng lưới thần kinh nhân tạo), kết quả có thể khó diễn giải hơn.

-Có thể xác nhận một mô hình bằng cách sử dụng các bài kiểm tra thống kê, giải thích được độ tin cậy của mô hình.

-Hoạt động tốt ngay cả khi các giả định bị vi phạm bởi mô hình thực mà dữ liệu được tạo ra.

2. Nhược điểm:

-Người học cây quyết định có thể tạo ra những cây quá phức tạp và không khái quát hóa tốt dữ liệu. Điều này được gọi là trang bị quá mức. Các cơ chế như cắt tỉa, đặt số lượng mẫu tối thiểu cần thiết tại một nút lá hoặc đặt độ sâu tối đa của cây là cần thiết để tránh vấn đề này.

-Cây quyết định có thể không ổn định vì những thay đổi nhỏ trong dữ liệu dẫn đến việc tạo ra một cây hoàn toàn khác. Vấn đề này được giảm thiểu bằng cách sử dụng cây quyết định trong một tập hợp.

-Dự đoán của cây quyết định không trơn tru cũng không liên tục mà xấp xỉ không đổi từng phần.

-Thuật toán học cây quyết định dựa trên các thuật toán heuristic như thuật toán tham lam trong đó các quyết định tối ưu cục bộ được đưa ra tại mỗi nút. Các thuật toán như vậy không thể đảm bảo trả về cây quyết định tối ưu toàn cục.

-Có những khái niệm khó học vì cây quyết định không thể hiện chúng một cách dễ dàng, chẳng hạn như các vấn đề về XOR, tính chẵn lẻ hoặc bộ ghép kênh.

-Người học cây quyết định tạo ra cây thiên vị nếu một số lớp chiếm ưu thế. Do đó, nên cân bằng tập dữ liệu trước khi khớp với cây quyết định.

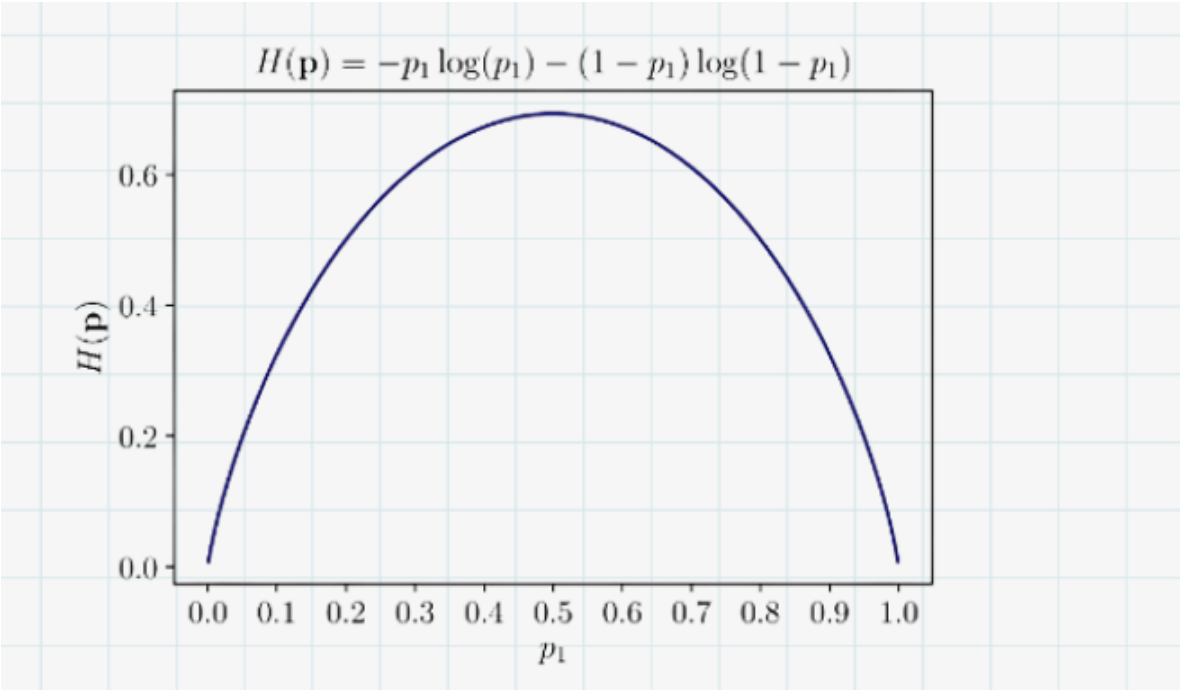
F. Hàm số entropy

Cho một phân phối xác suất của một biến rời rạc x có thể nhận n giá trị khác nhau $x_1, x_2, \dots, x_n$. Giả sử rằng xác suất để x nhận các giá trị này là $p_i = p(x = x_i)$ với $0 \leq p_i \leq 1, \sum_{i-1}^{n} p_i = 1$. Ký hiệu phân phối này là $\boldsymbol{p} = (p_1, p_2, \dots, p_n)$. Entropy của phân phối này được định nghĩa là:

$$H(\boldsymbol{p}) = -\sum_{i=1}^{n} p_i \log(p_i)$$

trong đó log là logarit tự nhiên (Một số tài liệu dùng logarit cơ số 2, nhưng giá trị của $H(\boldsymbol{p})$ chi khác đi bằng cách nhân với một hằng số) và quy ước $0\log(0) = 0$.

Xét một ví dụ với $n = 2$, trong trường hợp p là tinh khiết nhất, tức một trong hai giá trị p_i bằng 1, giá trị kia bằng 0 , entropy của phân phối này là $H(\boldsymbol{p}) = 0$. Khi $\boldsymbol{p}$ là vẩn đục nhất, tức cả hai giá trị $p_i = 0.5$, hàm entropy đạt giá trị cao nhất.



Đồ thị của hàm Entropy với n =2

G. Tìm hiểu Code:

DecisionTreeClassifier là lớp có khả năng thực hiện phân loại nhiều lớp trên tập dữ liệu.

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
```

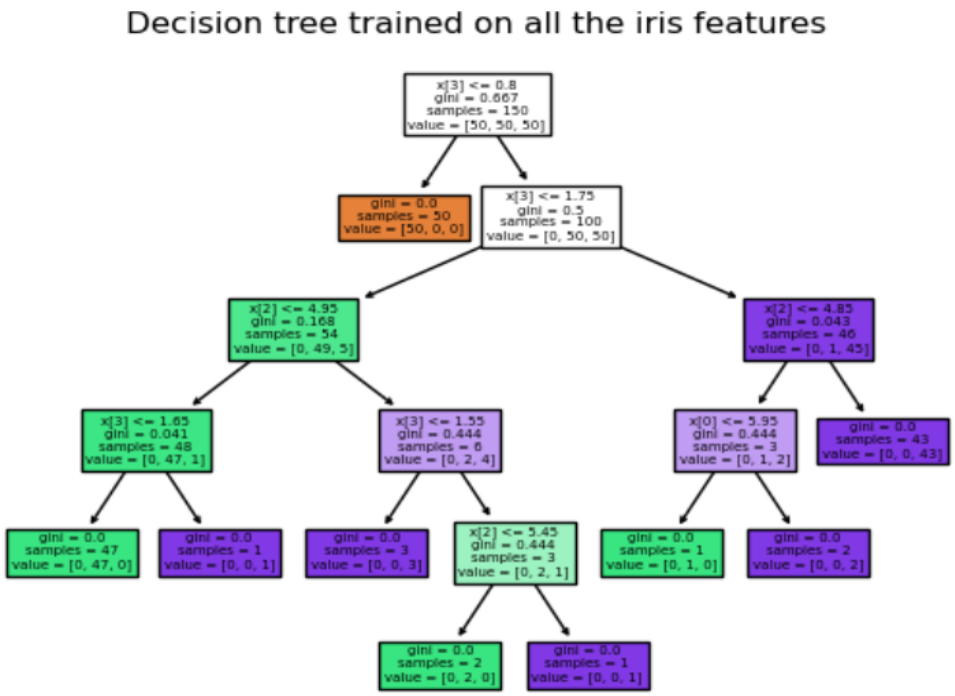
Sau khi được trang bị, mô hình có thể được sử dụng để dự đoán lại mẫu:

```
>>> clf.predict([[2., 2.]])
array([1])
```

Sau khi được đào tạo, có thể vẽ đồ thị cây với hàm plot_tree

Câu lệnh:

```
>>> tree.plot_tree(clf)
[...]
```



V. Kết luận:

Trong bài báo cáo này, nhóm chúng em đã khám phá hai mô hình học máy giám sát không tham số: K-Nearest Neighbors (KNN) và Decision Trees. KNN là một phương pháp đơn giản nhưng hiệu quả, sử dụng khoảng cách giữa các điểm dữ liệu để dự đoán. Decision Trees giúp tạo ra các cây quyết định rõ ràng và dễ hiểu, dựa trên các đặc điểm của dữ liệu.

Cả 2 mô hình này đều có những ưu điểm và hạn chế riêng, nhưng đều chứng minh được giá trị trong nhiều ứng dụng thực tế. KNN mạnh mẽ trong việc xử lý các bài toán với dữ liệu không có cấu trúc phức tạp, trong khi Decision Trees lại dễ dàng trực quan hóa và giải thích quá trình ra quyết định.

Tóm lại, việc hiểu và áp dụng đúng các mô hình học máy giám sát không tham số như KNN và Decision Trees sẽ giúp chúng ta giải quyết nhiều vấn đề phức tạp trong thực tiễn, từ phân loại, dự đoán cho đến phân tích dữ liệu.

Nhóm chúng em xin chân thành cảm ơn sự tận tâm của thầy giáo trong 15 tuần học vừa qua, qua đó cung cấp cho nhóm chúng em những kiến thức vô cùng cần thiết để có thể hoàn thành được bài tập lần này. Một lần nữa chúng em xin chân thành cảm ơn thầy.

Tài liệu tham khảo

1. Thevapalan, A., & Le, J. (2023, June 1). *Decision Trees in Machine Learning Using R*. https://www.datacamp.com/tutorial/decision-trees-R
2. Vu, T. (2018, January 14). *Bài 34: Decision Trees (1): Iterative Dichotomiser 3*. Tiep Vu's Blog. https://machinelearningcoban.com/2018/01/14/id3/
3. Vu, T. (2017, January 8). *Bài 6: K-nearest neighbors*. Tiep Vu's Blog. https://machinelearningcoban.com/2017/01/08/knn/
4. *CodeLearn*. (n.d.). https://codelearn.io/sharing/thuat-toan-k-nearest-neighbors-knn