

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT**



## **BÁO CÁO SEMINAR**

### **K-NEAREST NEIGHBORS**

Link Google Colab:

[https://colab.research.google.com/drive/1FBBnPm1ViGk4BUCKNVGS\\_RKFojNw6jM5?  
authuser=1](https://colab.research.google.com/drive/1FBBnPm1ViGk4BUCKNVGS_RKFojNw6jM5?authuser=1)

**Môn học:** Phân tích dữ liệu với R/Python

**Mã học phần:** 222IS2901

**Giảng viên:** Thầy Nguyễn Phát Đạt

**Trợ giảng:** Anh Trần Lê Tấn Thịnh

*Thành phố Hồ Chí Minh, ngày 09 tháng 03 năm 2023*

# MỤC LỤC

<b>LỜI CẢM ƠN</b>	6
<b>Chương 1: Thuật toán KNN</b>	7
1.1 Giới thiệu thuật toán	7
1.2 Cơ sở lý thuyết của thuật toán (toán học)	8
1.2.1 Định nghĩa	8
1.2.2 Quy trình làm việc của thuật toán KNN	8
1.2.3 Khoảng cách trong không gian vector	10
1.2.3.1 Định nghĩa	11
1.2.3.2 Một số norm thường dùng	11
1.3 Thuật toán KNN trong Python	12
1.4 Đo lường KNN	13
1.4.1 Đo lường KNN bằng Confusion matrix	13
1.4.1.1 Accuracy	14
1.4.1.2 Precision	14
1.4.1.3 Recall	14
1.4.1.4 F1 score	15
1.4.2 Đo lường KNN bằng AUC	15
1.5 Ứng dụng của thuật toán	17
1.6 Ưu điểm và nhược điểm & vấn đề mở rộng	21
1.6.1 Ưu điểm	21
1.6.2 Nhược điểm	21
<b>Chương 2: Ứng dụng vào bài toán</b>	23
2.1 Mô tả bài toán	23
2.2 Mô tả dữ liệu	23
2.3 Quy trình đề xuất	24
2.4 Phân tích và khai thác dữ liệu (EDA)	24

2.4.1 Phân tích và khai thác dữ liệu EDA	24
2.4.2 Tiền xử lý dữ liệu	34
2.5 Xây dựng mô hình & đánh giá	37
2.5.1 Mô hình KNN với sự thay đổi hệ số k	37
2.5.1.1 Vấn đề Overfitting và Underfitting trong KNN	37
2.5.1.2 Cách chọn k tối ưu	38
2.5.1.3 Thực hiện tìm k tối ưu	38
2.5.2 Mô hình KNN với lời nguyên của chiều dữ liệu	40
2.5.3 Mô hình khác có thể áp dụng vào bài toán - Decision Trees	42
2.6 Đánh giá và Kết luận	44
2.6.1 Mô hình KNN với K tối ưu ( $n_{\text{neighbor}} = 9$ )	46
2.6.2 Mô hình KNN với lời nguyên dữ liệu ( $n_{\text{features}} = 1$ )	50
2.6.3 Mô hình Decision Tree	53
2.6.4 So sánh các mô hình và lựa chọn	55
<b>Phụ lục: Báo cáo quá trình làm việc nhóm</b>	<b>58</b>
<b>Tài liệu tham khảo:</b>	<b>61</b>

## Danh mục bảng biểu

Hình 1. Ví dụ minh họa thuật toán KNN	9
Hình 2. Norm 1 và norm 2 trong không gian hai chiều	12
Hình 3. Confusion matrix	13
Hình 4: Đồ thị ROC	16
Hình 5: Bảng số liệu Height - Age - Weight	17
Hình 6: Biểu thị dữ liệu lên biểu đồ	17
Hình 7: Khoảng cách giữa điểm ID với các điểm cho trước lên	18
Hình 8: Chọn k=3	18
Hình 9: Chọn k=1, k=5	19
Hình 10: Kết quả kiểm tra dữ liệu	24
Hình 11: Kết quả kiểm tra giá trị null	25
Hình 12: Bảng dữ liệu sau khi xóa cột User ID	26
Hình 13: Kiểm tra giá trị trung bình, độ lệch chuẩn, min, max.	26
Hình 14. Biểu đồ của biến Age và EstimatedSalary	27
Hình 15. Biểu đồ của biến Age, EstimatedSalary và Purchased	28
Hình 16. Biểu đồ hình tròn của biến Gender	29
Hình 17. Biểu đồ thanh của biến mục tiêu Purchased	29
Hình 18. Biểu đồ hình tròn của biến mục tiêu Purchased	30
Hình 19. Sơ đồ phân tán của biến Age, EstimatedSalary và Purchased	31
Hình 20. Biểu đồ của biến Gender, Age và Purchased	32
Hình 21. Biểu đồ của biến Gender, EstimatedSalary và Purchased	33
Hình 22. Ma trận tương quan của biến Age, EstimatedSalary và Purchased	34
Hình 23. Biến độc lập được phân chia ra	35
Hình 24. Biến phụ thuộc của bài toán	35
Hình 25. Sử dụng LabelEncoder cho cột Gender	36
Hình 26. Sử dụng StandardScaler cho tập dữ liệu	37
Hình 27. Độ chính xác của KNN theo k	40

Hình 28. Ma trận tương quan giữa Age, Estimated Salary và Purchased	42
Hình 29. Đồ thị KNN thay đổi hệ số k	45
Hình 30. Đồ thị ROC với giá trị k=9	47
Hình 31. Ma trận nhầm lẫn	48
Hình 32. Đồ thị ROC với n_neighbors=9 và n_features =1	51
Hình 33. Ma trận nhầm lẫn n_features = 2	52
Hình 34. Đồ thị ROC của Decision Tree	54
Hình 35. Ma trận nhầm lẫn mô hình decision tree	55
Hình 36. Biểu đồ so sánh chỉ số đo lường	56

## LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Phát Đạt - Giảng viên hướng dẫn bộ môn Phân tích dữ liệu với R/Python và anh Trần Lê Tấn Thịnh - Trợ giảng bộ môn đã tận tình hướng dẫn và truyền đạt những kiến thức đầy bổ ích cho chúng em trong suốt khoảng thời gian vừa qua. Nhờ đó, chúng em có cách nhìn đúng đắn và sâu sắc hơn để có thể hoàn thiện đồ án môn học này và vận dụng kiến thức cho công việc thực tế về sau.

Đối với chúng em, môn học “Phân tích dữ liệu R/Python” mang tính thực tế khá cao và thú vị. Trong suốt quá trình đồng hành cùng nhau, mỗi thành viên trong nhóm đều cố gắng để tìm tòi tài liệu và hoàn thành công việc một cách chỉnh chu nhất, tuy nhiên do vốn kiến thức và kinh nghiệm làm bài còn hạn chế nên bài nghiên cứu vẫn có nhiều chỗ chưa hoàn thiện. Vậy nên, nhóm chúng em rất mong có thể nhận được sự nhận xét và góp ý từ thầy và anh để nhóm có thể rút kinh nghiệm và hoàn thành tốt hơn cho những đề tài sau.

Một lần nữa, nhóm chúng em xin cảm ơn thầy và anh!

## Chương 1: Thuật toán KNN

Tóm tắt chương 1: Chương 1 nhằm mục đích giới thiệu tổng quan về thuật toán K-nearest neighbors, đi sâu vào việc cung cấp các khái niệm về các thuật ngữ, cơ sở lý thuyết, cách thức đo lường, đánh giá thuật toán cũng như trình bày về một số bài toán, ứng dụng phổ biến của K-nearest neighbors.

### 1.1 Giới thiệu thuật toán

Mở đầu với một ví dụ đơn giản như sau: Giả sử nay lớp bạn có một tiết kiểm tra đột xuất, đề trắc nghiệm có một câu khó và lạ trong khi bạn chưa ôn bài kỹ trước nên bạn quyết định hỏi đáp án của các bạn xung quanh để có một đáp án chắc chắn hơn. Sau một hồi hỏi thì bạn thu được kết quả có 2 người cho đáp án A và 1 người cho đáp án B. Qua kết quả trên thì xu hướng các bạn thường đưa ra là quyết định mạnh dạn chọn đáp án A cho câu hỏi đó đúng không nào? Bởi lẽ chúng ta thường thấy rằng sự lựa chọn nào càng có nhiều sự đồng tình thì thường sẽ là lựa chọn chính xác, tương tự với đó, cũng có một thuật toán mà nếu trong tập dữ liệu mình thu thập được có các dữ liệu tương tự, gần nhau càng nhiều thì dự đoán đưa ra sẽ càng chính xác.

K-nearest neighbor - một trong những thuật toán supervised-learning đơn giản nhất mà hiệu quả trong một vài trường hợp trong Machine Learning. Ý tưởng của thuật toán này là khi training, nó sẽ không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.

Ví dụ trên được xếp vào dạng bài toán Classification có thể ứng dụng thuật toán KNN để giải quyết với ý tưởng label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong đề thi) sẽ được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set gần nó nhất (đáp án của các bạn xung quanh).

## 1.2 Cơ sở lý thuyết của thuật toán (toán học)

### 1.2.1 Định nghĩa

K-Nearest neighbor (KNN) là một trong những thuật toán học có giám sát đơn giản nhất trong Machine Learning. Ý tưởng của KNN là tìm ra output của dữ liệu dựa trên thông tin của những dữ liệu training gần nó nhất.

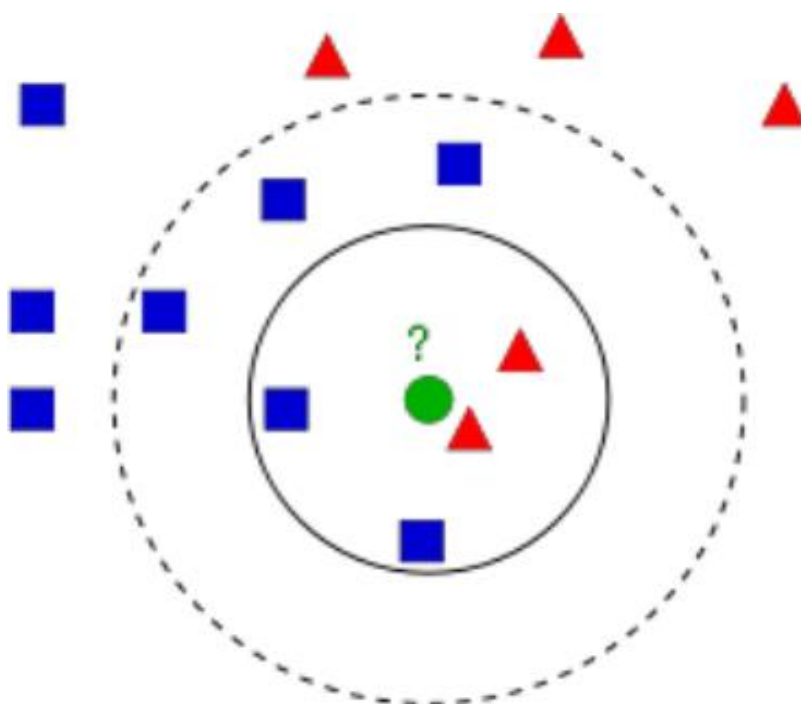
### 1.2.2 Quy trình làm việc của thuật toán KNN

- Bước 1: Xác định tham số  $K$  = số láng giềng gần nhất.
- Bước 2: Tính khoảng cách đối tượng cần phân lớp với tất cả các đối tượng trong training data.
- Bước 3: Sắp xếp khoảng cách theo thứ tự tăng dần và xác định  $K$  láng giềng gần nhất với đối tượng cần phân lớp.
- Bước 4: Lấy tất cả các lớp của  $K$  láng giềng gần nhất.
- Bước 5: Dựa vào phần lớn lớp của  $K$  để xác định lớp cho đối tượng cần phân lớp.

*Ví dụ minh họa về thuật toán KNN:*

Giả sử bài toán được đặt ra: mình mới quen một người bạn, mình cần biết người bạn này có thích học tiếng Anh hay không. Qua thời gian tìm hiểu mình đã thu thập được một số dữ liệu và đã biểu hiện dưới dạng hình vẽ dưới đây.





*Hình 1. Ví dụ minh họa thuật toán KNN*

Ta dễ dàng nhìn thấy có hai loại: hình vuông màu xanh biểu diễn cho những người thích học tiếng anh, tam giác màu đỏ biểu diễn cho những người không thích học tiếng anh, hình tròn màu xanh là người bạn mình muốn biết có thích học tiếng anh hay không, khoảng cách giữa chấm tròn và các điểm còn lại biểu diễn độ thân thiết của bạn đó với những người bạn.

Phương pháp đơn giản nhất để kiểm tra xem bạn đó chơi thân với người bạn nào nhất, tức là tìm xem điểm gần chấm xanh thuộc class nào (hình vuông hay tam giác). Từ hình trên ta dễ dàng nhận thấy điểm gần chấm xanh nhất là hình tam giác màu đỏ, do đó nó sẽ được phân vào lớp tam giác màu đỏ.

Có một vấn đề trong phương pháp trên, xung quanh chấm xanh xuất hiện rất nhiều hình vuông màu xanh nên việc xét điểm gần nhất là chưa khả thi. Vì vậy, ta sẽ xét  $k$  điểm gần nhất. Giả sử, ta lấy  $K=3$ , dựa theo hình trên ta dễ dàng nhận ra có hai hình tam giác đỏ và một hình vuông xanh có khoảng cách gần chấm xanh nhất, do đó chấm xanh được phân vào lớp tam giác đỏ. Lấy  $K=7$ , ta có năm hình vuông xanh và hai hình tam giác đỏ, lúc này chấm xanh được xếp vào lớp hình vuông xanh. Trường hợp lấy  $K=4$ , ta nhận thấy sẽ có hai hình vuông xanh và hai hình tam giác đỏ, đây là

trường hợp có điểm bằng nhau, với trường hợp này KNN sẽ xử lý bằng cách so sánh tổng khoảng cách của các hình gần nhất với điểm ta đang xét.

Do xuất hiện trường hợp có điểm bằng nhau, vì vậy người ta thường chọn  $k$  là số lẻ. Đó cũng là ý tưởng của KNN.

### 1.2.3 Khoảng cách trong không gian vector

Trong không gian một chiều, việc đo khoảng cách giữa hai điểm đã rất quen thuộc: lấy trị tuyệt đối của hiệu giữa hai giá trị đó. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu  $x, y$  có  $k$  thuộc tính:

**Distance functions**

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k  x_i - y_i $
Minkowski	$\left( \sum_{i=1}^k ( x_i - y_i )^q \right)^{1/q}$

Trong không gian hai chiều, tức mặt phẳng, chúng ta thường dùng khoảng cách Euclid để đo khoảng cách giữa hai điểm.

Việc đo khoảng cách giữa hai điểm dữ liệu nhiều chiều, tức hai vector, là rất cần thiết trong Machine Learning. Chúng ta cần đánh giá xem điểm nào là điểm gần nhất của một điểm khác; chúng ta cũng cần đánh giá xem độ chính xác của việc ước lượng; và trong rất nhiều ví dụ khác nữa.

Để xác định khoảng cách giữa hai vector  $y$  và  $z$ , người ta thường áp dụng một hàm số lên vector hiệu  $x = y - z$ . Một hàm số được dùng để đo các vector cần có một vài tính chất đặc biệt.

### 1.2.3.1 Định nghĩa

Một hàm số  $f()$  ánh xạ một điểm  $x$  từ không gian  $n$  chiều sang tập số thực một chiều được gọi là norm nếu nó thỏa mãn ba điều kiện sau đây:

- $f(x) \geq 0$ . Dấu bằng xảy ra  $x = 0$ .
- $f(\alpha x) = |\alpha|f(x), \forall \alpha \in \mathbb{R}$ .
- $f(x_1) + f(x_2) \geq f(x_1 + x_2), \forall x_1, x_2 \in \mathbb{R}$

### 1.2.3.2 Một số norm thường dùng

Giả sử các vector  $x = [x_1; x_2 \dots x_n], y = [y_1; y_2 \dots y_n]$ .

Nhận thấy khoảng cách Euclid chính là một norm, thường được gọi là norm 2:

$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots x_n^2} \quad (1)$$

Với  $p$  là một số không nhỏ hơn 1 bất kỳ, hàm số sau đây:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \dots |x_n|^p)^{\frac{1}{p}} \quad (2)$$

Được chứng minh thỏa mãn ba điều kiện trên, và được gọi là norm  $p$ .

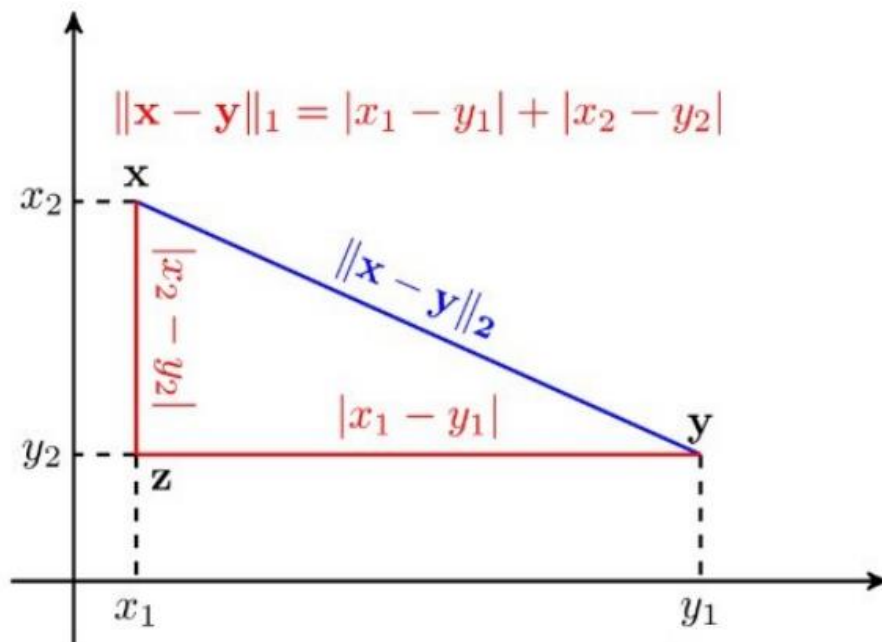
Nhận thấy rằng khi  $p \rightarrow 0$  thì biểu thức bên trên trở thành số các phần tử khác 0 của  $x$ . Hàm số (2) khi  $p=0$  được gọi là giả chuẩn (pseudo norm) 0. Nó không phải là norm vì nó không thỏa mãn điều kiện 2 và 3 của norm. Giả -chuẩn này, thường được ký hiệu là  $\|x\|_0$ , khá quan trọng trong ML vì trong nhiều bài toán, chúng ta cần có ràng buộc “sparse”, tức số lượng thành phần “active” của  $x$  là nhỏ.

Có một vài giá trị của  $p$  thường được dùng:

- Khi  $p = 2$  chúng ta có norm2 như ở trên.
- Khi  $p = 1$  chúng ta có:

$$\|x\|_1 = |x_1| + |x_2| + |x_3| + \dots |x_n| \quad (3)$$

Là tổng các giá trị tuyệt đối của từng phần tử của  $x$ . Norm 1 thường được dùng như xấp xỉ của norm 0 trong các bài toán có ràng buộc. Dưới đây là một ví dụ so sánh norm 1 và norm 2 trong không gian hai chiều:



Hình 2. Norm 1 và norm 2 trong không gian hai chiều

Norm 2 (màu xanh) chính là đường chim bay nối giữa vector  $x$  và vector  $y$ . Khoảng cách norm 1 giữa hai điểm này (màu đỏ) có thể diễn giải như là đường đi từ  $x$  đến  $y$  trong một thành phố mà thành phố được tạo hình bàn cờ, chúng ta chỉ có thể đi theo dọc bàn cờ chứ không thể đi theo đường thẳng.

Khi  $p \rightarrow \infty$ , ta có norm  $p$  chính là trị tuyệt đối của phần tử lớn nhất của vector đó:

$$\|x\|_{\infty} = \max_{i=1,2,\dots,n} |x_i|$$

### 1.3 Thuật toán KNN trong Python

Để tiến hành sử dụng thuật toán KNN nhằm giải quyết bài toán phân loại thì bước đầu chúng ta cần nạp thư viện.

Sau đây là ví dụ nạp thư viện “sklearn” để huấn luyện mô hình

```
import sklearn
from sklearn.neighbors import KNeighborsClassifier
```

Tiếp theo là ví dụ đào tạo và dự đoán để phân loại:

```
KNclassifier = KNeighborsClassifier(n_neighbors = i,
metric = 'minkowski', p = 2)
KNclassifier.fit(X_train, y_train)
```

Trong đó:

**n\_neighbors:** số láng giềng gần nhất (số  $K$  trong KNN)

**metric:** phương pháp tính khoảng cách (khoảng cách Minkowski thường được sử dụng phổ biến)

**p:(nếu có)** Giá trị  $p$  trong công thức tính khoảng cách Minkowski

**X\_train:** là biến độc lập của dữ liệu huấn luyện

**Y\_train:** là biến phụ thuộc dữ liệu huấn luyện

## 1.4 Đo lường KNN

### 1.4.1 Đo lường KNN bằng Confusion matrix

Confusion matrix là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Về cơ bản, confusion matrix thể hiện có bao nhiêu điểm dữ liệu thực sự thuộc vào một class, và được dự đoán là rơi vào một class. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Hình 3. Confusion matrix

Các chỉ số TP, FP, TN, FN lần lượt có ý nghĩa là :

- TP (True Positive): Tổng số trường hợp dự báo khớp Positive.
- TN (True Negative): Tổng số trường hợp dự báo khớp Negative.
- FP (False Positive): Tổng số trường hợp dự báo các quan sát thuộc nhãn Negative thành Positive.

- FN (False Negative): Tổng số trường hợp dự báo các quan sát thuộc nhãn Positive thành Negative.

Từ 4 chỉ số này, ta có thể đánh giá mức độ tin cậy của một mô hình bằng những chỉ số sau:

#### 1.4.1.1 Accuracy

Cách đơn giản và hay được sử dụng nhất là *accuracy* (độ chính xác). Cách đánh giá này đơn giản tính tỉ lệ giữa số điểm được dự đoán, phân lớp đúng và tổng số điểm trong tập dữ liệu kiểm thử.

Khi xây dựng mô hình phân loại chúng ta sẽ muốn biết một cách khái quát tỷ lệ các trường hợp được dự báo đúng trên tổng số các trường hợp là bao nhiêu. Tỷ lệ đó được gọi là độ chính xác. Độ chính xác giúp ta đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu. Độ chính xác càng cao thì mô hình của chúng ta càng chuẩn xác. Công thức tính độ chính xác:

$$\text{Accuracy} = \frac{TP + TN}{\text{total sample}}$$

#### 1.4.1.2 Precision

Precision trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng? Và tất nhiên precision càng cao thì mô hình của chúng ta càng tốt trong việc phân loại hồ sơ BAD (BAD chính là nhóm positive). Công thức của precision như sau:

$$\text{Precision} = \frac{TP}{TP + FP}$$

#### 1.4.1.3 Recall

Trong tất cả các trường hợp Positive, bao nhiêu trường hợp đã được dự đoán chính xác? Chỉ số này được tính theo công thức:

$$\text{Recall} = \frac{TP}{TP + FN}$$

#### 1.4.1.4 F1 score

Để đánh giá độ tin cậy chung của mô hình, người ta đã kết hợp 2 chỉ số Precision và Recall thành một chỉ số duy nhất: F1-score, được tính theo công thức:

$$F - \text{measure} = \frac{2 * \text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$

Một mô hình có chỉ số F-score cao chỉ khi cả 2 chỉ số Precision và Recall đều cao. Một trong 2 chỉ số này thấp đều sẽ kéo điểm F-score xuống. Trường hợp xấu nhất khi 1 trong hai chỉ số Precision và Recall bằng 0 sẽ kéo điểm F-score về 0. Trường hợp tốt nhất khi cả điểm chỉ số đều đạt giá trị bằng 1, khi đó điểm F-score sẽ là 1.

#### 1.4.2 Đo lường KNN bằng AUC

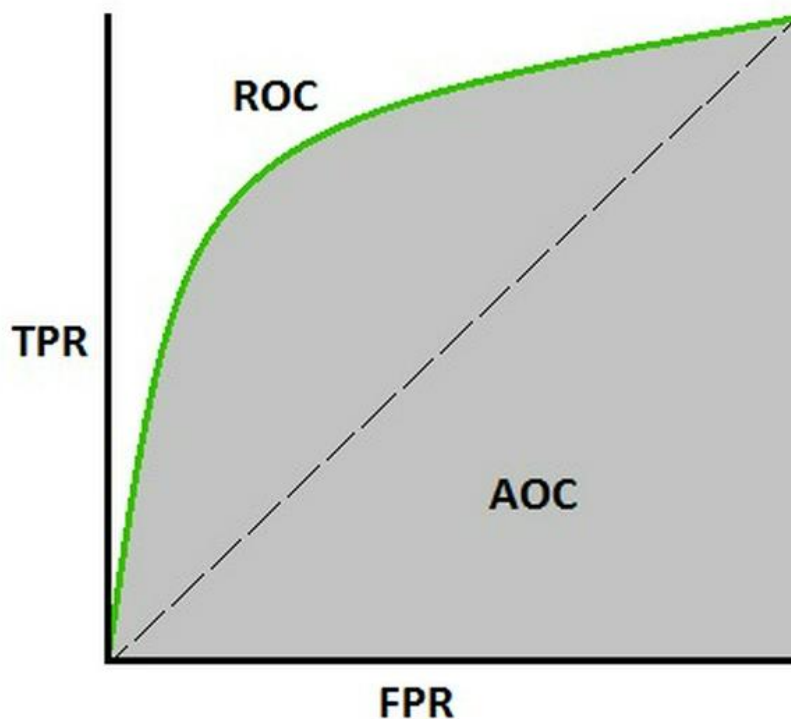
ROC là đường cong biểu diễn khả năng phân loại của một mô hình phân loại tại các ngưỡng threshold. Đường cong này dựa trên hai chỉ số :

- TPR (true positive rate): Hay còn gọi là recall hoặc sensitivity. Là tỷ lệ các trường hợp phân loại đúng positive trên tổng số các trường hợp thực tế là positive. Chỉ số này sẽ đánh giá mức độ dự báo chính xác của mô hình trên positive. Khi giá trị của nó càng cao, mô hình dự báo càng tốt trên nhóm positive. Nếu TPR=0.9, chúng ta tin rằng 90% các mẫu thuộc nhóm positive đã được mô hình phân loại đúng.

$$\text{TPR/recall/sensitivity} = \frac{TP}{\text{total positive}}$$

- FPR (false positive rate): Tỷ lệ dự báo sai các trường hợp thực tế là negative thành thành positive trên tổng số các trường hợp thực tế là negative. Nếu giá trị của  $FPR=0.1$ , mô hình đã dự báo sai 10% trên tổng số các trường hợp là negative. Một mô hình có FPR càng thấp thì mô hình càng chuẩn xác vì sai số của nó trên nhóm negative càng thấp. Phần bù của FPR là specificity đo lường tỷ lệ dự báo đúng các trường hợp negative trên tổng số các trường hợp thực tế là negative.

$$FPR = 1 - \text{specificity} = \frac{FP}{\text{total negative}}$$



Hình 4: Đồ thị ROC

AUC là chỉ số được tính toán dựa trên đường cong ROC (receiving operating curve) nhằm đánh giá khả năng phân loại của mô hình tốt như thế nào? Phần diện tích gạch chéo nằm dưới đường cong ROC và trên trục hoành là AUC (area under curve) có giá trị nằm trong khoảng  $[0, 1]$ . Khi diện tích này càng lớn thì đường cong ROC có xu hướng tiệm cận đường thẳng và khả năng phân loại của mô hình càng tốt. Khi đường cong ROC nằm sát với đường chéo đi qua hai điểm  $(0, 0)$  và  $(1, 1)$ , mô hình sẽ tương đương với một phân loại ngẫu nhiên.



## 1.5 Ứng dụng của thuật toán

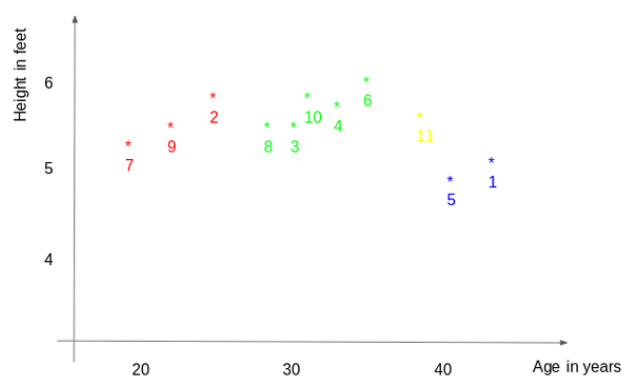
Thuật toán KNN có thể được áp dụng vào cả hai loại bài toán của Supervised learning là Classification (Phân loại) và Regression (Hồi quy). Trong ngành Khoa học Dữ liệu, **KNN** được sử dụng rộng rãi hơn để giải quyết các vấn đề phân loại.

- **KNN Classification:** Dự đoán một lớp bằng cách sử dụng loại chiếm đa số cao nhất trong số k hàng xóm gần nhất của nó. Nhãn của một điểm dữ liệu mới được suy ra trực tiếp từ K điểm dữ liệu gần nhất.
- **KNN Regression:** Dự đoán một giá trị bằng cách sử dụng giá trị trung bình của k hàng xóm gần nhất. Đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất.

*Ví dụ về KNN Regression: Xác định trọng lượng ID = 11 dựa trên biểu đồ?*

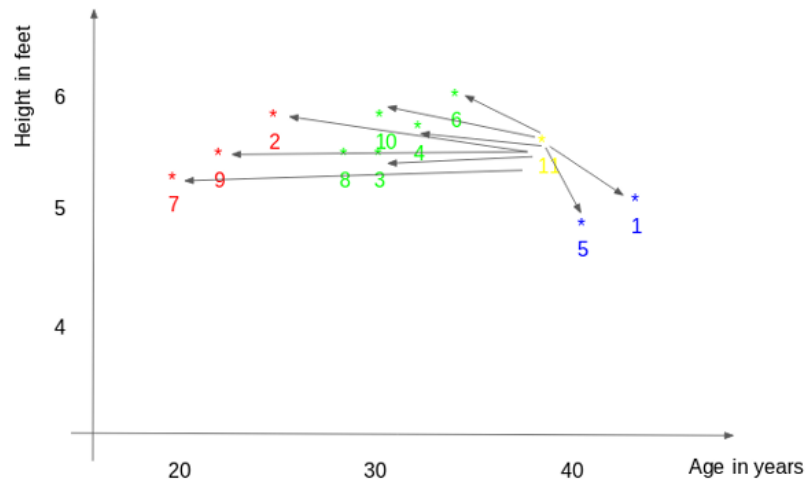
ID	Height	Age	Weight
1	5	45	77
2	5.11	26	47
3	5.6	30	55
4	5.9	34	59
5	4.8	40	72
6	5.8	36	60
7	5.3	19	40
8	5.8	28	60
9	5.5	23	45
10	5.6	32	58
11	5.5	38	?

*Hình 5: Bảng số liệu Height - Age - Weight*



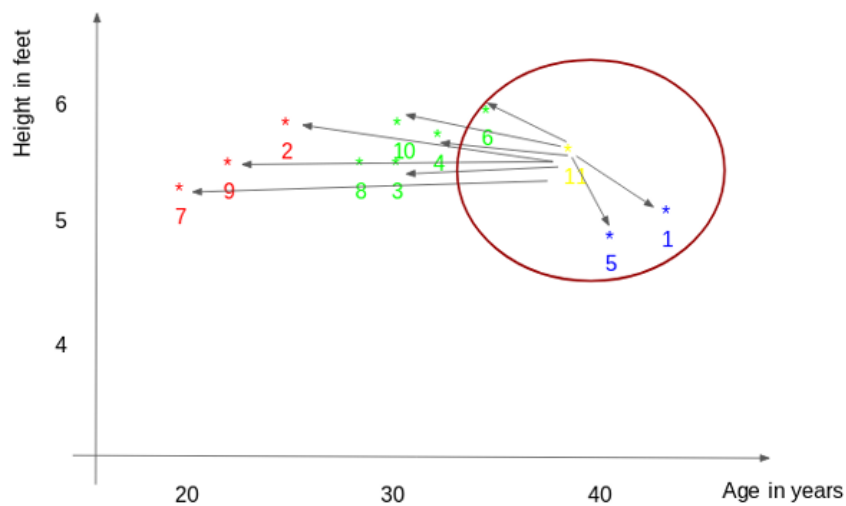
*Hình 6: Biểu thị dữ liệu lên biểu đồ*

Bước 1: Tính khoảng cách giữa điểm ID=11 đến các điểm cho trước



Hình 7: Khoảng cách giữa điểm ID với các điểm cho trước lên

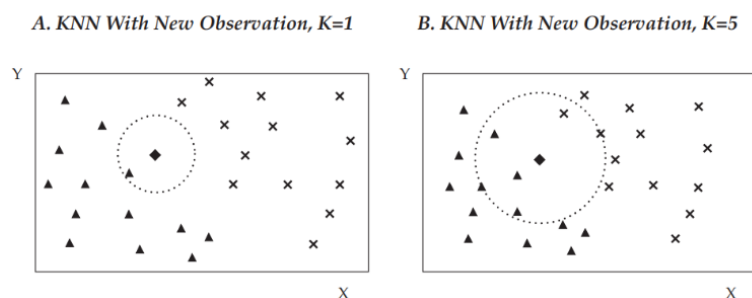
Bước 2: Chọn K điểm dữ liệu gần nhất,  $k=3$  với các điểm 1,5,6.



Hình 8: Chọn hệ số  $k=3$

Bước 3: Giá trị trung bình của các điểm dữ liệu này là dự đoán cuối cùng cho điểm mới. Ở đây ta có cân nặng là  $ID_{11} = (77+72+60)/3 = 69,66$  kg.

Ví dụ về KNN classification: Dự đoán hình là hình chữ thập hay hình tam giác



Hình 9: Chọn hệ số  $k=1, k=5$

Hình thoi trong Hình 1 đang cần được phân loại thuộc hình chữ thập hoặc hình tam giác.

- Nếu  $k = 1$ , hình thoi sẽ được phân loại vào cùng loại với điểm dữ liệu gần nhất của nó (tức là hình tam giác trong bảng bên trái - bảng A).
- Bảng bên phải (bảng B) thể hiện trường hợp  $k = 5$ , thuật toán sẽ xem xét 5 điểm dữ liệu gần hình thoi nhất, đó là 3 hình tam giác và 2 hình chữ thập. Quy tắc quyết định là chọn phân loại có số lượng lớn nhất trong 5 điểm dữ liệu được xem xét. Vì vậy, trong trường hợp này, hình thoi cũng được xếp vào phân loại tam giác.

### 1.5.1. Lĩnh vực nông nghiệp

KNN được áp dụng trong các lĩnh vực liên quan đến nông nghiệp, ví dụ như để mô phỏng lượng mưa hàng ngày và các biến số thời tiết khác, đánh giá kiểm kê rừng và ước tính các biến số rừng, dự báo khí hậu và ước tính các thông số nước trong đất.

### 1.5.2. Lĩnh vực tài chính

Dự báo thị trường chứng khoán là một trong những nhiệm vụ tài chính cốt lõi nhất của KNN. Một số ứng dụng khác của KNN trong tài chính gồm:

- Dự đoán giá cổ phiếu (*trên cơ sở đo lường hiệu quả hoạt động của công ty và dữ liệu kinh tế*). Thuật toán KNN rất hữu ích trong việc ước tính giá trị tương lai của cổ phiếu dựa trên dữ liệu trước đó vì thuật toán này có khả năng dự đoán giá của các thực thể không xác định.
- Điểm tín dụng: Thuật toán KNN so sánh xếp hạng tín dụng của một cá nhân với những người khác có các đặc điểm tương đương để giúp tính toán xếp hạng tín dụng của họ.

- Quản lý phê duyệt khoản vay: Kỹ thuật K hàng xóm gần nhất, tương tự như chấm điểm tín dụng, rất hữu ích trong việc phát hiện những người có nhiều khả năng vỡ nợ hơn bằng cách so sánh thuộc tính của họ với thuộc tính của những người tương tự.
- Dự đoán phá sản
- Quản lý rủi ro tài chính
- Phân tích rửa tiền

### 1.5.3. Lĩnh vực y học

Thuật toán KNN được áp dụng trong lĩnh vực y học, dùng để:

- Dự đoán liệu bệnh nhân nhập viện do nhồi máu cơ tim có bị nhồi máu cơ tim lần thứ hai hay không. Dự đoán dựa trên các phép đo nhân khẩu học, chế độ ăn uống và lâm sàng cho bệnh nhân đó.
- Ước tính lượng glucose trong máu của một người mắc bệnh tiểu đường, từ quang phổ hấp thụ tia hồng ngoại của máu người đó.
- Xác định các yếu tố nguy cơ gây ung thư tuyến tiền liệt, dựa trên các biến lâm sàng và nhân khẩu học.

Ngoài ra, thuật toán KNN cũng đã được áp dụng để phân tích dữ liệu biểu hiện gen vi mảng, trong đó thuật toán KNN đã được kết hợp với các thuật toán di truyền.

### 1.5.4. Thị giác máy tính

Trong lĩnh vực thị giác máy tính, thuật toán KNN được dùng để phân loại hình ảnh. *Nó quan trọng trong nhiều ứng dụng thị giác máy tính vì nó có thể nhóm các điểm dữ liệu có thể so sánh được lại với nhau.*

### 1.5.5. Các ứng dụng khác

**Nhận dạng các mẫu:** Khả năng nhận dạng các mẫu của thuật toán KNN mở ra rất nhiều khả năng. Ví dụ, nó có thể hỗ trợ phát hiện và phát hiện các mẫu đáng ngờ trong việc sử dụng thẻ tín dụng. Tính năng phát hiện mẫu cũng có thể được sử dụng để phát hiện các mẫu trong thói quen mua hàng của khách hàng.

**Tiền xử lý dữ liệu:** Nhiều giá trị bị thiếu có thể được tìm thấy trong bộ dữ liệu. Thiếu dữ liệu bị thiếu là một quy trình sử dụng thuật toán KNN để ước tính các giá trị bị thiếu.

**Hệ thống đề xuất, khuyến nghị:** KNN có thể được sử dụng trong các hệ thống đề xuất vì nó có thể giúp định vị những người có đặc điểm tương tự. Ví dụ: Nó có thể được sử dụng trong một nền tảng truyền phát video trực tuyến để đề xuất nội dung mà người dùng có nhiều khả năng xem hơn dựa trên những gì người dùng khác xem.

## 1.6 Ưu điểm và nhược điểm & vấn đề mở rộng

### 1.6.1 Ưu điểm

- Thuật toán đơn giản, dễ triển khai, độ phức tạp thấp.
- Cho ra kết quả chính xác hơn nếu tập data càng lớn.
- Ít siêu tham số: KNN chỉ yêu cầu giá trị  $k$  và chỉ số khoảng cách  $p$ .
- Có thể thêm các quan sát (observations) mới vào tập dữ liệu bất cứ lúc nào một cách dễ dàng.

### 1.6.2 Nhược điểm

- Giá trị  $k$  cần được tính toán kỹ càng. Giá trị  $k$  thấp sẽ dẫn đến overfitting. Giá trị  $k$  lớn hơn sẽ giảm rủi ro overfitting, nhưng nếu  $k$  quá lớn sẽ dẫn đến việc phân lớp quá rộng và không đạt hiệu quả cao [3]. Không có quy tắc cụ thể nào cho việc lựa chọn giá trị  $k$  tốt nhất. Một số mẹo chọn  $k$  tốt hơn:
  - Chọn  $k = \sqrt{n}$  của  $n$  (Với  $n$  là tổng mẫu huấn luyện). Nên chọn  $k$  là số lẻ để tránh xảy ra trường hợp số lượng 2 lớp bằng nhau [6].
  - Thực hiện thay đổi  $k$  và đánh giá từng mô hình, sau đó chọn  $k$  có kết quả tốt nhất [5].
- Thuật toán này cần nhiều dung lượng và thời gian để thực hiện. KNN là một thuật toán “lười” (lazy algorithm) vì nó không trải qua việc huấn luyện mô hình mà thay vào đó là “ghi nhớ” tập dữ liệu huấn luyện [1]. Chính vì vậy mà KNN sẽ cần nhiều bộ nhớ hơn để lưu trữ dữ liệu huấn luyện. Bên cạnh đó, thời gian xử lý của KNN cũng khá lâu do thuật toán này phải thực hiện tính toán khoảng cách của tất cả giá trị trong tập dữ liệu mỗi lần chạy mô hình.
- KNN chịu ảnh hưởng của “lời nguyền của chiều dữ liệu”: Với những bộ dữ liệu nhiều chiều thì việc sử dụng KNN sẽ không hiệu quả. Khi số lượng tính năng (feature) tăng lên, khoảng cách giữa các điểm dữ liệu trở nên khó phân biệt hơn, tổng diện tích thuật toán cần bao phủ để tìm  $k$  giá trị tăng lên. Từ đó dễ xảy ra

hiện tượng overfitting. Lúc này chúng ta có thể sử dụng các kỹ thuật về giảm chiều để hạn chế ảnh hưởng từ lời nguyên của chiều dữ liệu [4].

- Khắc phục: sử dụng các phương pháp giảm chiều dữ liệu bao gồm cả feature selection và dimensionality reduction tùy theo ngữ cảnh từng bài toán.
- Khó có thể áp dụng KNN lên tập dữ liệu có phân loại (categorical features), loại dữ liệu không phải là số [2]. Nếu giá trị là “Thấp”, “Trung bình”, “Cao” thì việc mã hóa chúng thành các giá trị “1”, “2”, “3” có thể cân nhắc. Tuy nhiên với một bộ giá trị như “Đỏ”, “Vàng”, “Xanh” thì việc mã hóa thành số không còn phù hợp. Chúng ta có thể mã hóa chúng trong không gian với tọa độ  $(1;0;0)$ ,  $(0;1;0)$ ,  $(0;0;1)$  để đảm bảo khoảng cách giữa các giá trị bằng nhau. Tuy nhiên việc tính toán khoảng cách sẽ trở nên phức tạp khi số giá trị tăng lên. Câu hỏi cốt lõi ở đây sẽ là “Khoảng cách trong bộ dữ liệu của bạn được định nghĩa như thế nào?”.

## Chương 2: Ứng dụng vào bài toán

Tóm tắt chương 2: Chương 2 trình bày quá trình áp dụng KNN để huấn luyện mô hình học máy nhằm đưa ra dự đoán quyết định mua sản phẩm của người tiêu dùng dựa trên bộ dữ liệu quảng cáo mạng xã hội. Quá trình này bao gồm các công đoạn: làm sạch bộ dữ liệu, phân tích khai phá bộ dữ liệu, xây dựng mô hình, đánh giá mô hình. Trong quá trình xây dựng mô hình, nhóm thực hiện huấn luyện mô hình trên các điều kiện khác nhau bao gồm: Mô hình KNN với sự thay đổi hệ số  $k$ , Mô hình KNN với lời nguyên của chiều dữ liệu. Ngoài ra còn sử dụng thuật toán Decision Trees để giải bài toán trên nhằm so sánh với thuật toán KNN.

### 2.1 Mô tả bài toán

Để hiểu rõ hơn về ứng dụng của KNN trong thực tế, nhóm đã chọn bài toán sử dụng bộ dữ liệu quảng cáo mạng xã hội để dự đoán về quyết định mua sản phẩm của người tiêu dùng. Dự đoán này sẽ giúp cho doanh nghiệp biết được nhu cầu của khách hàng về sản phẩm và đưa ra những quyết định sản xuất cũng như chiến lược phát triển một cách chính xác và đúng thời điểm.

Tập dữ liệu nhóm sử dụng sẽ chứa thông tin chi tiết về người dùng trong trang mạng xã hội. Ứng dụng phương pháp KNN phân loại dựa trên  $K$  hàng xóm gần nhất, nhóm sẽ lựa chọn các thuộc tính cần thiết và xây dựng mô hình để dự đoán xem người dùng có mua sản phẩm bằng cách nhấp vào quảng cáo trên trang hay không dựa trên mức lương, tuổi và giới tính của họ.

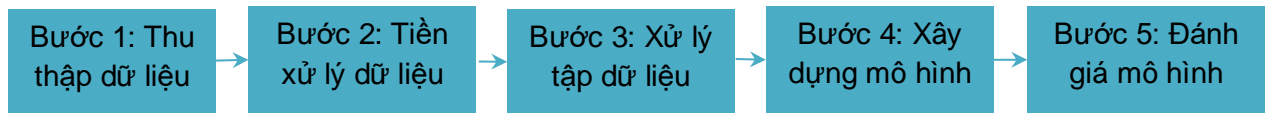
### 2.2 Mô tả dữ liệu

Dữ liệu của bài toán được lấy từ tệp *Social\_Network\_Ads.csv* của tác giả Rakesh Raushan được đăng tải trên trang web Kaggle.com.

Dữ liệu bao gồm 5 trường: User ID, Gender, Age, EstimatedSalary, Purchased. Tổng cộng có 400 dòng dữ liệu được thu thập từ 196 người tiêu dùng nam và 204 người tiêu dùng nữ có độ tuổi từ 18 đến 60, mức thu nhập ước tính từ 15000-150000.

## 2.3 Quy trình đề xuất

Quy trình xử lý bài toán



Bước 1: Thu thập dữ liệu (lựa chọn dữ liệu phù hợp với mô hình)

Bước 2: Tiền xử lý dữ liệu (loại bỏ các giá trị vô nghĩa, outliers và điền vào các giá trị còn thiếu)

Bước 3: Xử lý tập dữ liệu (phân tích và khai phá bộ dữ liệu, đánh giá các biến)

Bước 4: Xây dựng mô hình (chia tập dữ liệu thành 2 tập training và testing, training mô hình)

Bước 5: Đánh giá mô hình (đánh giá mô hình bằng Confusion Matrix)

## 2.4 Phân tích và khai thác dữ liệu (EDA)

Trước khi xây dựng mô hình dự đoán cho bài toán, ta phải tiến hành xem xét và hiểu rõ về bộ dữ liệu trước. Nhóm sẽ tiến hành phân tích và khai thác dữ liệu qua các chỉ số đo, độ phân tán của biến và biểu đồ thể hiện mối tương quan giữa các biến. Qua đó, chúng ta sẽ biết được dữ liệu đã sạch hay chưa và tiến hành làm sạch dữ liệu nếu chưa đáp ứng.

### 2.4.1 Phân tích và khai thác dữ liệu EDA

Kiểm tra xem bộ dữ liệu có những thông tin, giá trị và kiểu dữ liệu nào và kiểm tra xem dữ liệu có giá trị null hay không.

```
social_network_ads.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   User ID               400 non-null   int64
1   Gender                400 non-null   object
2   Age                   400 non-null   int64
3   EstimatedSalary       400 non-null   int64
4   Purchased             400 non-null   int64
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

Hình 10: Kết quả kiểu dữ liệu



#check xem có giá trị bị thiếu không?

```
social_network_ads.isna().sum()
```

```
User ID      0
Gender       0
Age          0
EstimatedSalary  0
Purchased    0
dtype: int64
```

---

Hình 11: Kết quả kiểm tra giá trị null

Bộ dữ liệu gồm 400 dòng, 5 cột (User ID, Gender, Age, EstimatedSalary và Purchased), không có giá trị null và có kiểu dữ liệu là int và object.

Kiểm tra xem dữ liệu có bị duplicate hay không?

#check xem dữ liệu có duplicated hay không?

```
social_network_ads.duplicated().sum()
```


```
0
```

Kết quả là dữ liệu không bị duplicate.

Vì cột User ID không sử dụng đến trong mô hình nên ta sẽ tiến hành xóa cột này.

#Xóa cột User ID

```
social_network_ads.drop('User ID', axis = 1, inplace=True)
```



	Gender	Age	EstimatedSalary	Purchased
0	Male	19	19000	0
1	Male	35	20000	0
2	Female	26	43000	0
3	Female	27	57000	0
4	Male	19	76000	0
...	...	...	...	...
395	Female	46	41000	1
396	Male	51	23000	1
397	Female	50	20000	1
398	Male	36	33000	0
399	Female	49	36000	1

400 rows x 4 columns

Hình 12: Bảng dữ liệu sau khi xóa cột User ID

Tiếp đến, ta sẽ mô tả tổng quan về bộ dữ liệu thông qua các giá trị như giá trị trung bình, độ lệch chuẩn, min, max,... của các biến trong bộ dữ liệu.

```
social_network_ads.describe()
```

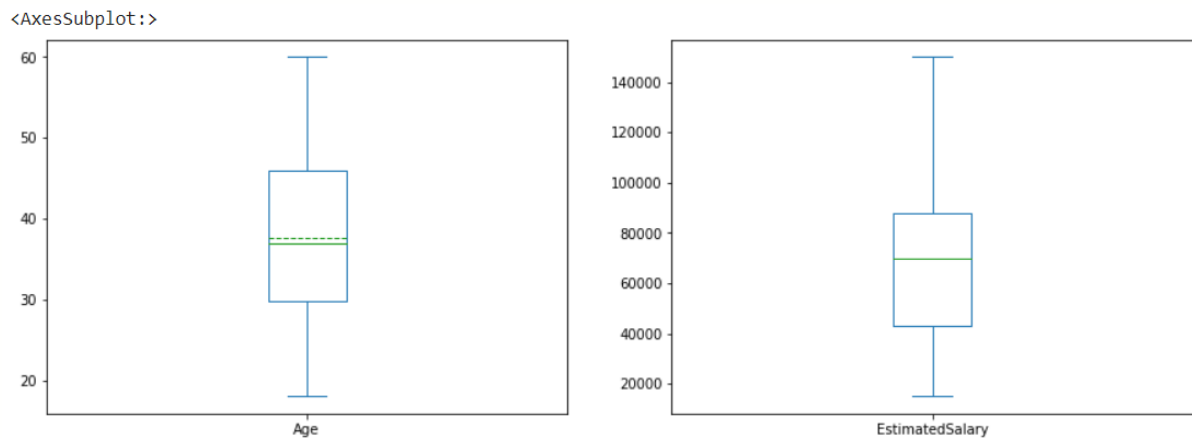
	Age	EstimatedSalary	Purchased
<b>count</b>	400.000000	400.000000	400.000000
<b>mean</b>	37.655000	69742.500000	0.357500
<b>std</b>	10.482877	34096.960282	0.479864
<b>min</b>	18.000000	15000.000000	0.000000
<b>25%</b>	29.750000	43000.000000	0.000000
<b>50%</b>	37.000000	70000.000000	0.000000
<b>75%</b>	46.000000	88000.000000	1.000000
<b>max</b>	60.000000	150000.000000	1.000000

Hình 13: Kiểm tra giá trị trung bình, độ lệch chuẩn, min, max.

Kiểm tra xem bộ dữ liệu có các giá trị ngoại lệ hay không, nếu có phải tiến hành xử lý vì nó mang lại kết quả không tốt cho mô hình. Và để quan sát các biến một cách trực quan nhất, chúng ta sử dụng biểu đồ Box Plots và Histogram.

### #Kiểm tra giá trị ngoại lệ

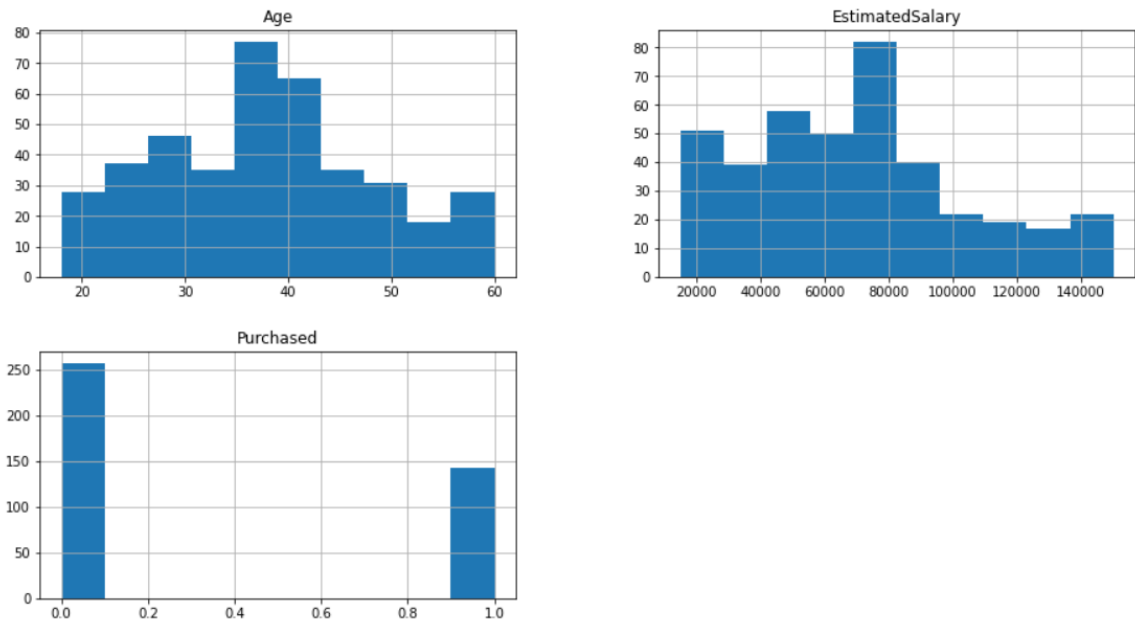
```
fig, ax = plt.subplots(1,2, figsize = (15, 5)) #1 dòng, 2 cột  
social_network_ads['Age'].plot.box(ax = ax[0], showmeans = True, meanline  
= True)  
social_network_ads['EstimatedSalary'].plot.box(ax = ax[1], showmeans =  
True, meanline = True)
```



Hình 14. Biểu đồ của biến Age và EstimatedSalary

Dựa trên biểu đồ, chúng ta không nhìn thấy điểm nằm ngoài cực đại và cực tiểu của biến, vậy nên bộ dữ liệu không có giá trị ngoại lệ.

```
social_network_ads.hist(figsize = (15, 8))
```



Hình 15. Biểu đồ của biến Age, EstimatedSalary và Purchased

Biểu đồ Histogram thể hiện:

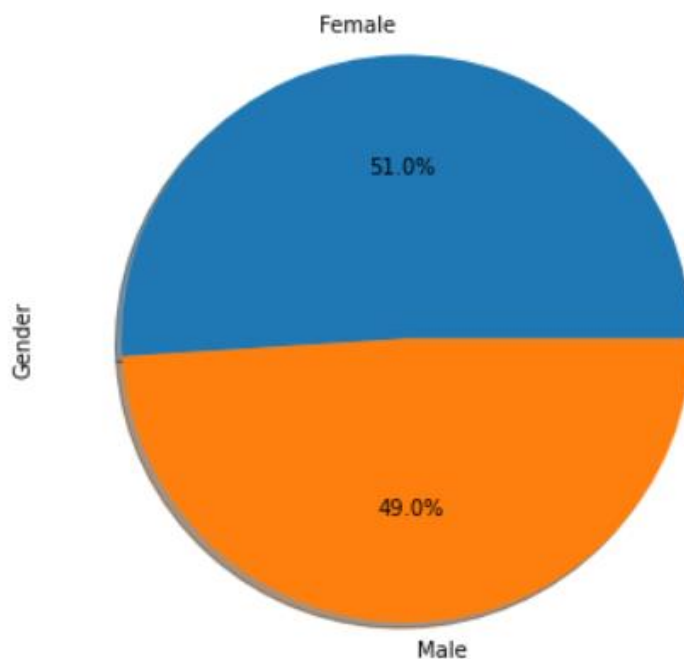
- Biến tuổi phân bố từ 18 đến 60 tuổi
- Biến tiền lương phân bố từ 15000 đến 150000
- Biến mục tiêu có giá trị min là 0 và giá trị max là 1

Đếm giá trị của cột giới tính, để biết số lượng nam và nữ là bao nhiêu. Sau đó sử dụng biểu đồ hình tròn để thể hiện sự phân bố.

#Đếm giá trị của cột Gender

```
social_network_ads['Gender'].value_counts()
```

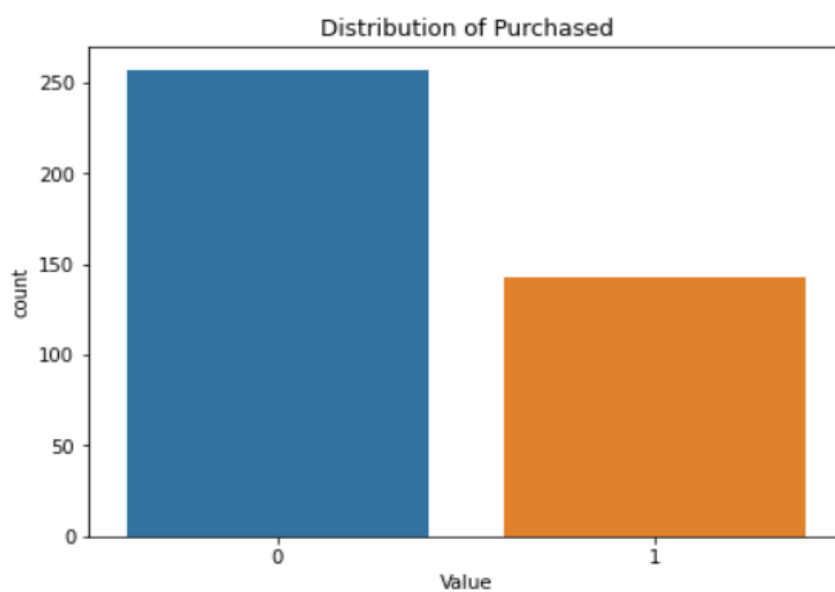
```
Female    204
Male      196
Name: Gender, dtype: int64
```



Hình 16. Biểu đồ hình tròn của biến Gender

Sử dụng biểu đồ Countplot để thể hiện dữ liệu của biến mục tiêu Purchased:

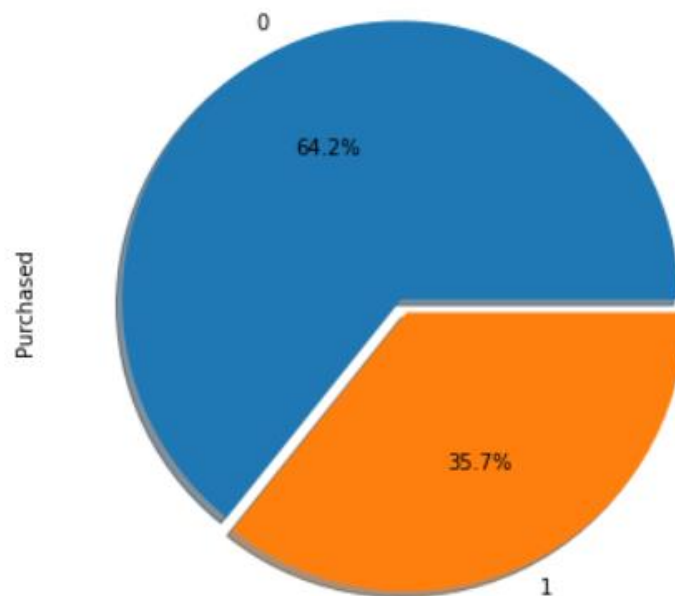
```
fig, ax = plt.subplots(figsize = (7, 5))
sns.countplot('Purchased', data = social_network_ads)
ax.set_title('Distribution of Purchased')
ax.set_xlabel('Value')
```



Hình 17. Biểu đồ thanh của biến mục tiêu Purchased

Ta cũng có thể biểu hiện sự phân bố của biến mục tiêu ở biểu đồ hình tròn:

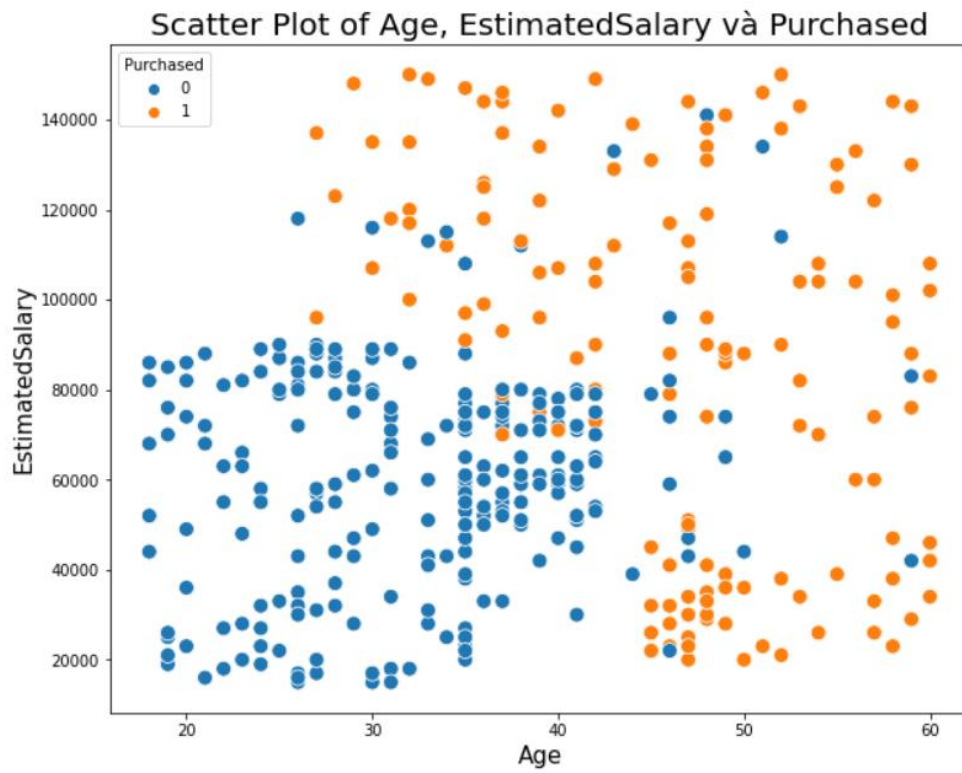
```
social_network_ads['Purchased'].value_counts().plot.pie(autopct = '%1.1f%%', shadow=True, figsize = (6, 8), explode = [0, 0.05])
```



*Hình 18. Biểu đồ hình tròn của biến mục tiêu Purchased*

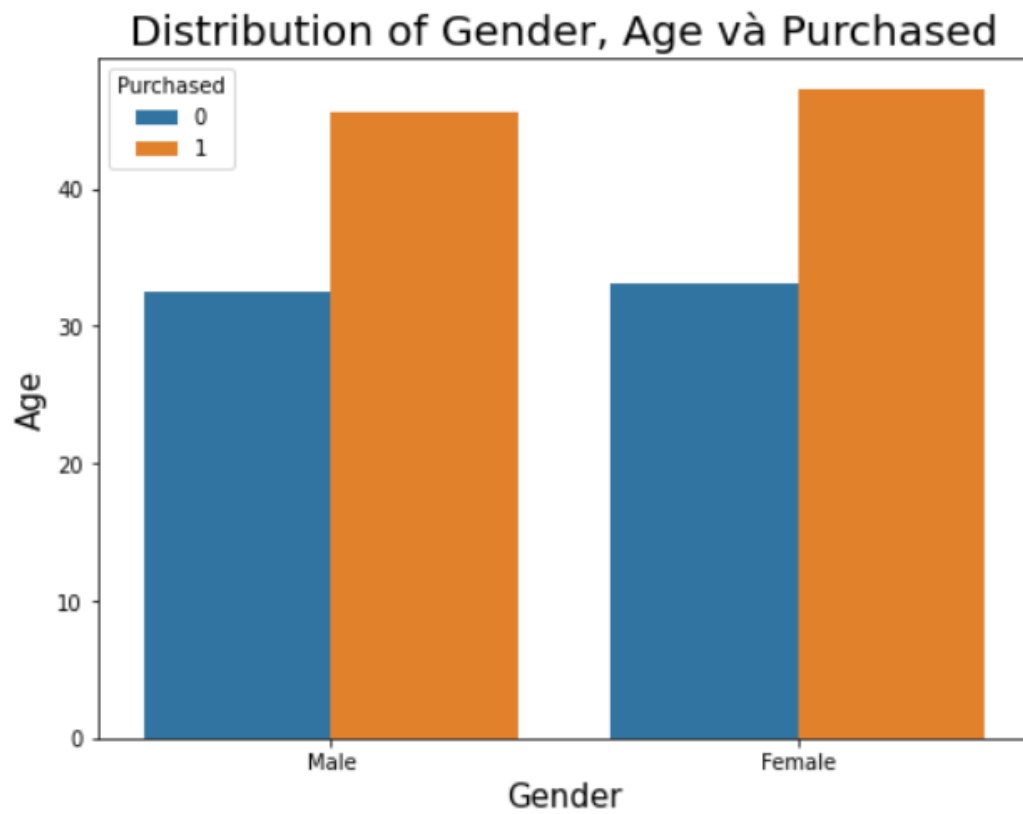
Phân tích mối tương quan giữa các biến bằng cách sử dụng đồ thị phân tán (Scatter Plot)

```
plt.figure(figsize=(10,8))  
plt.title('Đồ thị phân tán giữa Age, Estimated và Purchased', fontsize=20)  
plt.xlabel('Age', fontsize=15)  
plt.ylabel('EstimatedSalary', fontsize=15)  
sns.scatterplot(data=social_network_ads, x='Age', y='EstimatedSalary',  
hue='Purchased', s=100);
```



*Hình 19. Sơ đồ phân tán của biến Age, EstimatedSalary và Purchased*

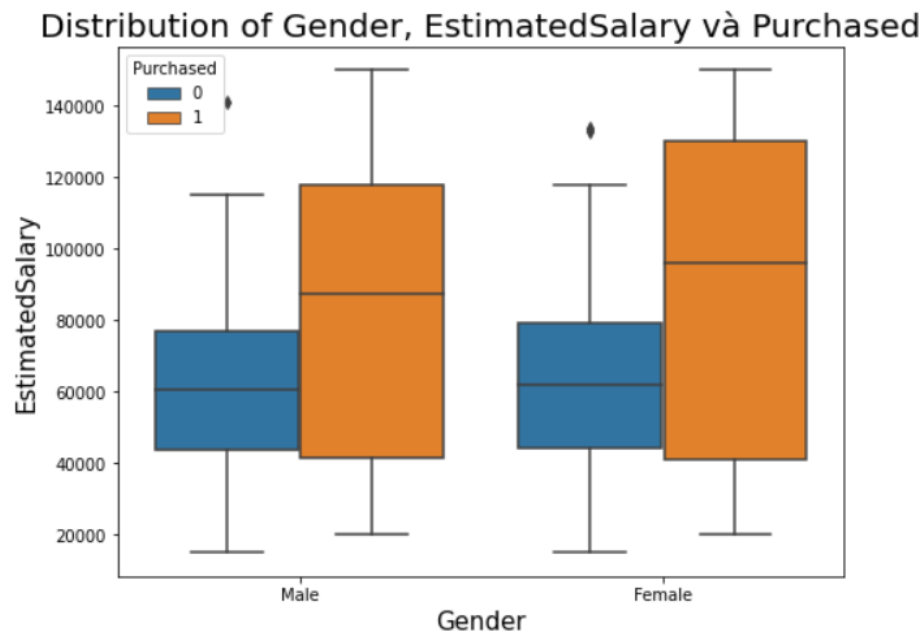
Sử dụng biểu đồ thanh (BarPlot) để thể hiện sự phân bố của biến Gender, Age và Purchased.



*Hình 20. Biểu đồ của biến Gender, Age và Purchased*

Tiếp đến là sự phân bố của biến Gender, EstimatedSalary và Purchased.





Hình 21. Biểu đồ của biến Gender, EstimatedSalary và Purchased

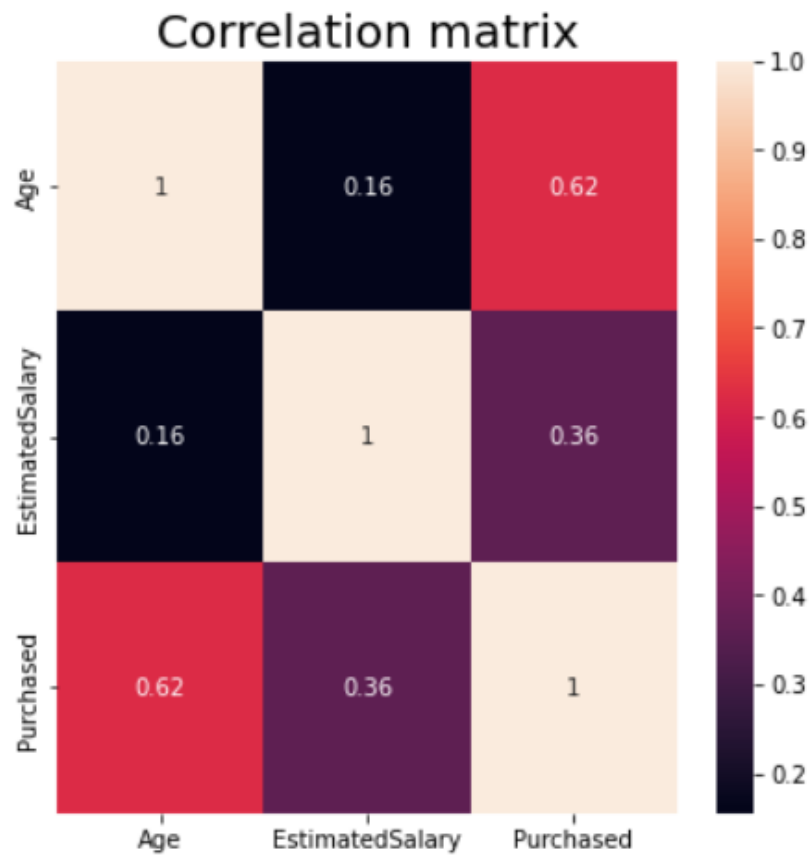
Dựa trên 2 biểu đồ, ta nhận thấy việc sử dụng tính năng giới tính không cần thiết vì biến Gender ảnh hưởng khá ít đến biến mục tiêu Purchased. Vậy nên ta sẽ xóa đi cột Gender.

```
social_network_ads.drop('Gender', inplace=True, axis = 1)
```

Sử dụng bản đồ nhiệt heatmap để thể hiện mối tương quan giữa các biến trên ma trận tương quan dưới đây:

*#ma trận tương quan giữa các biến*

```
plt.figure(figsize=(6, 6))
sns.heatmap(social_network_ads.corr(), annot=True)
plt.title('Correlation matrix', fontsize=20);
```



Hình 22. Ma trận tương quan của biến Age, EstimatedSalary và Purchased

Ta chia bộ dữ liệu thành 2 phần:

- Phần cắt đi biến phụ thuộc Purchased (X)
- Phần chứa biến phụ thuộc Purchased (Y), đây là mục tiêu bài toán

```
X = social_network_ads.drop('Purchased', axis = 1)
y = social_network_ads['Purchased']
```

## 2.4.2 Tiền xử lý dữ liệu

### 2.4.2.1. Chia biến

Ta chia các biến thành biến độc lập và biến phụ thuộc.

```
#independent variables
X = social_network_ads.iloc[:, :-1].values

array([[ 'Male', 19, 19000],
       [ 'Male', 35, 20000],
       [ 'Female', 26, 43000],
       ...,
       [ 'Female', 50, 20000],
       [ 'Male', 36, 33000],
       [ 'Female', 49, 36000]], dtype=object)
```

*Hình 23. Biến độc lập được phân chia ra*

```
#dependent variable
Y = social_network_ads.iloc[:, -1].values

array([0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0,
       1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1])
```

*Hình 24. Biến phụ thuộc của bài toán*

### 2.4.2.2. LabelEncoder

LabelEncoder (Mã hóa nhãn dữ liệu): Giúp chuyển đổi từ kiểu chuỗi (string) sang kiểu số (integer). Áp dụng cho những bài toán yêu cầu dữ liệu phải ở dạng số.

Ví dụ: Ba giá trị đầu ra kiểu string (Iris-setosa, Iris-versicolor, Iris-virginica) sẽ được ánh xạ thành các giá trị số tương ứng (0, 1, 2)

Trong bài toán này, ta sẽ sử dụng LabelEncoder để mã hóa cột Giới tính thành 0 (Nữ) và 1 (Nam)

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
X[:,0] = le.fit_transform(X[:,0])

array([1, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0,
       1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1,
       1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0,
       1, 0, 1, 0], dtype=object)
```

Hình 25. Sử dụng LabelEncoder cho cột Gender

#### 2.4.2.3. Chia tập dữ liệu

Để bắt đầu việc xây dựng các mô hình, ta tiến hành chia tập dữ liệu đã chuẩn hóa thành 2 phần theo tỉ lệ 8:2

- Train (dùng để tìm các hệ số và xây dựng mô hình)
- Test (dùng để đưa ra các giá trị predict và đánh giá mô hình)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.20, random_state = 0)
```

#### 2.4.2.4. StandardScaler

StandardScaler: Hàm StandardScaler() được dùng để quy đổi tỷ lệ của các giá trị khác nhau để so sánh. Trong thực tế dữ liệu có các giá trị khác nhau và đơn vị đo lường khác nhau mà chúng khó có thể so sánh được.

Ví dụ: Làm thế nào để so sánh các giá trị là kg và m hoặc độ cao và thời gian... Để giải quyết cho bài toán này chúng ta có thể sử dụng tính năng mở rộng Scale để quy đổi sang một giá trị mới để dễ dàng so sánh hơn.

Trong bài toán này, StandardScaler sẽ chia tỷ lệ cho tập huấn luyện và kiểm tra các biến độc lập để giảm kích thước xuống các giá trị nhỏ hơn phù hợp với việc chạy mô hình.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
[ 1.02532046e+00,  7.59458956e-02,  1.04507694e+00],
[-9.75304830e-01, -1.18475597e-01, -3.74554562e-01],
[-9.75304830e-01, -1.18779381e+00,  6.00265106e-02],
[-9.75304830e-01, -3.12897090e-01, -1.35960499e+00],
[-9.75304830e-01,  1.53410709e+00,  1.10302108e+00],
[ 1.02532046e+00, -7.98950822e-01, -1.53343742e+00],
[ 1.02532046e+00,  7.59458956e-02,  1.85629494e+00],
[ 1.02532046e+00, -8.96161568e-01, -7.80163563e-01],
[ 1.02532046e+00, -5.07318583e-01, -7.80163563e-01],
[ 1.02532046e+00, -3.12897090e-01, -9.25023920e-01],
[ 1.02532046e+00,  2.70367388e-01, -7.22219420e-01],
[-9.75304830e-01,  2.70367388e-01,  6.00265106e-02],
[-9.75304830e-01,  7.59458956e-02,  1.85629494e+00],
[-9.75304830e-01, -1.09058306e+00,  1.94321116e+00],
[-9.75304830e-01, -1.67384754e+00, -1.56240949e+00],
[ 1.02532046e+00, -1.18779381e+00, -1.09885635e+00],
[ 1.02532046e+00, -7.01740076e-01, -1.13805918e-01],
[-9.75304830e-01,  7.59458956e-02,  8.89985821e-02],
[ 1.02532046e+00,  2.70367388e-01,  2.62831011e-01],
[-9.75304830e-01,  8.53631867e-01, -5.77359062e-01],
[-9.75304830e-01,  2.70367388e-01, -1.15680049e+00],
[-9.75304830e-01, -1.18475597e-01,  6.68440012e-01],
[-9.75304830e-01,  2.11737157e+00, -6.93247348e-01],
[ 1.02532046e+00, -1.28500455e+00, -1.38857706e+00],
```

Hình 26. Sử dụng StandardScaler cho tập dữ liệu

## 2.5 Xây dựng mô hình & đánh giá

### 2.5.1 Mô hình KNN với sự thay đổi hệ số k

#### 2.5.1.1 Vấn đề Overfitting và Underfitting trong KNN

Trong KNN, việc lựa chọn k rất quan trọng, nó ảnh hưởng trực tiếp đến chất lượng của mô hình và gây ra 2 trường hợp phổ biến thường gặp trong các mô hình học máy: **Overfitting và Underfitting**

- $k = 1$ : Mô hình quá cụ thể và không khái quát hóa tốt. Nó cũng có xu hướng nhạy cảm với các dữ liệu bất thường (nhiều) trong tập dữ liệu. Mô hình đạt được độ chính xác cao trên dữ liệu đào tạo nhưng sẽ là một công cụ dự đoán kém đối với các điểm dữ liệu mới chưa từng gặp trước đây. Do đó, chúng ta có thể sẽ tạo ra một mô hình quá khớp (overfit).
- $k = n$ : Ngược lại với trường hợp trên với  $k=n$  ( $n$  là tổng mẫu huấn luyện), mô hình quá tổng quát và không phải là một công cụ dự báo tốt trên cả tập dữ liệu đào tạo và tập dữ liệu kiểm tra. Tình trạng này được gọi là mô hình chưa khớp (underfit).

Vì để tránh rơi vào trường hợp Overfitting và Underfitting, chúng ta cần phải chọn  $k$  làm sao để tối ưu nhất.

#### 2.5.1.2 Cách chọn $k$ tối ưu

Theo nghiên cứu của nhóm, không có một phương thức tường minh nào được áp dụng chính thức để xác định  $k$  tối ưu. Tuy nhiên, sau những tìm hiểu của nhóm thì nhóm đã tìm ra được một số cách xác định  $k$  tối ưu như sau:

- Phương pháp 1: Chọn  $k = \sqrt{n}$  (Với  $n$  là tổng mẫu huấn luyện). Nên chọn  $k$  là số lẻ để tránh xảy ra trường hợp số lượng 2 lớp bằng nhau.
- Phương pháp 2: Thực hiện thao tác thay đổi  $k$  và đánh giá từng mô hình, sau đó chọn  $k$  có biểu hiện tốt nhất.
- Phương pháp 3: Sử dụng GridSearchCV để xác định  $k$  tối ưu được tham khảo tại <https://datasciencebasic.com/?p=134>

#### 2.5.1.3 Thực hiện tìm $k$ tối ưu

##### Sử dụng Phương pháp 1:

Ta có tổng số mẫu huấn luyện là  $0.8 \times 400 = 320 \Rightarrow k = \sqrt{320} \approx 17.9$

Vì nên chọn  $k$  là số lẻ nên ta chọn  $k=17$ . Hãy thử kiểm chứng với  $k=17$  kết quả mô hình sẽ như thế nào. Điều này sẽ sáng tỏ sau khi có kết quả của phương pháp 3.

##### Sử dụng Phương pháp 2:

```
#Đầu tiên mình phải nạp hàm GridSearchCV
from sklearn.model_selection import GridSearchCV
#Tiến hành huấn luyện mô hình bằng hàm GridSearchCV
```

```

knn_grid = GridSearchCV(estimator = KNeighborsClassifier(),
param_grid={'n_neighbors': np.arange(1,50)}, cv=5)
knn_grid.fit(X_train,y_train)
#Tìm ra k tối ưu
knn_grid.best_params_
#Kết quả
{'n_neighbors': 9}

#Tiến hành đánh giá mô hình
y_pred=knn_grid.predict(X_test)
ac = accuracy_score(y_test,y_pred)
ac
#Kết quả đánh giá
0.95

```

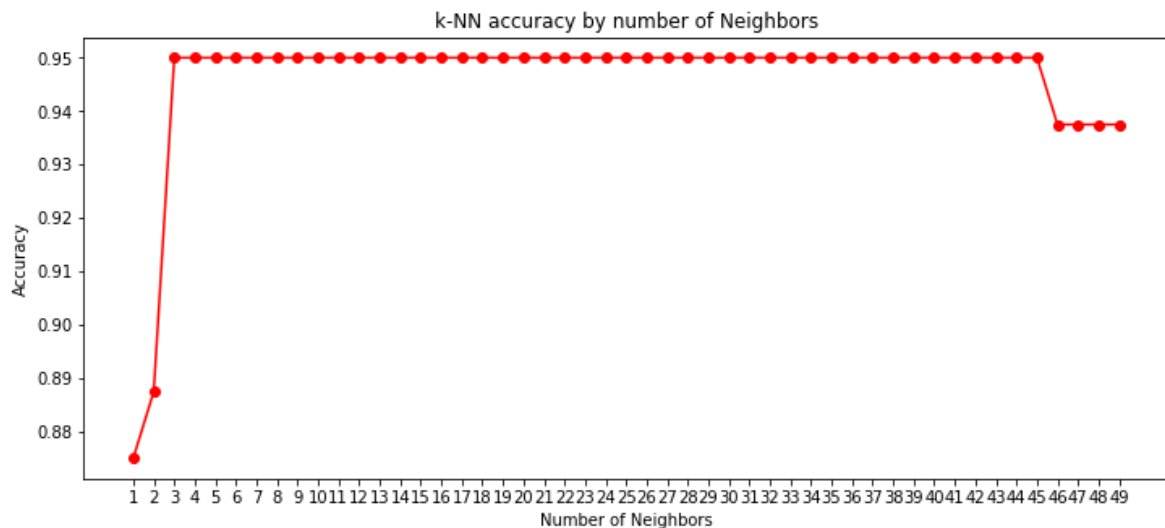
### Sử dụng phương pháp 3

```

#khởi tạo chuỗi
scoreListknn = []
#Khởi tạo huấn luyện mô hình knn với vòng lặp thay đổi k từ
1->50
for k in range(1,50):
    KNclassifier = KNeighborsClassifier(n_neighbors = k,
metric = 'minkowski', p = 2)
    KNclassifier.fit(X_train, y_train)
#Thêm phần tử đánh giá mô hình trên dữ liệu test
    scoreListknn.append(KNclassifier.score(X_test, y_test))
#Trực quan hóa kết quả đánh giá mô hình khi thay đổi K
plt.figure(figsize=(15,5))
plt.plot(range(1,50), scoreListknn,c='red',marker='o')
plt.xticks(np.arange(1,50,1))
plt.xlabel("K value")
plt.ylabel("Score")
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.title('k-NN accuracy by number of Neighbors')
plt.show()
#in giá trị accuracy tốt nhất
KNAcc = max(scoreListknn)

```

```
print("KNN best accuracy: {:.2f}%".format(KNAcc*100))
#Kết quả:
```



Hình 27. Độ chính xác của KNN theo k

KNN best accuracy: 95.00%

### Kết luận:

- Theo biểu đồ kết quả của phương pháp 3, ta thấy chọn K thuộc đoạn [3;45] là tối ưu với mô hình này ( khảo sát trong đoạn [1;50]). Vậy đồng thời chứng minh phương pháp 1 và 2 đều đúng khi chọn được k tối ưu với mô hình này (k được chọn lần lượt là 17 và 9)
- Nhìn chung, phương pháp 3 là tường minh, dễ nắm bắt nhất, song lại phức tạp, tiêu tốn nhiều thời gian, tài nguyên nếu gặp phải tập dữ liệu lớn.
- Để tiếp tục thực hiện nâng cao hiệu quả mô hình, nhóm chọn k=9 làm k tối ưu để thực hiện khảo sát với lời nguyên của chiều dữ liệu

### 2.5.2 Mô hình KNN với lời nguyên của chiều dữ liệu

Chúng ta thực hiện thay đổi chiều dữ liệu bằng phương pháp PCA (Principle Component Analysis) và thông số Accuracy để đánh giá sự ảnh hưởng của chiều dữ liệu lên hiệu quả của thuật toán KNN trong bài toán đã được đưa ra. Chúng ta có thể tiến hành chạy mô hình với số chiều dữ liệu trong khoảng  $0 < n\_features \leq 2$ .

a) Mô hình KNN với số chiều dữ liệu tự nhiên ( $n\_features = 2$ )

Biến độc lập X đưa vào với số feature là 2 (Age, EstimatedSalary)



Chạy mô hình KNN trên tập dữ liệu train:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors = 9, metric =
'minkowski', p = 2)
classifier.fit(X_train, y_train)

y_pred = classifier.predict(X_test)

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
```

Kết quả đo lường: mô hình có độ chính xác (accuracy) là 0.95

b) Với số chiều dữ liệu  $n_{\text{features}} = 1$

```
#Áp dụng PCA với số chiều = 1
pca = PCA(n_components = 1)

X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_

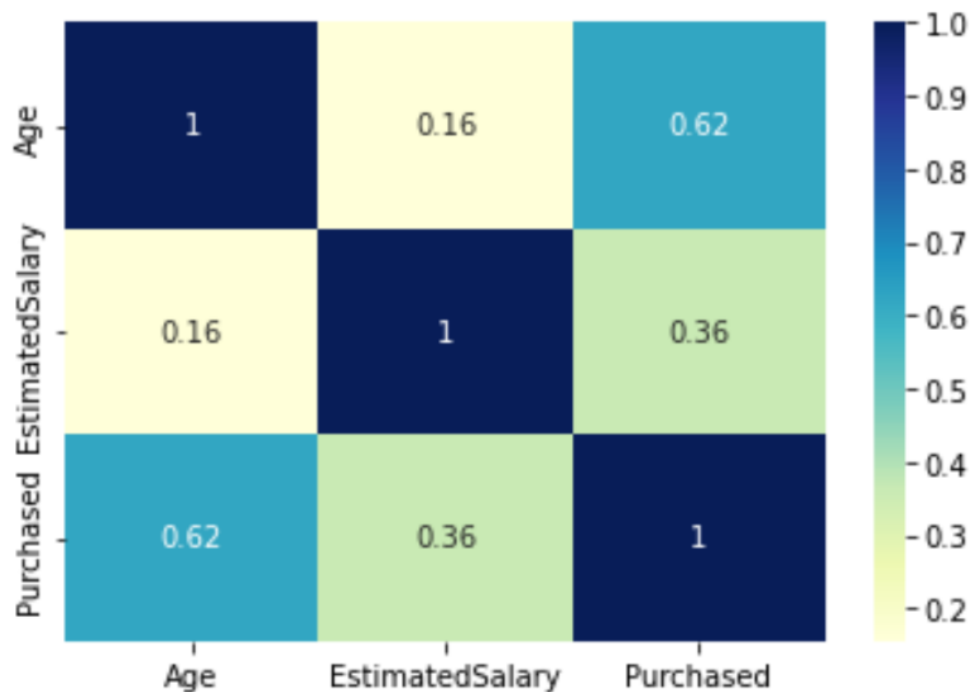
#Tiến hành chạy thử mô hình
classifier = KNeighborsClassifier(n_neighbors = 9, metric =
'minkowski', p = 2)
classifier.fit(X_train_pca, y_train)

#Áp dụng mô hình vào tập test
y_pred = classifier.predict(X_test_pca)

#Đo lường mô hình
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac = accuracy_score(y_test, y_pred)
print("KNN's Accuracy: ", ac)
```

Kết quả đo lường: mô hình có độ chính xác (accuracy) là 0.875

Kết luận:  $n\_features = 2$  cho kết quả mô hình tốt nhất. Điều này có thể giải thích bởi số chiều dữ liệu tự nhiên của tập dữ liệu khá nhỏ. Các thuộc tính trong biến X (Age, Estimated) đều có sự ảnh hưởng khá lớn đến biến Y (với hệ số tương quan lần lượt là 0.62, 0.36). Vì vậy khi giảm chiều dữ liệu, độ chính xác của mô hình sẽ giảm theo.



Hình 28. Ma trận tương quan giữa Age, Estimated Salary và Purchased

### 2.5.3 Mô hình khác có thể áp dụng vào bài toán - Thuật toán Decision Trees

Ngoài việc dự đoán bằng thuật toán KNN thì bài toán trên cũng có thể phân loại bằng những thuật toán khác. Và để so sánh xem KNN có phải là thuật toán tối ưu trong việc phân loại hay chưa thì nhóm chúng em đã quyết định chạy thêm một mô hình khác. Mô hình mà tụi em chọn là Decision Trees.

Sau khi đã xử lý dữ liệu, nhóm chúng em sẽ tiến hành set up môi trường để xây dựng mô hình Decision Tree

- Import những thư viện cần thiết

```
import numpy as np
```

```
import pandas as pd
import sklearn
from sklearn.tree import DecisionTreeClassifier
```

- **Setting up the Decision Tree**

Bây giờ `train_test_split` sẽ trả về 4 tham số khác nhau: `X_trainset`, `X_testset`, `y_trainset`, `y_testset`.

`Train_test_split` sẽ cần các tham số: `X`, `y`, `test_size=0.3` và `random_state=3`.

`X` và `y` là các mảng được yêu cầu trước khi phân tách, `test_size` biểu thị tỷ lệ của tập dữ liệu thử nghiệm và `random_state` đảm bảo rằng có được các phân tách giống nhau.

```
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X,
y, test_size=0.2, random_state=3)
```

- **Huấn luyện**

In hình của `X_trainset` và `y_trainset`

```
print(X_trainset.shape)
print(y_trainset.shape)
```

Kết quả:

```
(320, 2)
(320,)
```

```
print(X_testset.shape)
print(y_testset.shape)
```

Kết quả:

```
(80, 2)
(80,)
```

- **Modeling**

Đầu tiên chúng ta sẽ tạo một phiên bản của `DecisionTreeClassifier` có tên là `purchasedTree`. Bên trong bộ phân loại, chỉ định tiêu chí = "entropy" để chúng ta có thể thấy mức tăng thông tin của từng node

```
purchasedTree = DecisionTreeClassifier(criterion="entropy",
max_depth = 4)
```

Tiếp theo, tiến hành làm khớp dữ liệu với training feature matrix `X_trainset` và training response vector `y_trainset`

```
purchasedTree.fit(X_trainset,y_trainset)
```

- **Prediction**

Đưa ra một số dự đoán trên tập dữ liệu thử nghiệm và lưu trữ nó vào một biến có tên là `predTree`.

```
predTree = purchasedTree.predict(X_testset)
```

- **Evaluation**

```
from sklearn import metrics
import matplotlib.pyplot as plt
print("DecisionTrees's Accuracy: ",
metrics.accuracy_score(y_testset, predTree))
```

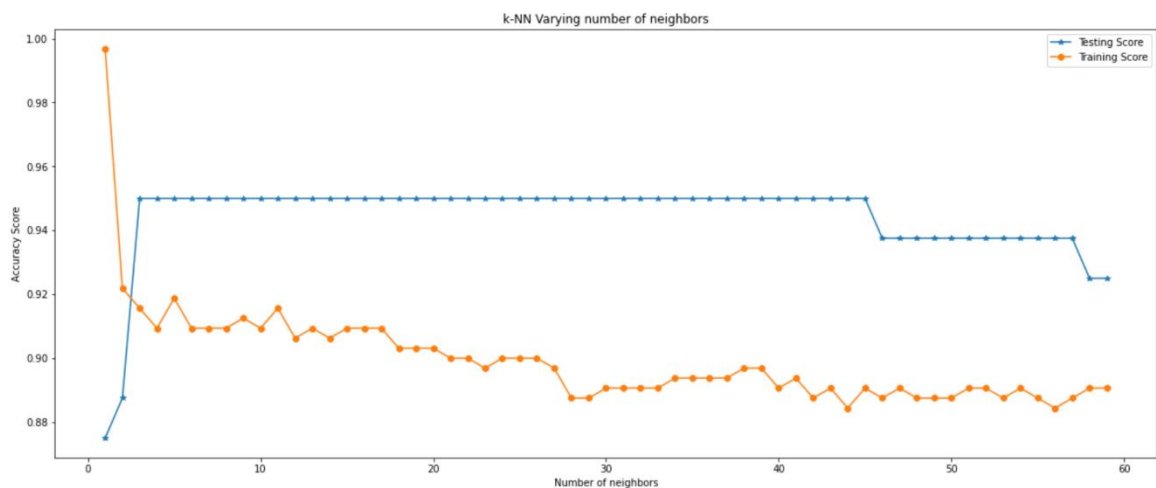
Kết quả:

```
DecisionTrees's Accuracy:  0.925
```

Với độ chính xác là 0.925 thì đây là một mô hình khá tốt.

## 2.6 Đánh giá và Kết luận

Đầu tiên, nhóm tiến hành đánh giá mối tương quan của mô hình trong trường hợp số lượng láng giềng `K` thay đổi, nhóm đã sử dụng tập dữ liệu đã chia thành 2 tập: một tập dữ liệu tập huấn và một tập dữ liệu kiểm thử ở trên để thực hiện tính độ chính xác của các tập và cho ra một đồ thị biểu diễn, so sánh độ chính xác của tập dữ liệu kiểm thử so với các dữ liệu `n_neighbors` trên trục `y` và trục `x` như sau.



Hình 29. Đồ thị KNN thay đổi hệ số  $k$

Từ đồ thị này, nhóm có thể nhận ra được một số đặc điểm của overfitting và underfitting trong mô hình. Khi xem xét ít hơn láng giềng sẽ tương ứng với một mô hình phức tạp hơn, biểu đồ sẽ được trải dài theo chiều ngang liên quan đến hình minh họa trong model\_complexity.

Xem xét một láng giềng, dự đoán trên dữ liệu tập huấn là tốt nhất. Xem xét thêm nhiều láng giềng, mô hình đơn giản hơn, và độ chính xác của tập huấn giảm xuống.

Độ chính xác của tập dữ liệu kiểm thử khi sử dụng một láng giềng sẽ thấp hơn khi sử dụng nhiều láng giềng, việc sử dụng một láng giềng sẽ dẫn đến một mô hình quá phức tạp. Có thể thấy khi chỉ có từ 1-2 láng giềng, kết quả độ chính xác của tập dữ liệu kiểm thử thấp hơn rất nhiều so với tập huấn luyện, điều này dẫn đến tình trạng overfitting của mô hình. Khi xem xét 60 láng giềng, mô hình sẽ rất đơn giản nhưng cũng làm giảm hiệu suất và tài nguyên khi chạy mô hình, ngoài ra có thể thấy một số dấu hiệu của tình trạng underfitting của mô hình khi kết quả trên tập huấn không được cao và kết quả trên tập kiểm thử cũng có tình trạng giảm xuống thấp hơn bình thường. Hiệu suất tốt nhất trên tập kiểm thử sẽ nằm đâu đó là ở giữa với  $K$  láng giềng được sử dụng là từ 3 đến 45 còn trên tập huấn luyện thì là 1. Tuy nhiên, hiệu suất thấp nhất trên tập kiểm thử chính xác khoảng 87.5% khi  $K$  láng giềng bằng 1 còn trên tập huấn luyện thì là 44 và 56.

Tổng kết lại, sử dụng 1 láng giềng, mỗi điểm trong tập dữ liệu tập huấn đều có sự ảnh hưởng rõ rệt trong những dự đoán, và những giá trị đã dự đoán vượt ra ngoài tất

cả của các điểm dữ liệu. Điều này sẽ dẫn đến một dự đoán không chắc chắn. Trong khi xét nhiều láng giềng sẽ đem lại những dự đoán đều đặn, nhưng cũng không phù hợp với tập dữ liệu tập huấn đồng thời cho kết quả trên tập kiểm giảm xuống. Xem xét K ở khoảng giữa từ 3 đến 45 sẽ cho ra hiệu suất tốt nhất trên tập kiểm và kết quả khá ổn trên tập huấn.

Tiếp tục, nhóm tiến hành thực hiện các phép đo lường thuật toán lần lượt là (Accuracy, ROC AUC, Recall, F1 score, Precision, Confusion Matrix,...) trên cả 3 mô hình (KNN với K tối ưu, KNN với lời nguyên dữ liệu và Decision Tree).

### 2.6.1 Mô hình KNN với K tối ưu (n\_neighbors = 9)

```
from sklearn.metrics import accuracy_score, roc_auc_score,
recall_score, f1_score, precision_score
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train,y_train)
y_pred = knn.predict(X_test)
acc = accuracy_score(y_test, y_pred)
roc = roc_auc_score(y_test, y_pred)
rec = recall_score(y_test, y_pred)
f1s = f1_score(y_test, y_pred)
pre = precision_score(y_test, y_pred)
print('Accuracy Score: ' + str(acc))
print('ROC AUC Score: ' + str(roc))
print('Recall Score: ' + str(rec))
print('F1 Score: ' + str(f1s))
print('Precision Score: ' + str(pre))
```

**Accuracy Score: 0.95**

**ROC AUC Score: 0.95141065830721**

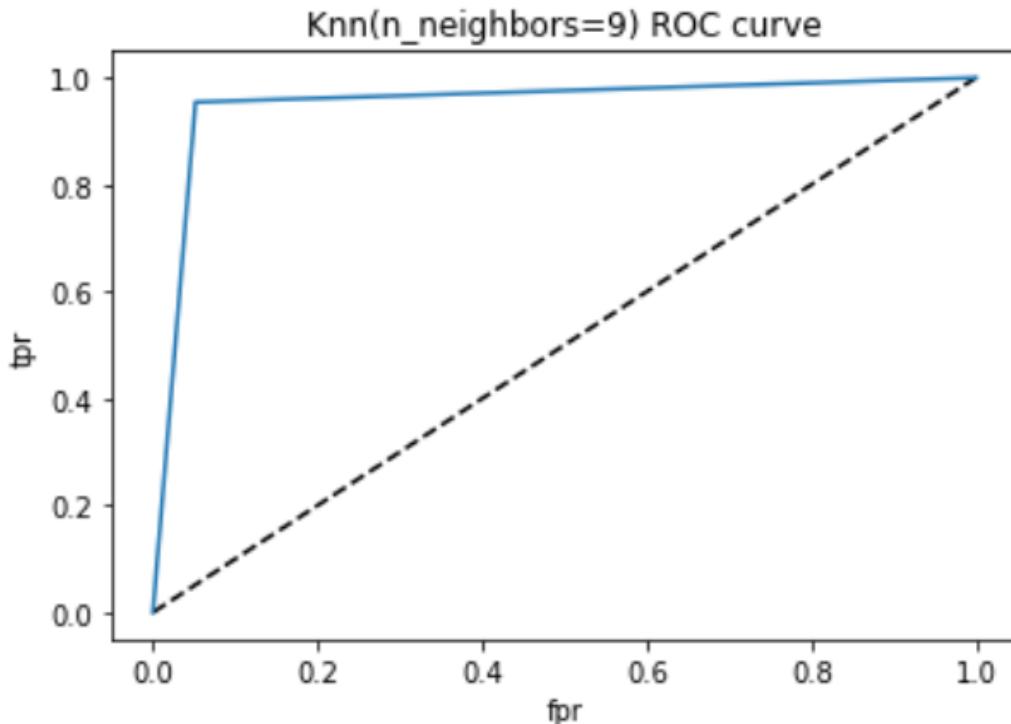
**Recall Score: 0.9545454545454546**

**F1 Score: 0.9130434782608695**

**Precision Score: 0.875**

```
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
plt.plot([0,1],[0,1], 'k--')
```

```
plt.plot(fpr, tpr, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=9) ROC curve')
plt.show()
```



*Hình 30. Đồ thị ROC với giá trị  $k=9$*

Ngoài ra, ROC Curve (The receiver operating characteristic curve) là một đường cong biểu diễn hiệu suất phân loại của một mô hình phân loại tại các ngưỡng threshold. Dựa trên ROC curve, ta có thể chỉ ra rằng một mô hình có hiệu quả hay không. Một mô hình hiệu quả khi có FPR (False Positive Rate - Tỷ lệ sai giả) thấp và TPR (True Positive Rate - Độ nhạy – Tỷ lệ đúng thực) cao, tức tồn tại một điểm trên ROC curve gần với điểm có tọa độ (0, 1) trên đồ thị (góc trên bên trái). Curve càng gần thì mô hình càng hiệu quả => Kết quả đánh giá điểm số ROC AUC của mô hình đạt 0.95 tương ứng với điểm FPR thấp và TPR cao tương ứng trên đồ thị, vì vậy có thể nói mô hình với K tối ưu này đang hoạt động có hiệu quả tốt.

```
from sklearn.metrics import confusion_matrix
from sklearn import metrics
confusion_matrix1 = confusion_matrix(y_test, y_pred)
```

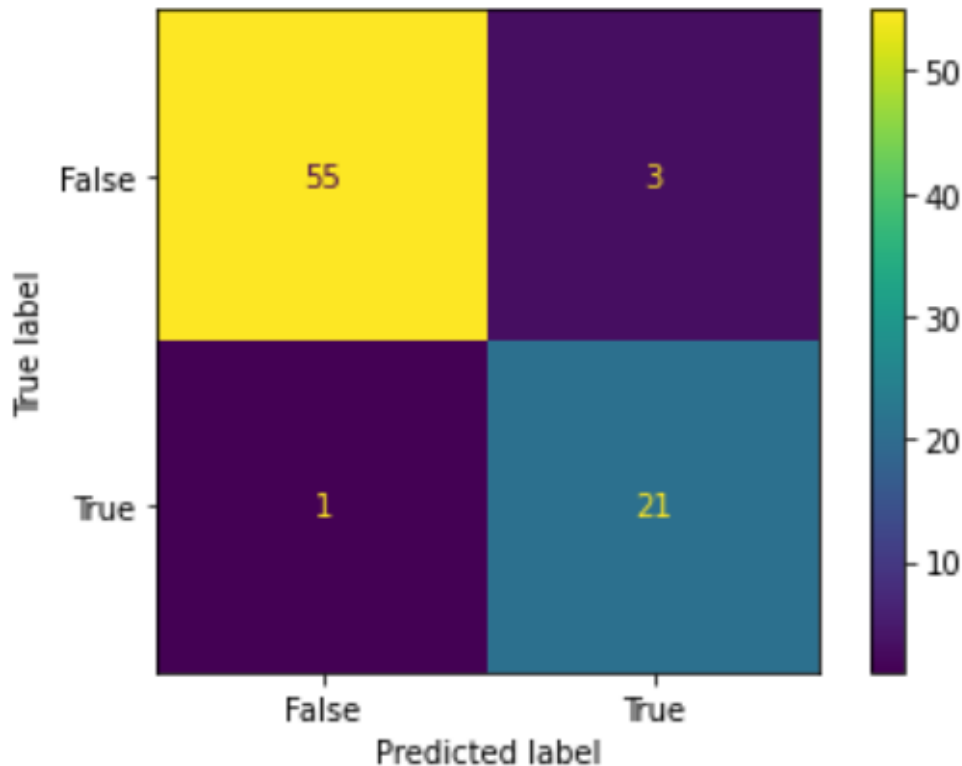
```

cm1_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix1, display_labels = [False, True])

cm1_display.plot()

plt.show()

```



Hình 31. Ma trận nhầm lẫn

Để dự đoán độ chính xác của mô hình, nhóm sử dụng ma trận nhầm lẫn. Ở ma trận này, nó so sánh các giá trị mục tiêu thực tế với các giá trị được dự đoán bởi mô hình. Do đó, nó cung cấp một cái nhìn tổng thể về cách thức hoạt động của một mô hình phân loại và những lỗi mà nó sẽ gặp phải.

Với ma trận nhầm lẫn đã tạo, nhóm thu được hai kết quả:

- Nhị phân 0/False
- Nhị phân 1/True

Trong đó Nhị phân 0/False có nghĩa là “Sai” và Nhị phân 1/True có nghĩa là “Đúng”. Dựa trên hai kết quả này, chúng ta có thể dễ dàng so sánh các kết quả phân loại



thu được với các giá trị thực tế của quan sát. Điều này giúp đánh giá hiệu suất của một mô hình phân loại.

Kết quả cho ra, trong số 80 điểm dữ liệu, có 22 điểm là đúng (True) còn 58 điểm còn lại là không phải (False).

- Trong 22 điểm là đúng thật thì có 21 điểm là được dự đoán đúng.
- Trong 58 điểm là không phải thì có 55 điểm là được dự đoán không phải

Cho thấy mô hình hoạt động tương đối chính xác, ít gặp phải lỗi.

```
#import classification_report
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.95	0.96	58
1	0.88	0.95	0.91	22
accuracy			0.95	80
macro avg	0.93	0.95	0.94	80
weighted avg	0.95	0.95	0.95	80

Cuối cùng là nhóm thực hiện xác định bằng cách xem lại báo cáo phân loại của mô hình cho thấy một kết quả trùng khớp với kết quả đo được ở trên, một lần nữa khẳng định mô hình KNN với giá trị K tối ưu ( $n\_neighbor = 9$ ) đang hoạt động rất tốt có thể ứng dụng vào thực tiễn.

### 2.6.2 Mô hình KNN với lời nguyên dữ liệu ( $n\_features = 1$ )

```
classifier = KNeighborsClassifier(n_neighbors = 9, metric =
'minkowski', p = 2)
classifier.fit(X_train_pca, y_train)
y_pred1 = classifier.predict(X_test_pca)
```

```

acc1 = accuracy_score(y_test, y_pred1)
roc1 = roc_auc_score(y_test, y_pred1)
rec1 = recall_score(y_test, y_pred1)
f1s1 = f1_score(y_test, y_pred1)
pre1 = precision_score(y_test, y_pred1)
print('Accuracy Score: ' + str(acc1))
print('ROC AUC Score: ' + str(roc1))
print('Recall Score: ' + str(rec1))
print('F1 Score: ' + str(f1s1))
print('Precision Score: ' + str(pre1))

```

**Accuracy Score: 0.875**

**ROC AUC Score: 0.829153605015674**

**Recall Score: 0.7272727272727273**

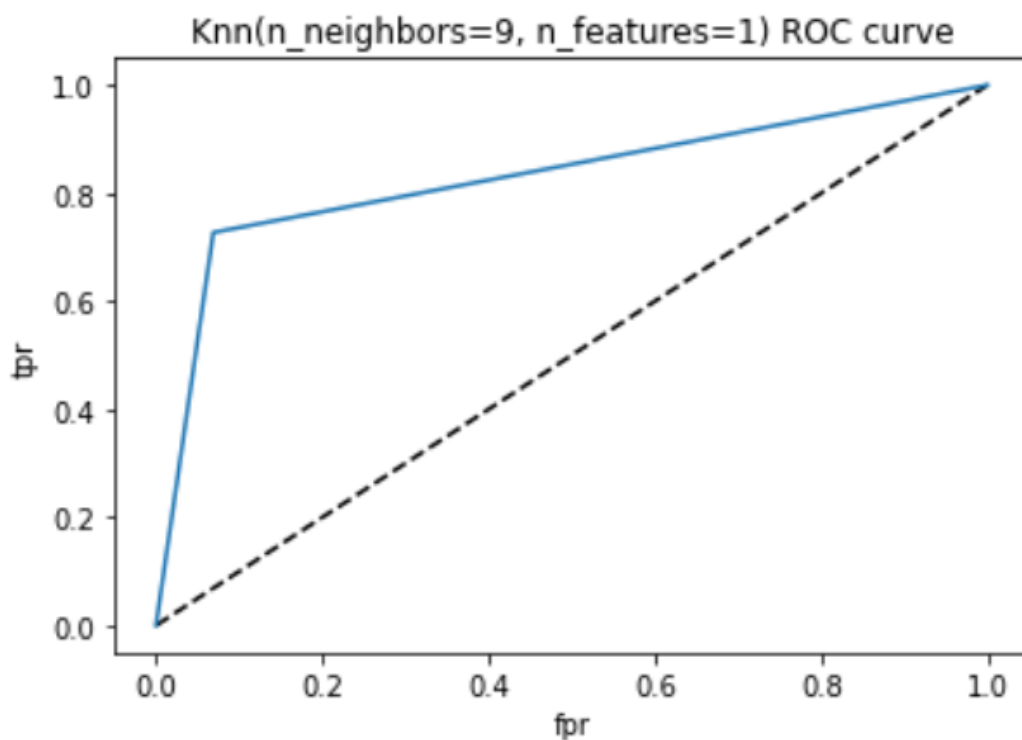
**F1 Score: 0.761904761904762**

**Precision Score: 0.8**

```

fpr1, tpr1, thresholds1 = roc_curve(y_test, y_pred1)
plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr1, tpr1, label='Knn')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('Knn(n_neighbors=9, n_features=1) ROC curve')
plt.show()

```



Hình 32. Đồ thị ROC với  $n\_neighbors=9$  và  $n\_features=1$

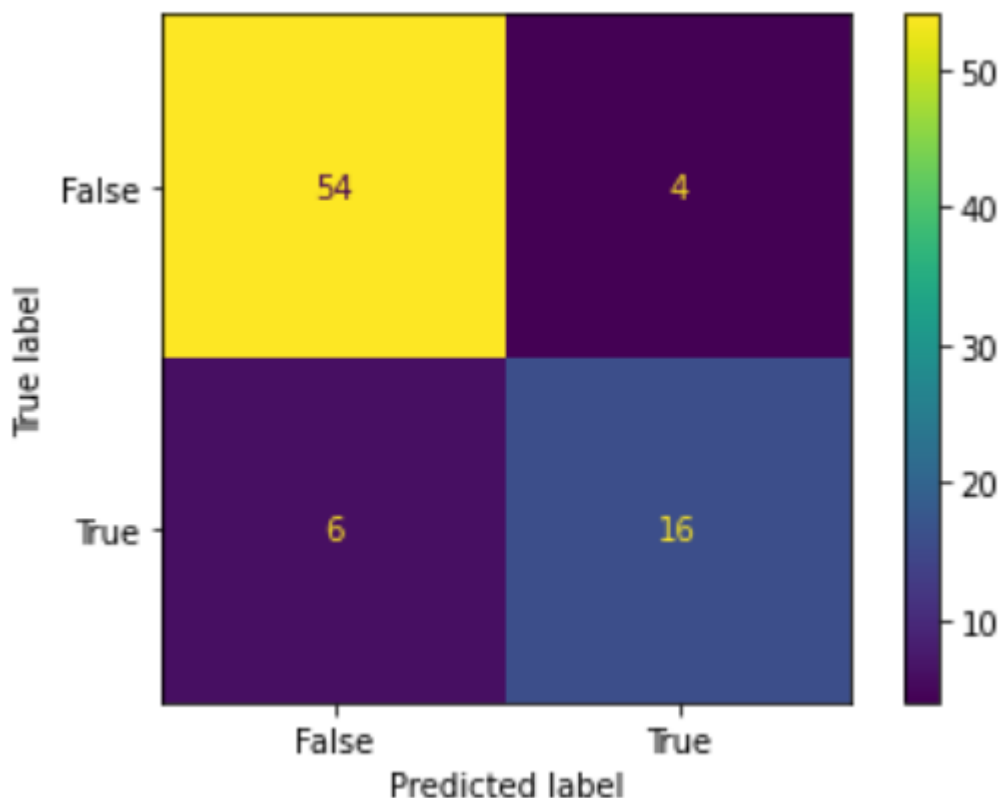
Tương tự với mô hình KNN với K tối ưu, nhóm cũng dựa vào ROC curve để đánh giá và nhận được điểm số ROC AUC của mô hình đạt gần 0.83 tương ứng với điểm fpr thấp và tpr cao ứng trên đồ thị, một con số có thể chấp nhận được, vì vậy có thể nói mô hình với lời nguyên dữ liệu ( $n\_features = 1$ ) này đang hoạt động một cách tương đối. Tuy nhiên,  $n\_features = 2$  (tự nhiên) vẫn đưa ra kết quả tốt hơn ( $accuracy = 0,95$ ) vì mô hình ít chiều và sự tương quan giữa các biến độc lập và biến phụ thuộc là đáng kể.

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics

confusion_matrix = metrics.confusion_matrix(y_test, y_pred1)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix, display_labels = [False, True])

cm_display.plot()
plt.show()
```



Hình 33. Ma trận nhầm lẫn  $n\_features = 2$

Kết quả từ ma trận nhầm lẫn của mô hình cho ra:

Trong số 80 điểm dữ liệu, có 22 điểm là đúng (True) còn 58 điểm còn lại là không phải (False).

- Trong 22 điểm là đúng thật thì chỉ có 16 điểm là được dự đoán đúng.
- Trong 58 điểm là không phải thì có 54 điểm là được dự đoán không phải.

Cho thấy mô hình hoạt động tương đối chính xác, có gặp phải lỗi nhiều hơn so với mô hình KNN với K tối ưu nhưng vẫn có thể chấp nhận được.

```
from sklearn.metrics import classification_report
target_names = ['class 0', 'class 1']
print(classification_report(y_test, y_pred1,
target_names=target_names))
```

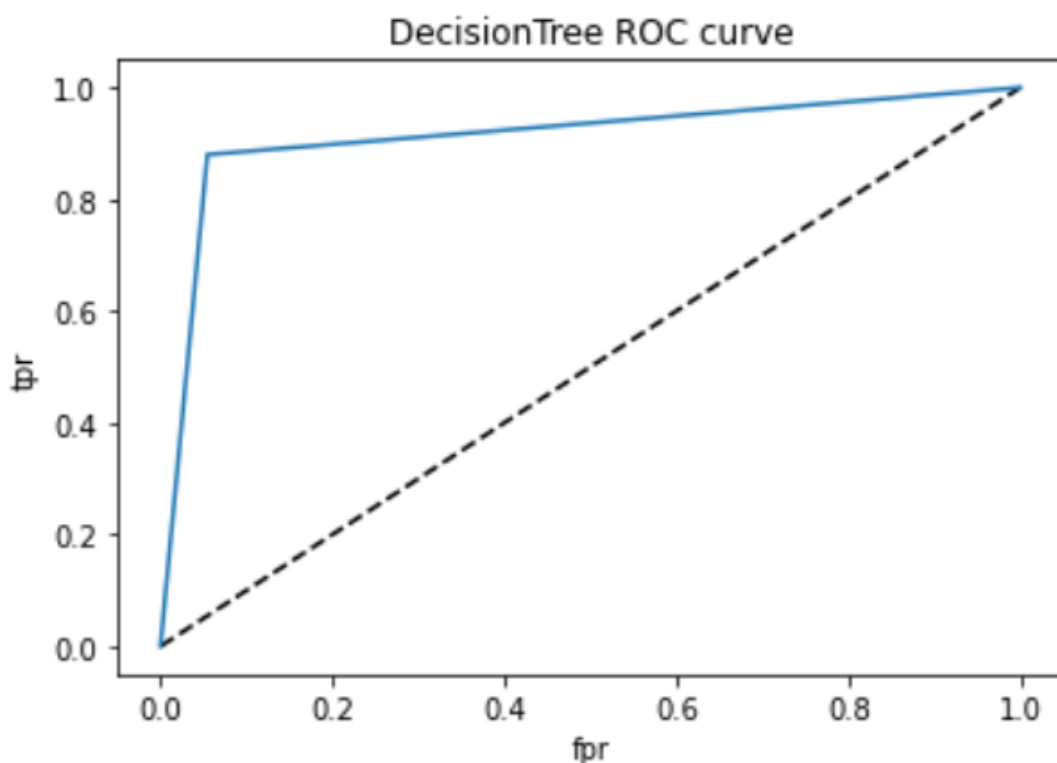
	precision	recall	f1-score	support
class 0	0.90	0.93	0.92	58
class 1	0.80	0.73	0.76	22
accuracy			0.88	80
macro avg	0.85	0.83	0.84	80
weighted avg	0.87	0.88	0.87	80

### 2.6.3 Mô hình Decision Tree

```
acc2 = accuracy_score(y_testset, predTree)
roc2 = roc_auc_score(y_testset, predTree)
rec2 = recall_score(y_testset, predTree)
f1s2 = f1_score(y_testset, predTree)
pre2 = precision_score(y_testset, predTree)
print("DecisionTrees's Accuracy Score: " + str(acc2))
print("DecisionTrees's ROC AUC Score: " + str(roc2))
print("DecisionTrees's Recall Score: " + str(rec2))
print("DecisionTrees's F1 Score: " + str(f1s2))
print("DecisionTrees's Precision Score: " + str(pre2))
```

DecisionTrees's Accuracy Score: 0.925  
DecisionTrees's ROC AUC Score: 0.9127272727272727  
DecisionTrees's Recall Score: 0.88  
DecisionTrees's F1 Score: 0.88  
DecisionTrees's Precision Score: 0.88

```
fpr2, tpr2, thresholds2 = roc_curve(y_testset, predTree)
plt.plot([0,1],[0,1], 'k--')
plt.plot(fpr2, tpr2, label='DecisionTree')
plt.xlabel('fpr')
plt.ylabel('tpr')
plt.title('DecisionTree ROC curve')
plt.show()
```



Hình 34. Đồ thị ROC của Decision Tree

Tương tự với 2 mô hình KNN ở trên, đối với mô hình Decision Tree nhóm cũng dựa vào ROC curve để đánh giá và nhận được điểm số ROC AUC của mô hình đạt tầm 0.91 tương ứng với điểm fpr thấp và tpr cao trên ứng trên đồ thị, một con số khá cao, vì vậy có thể nói mô hình Decision Tree này đang hoạt động khá tốt.

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics
```

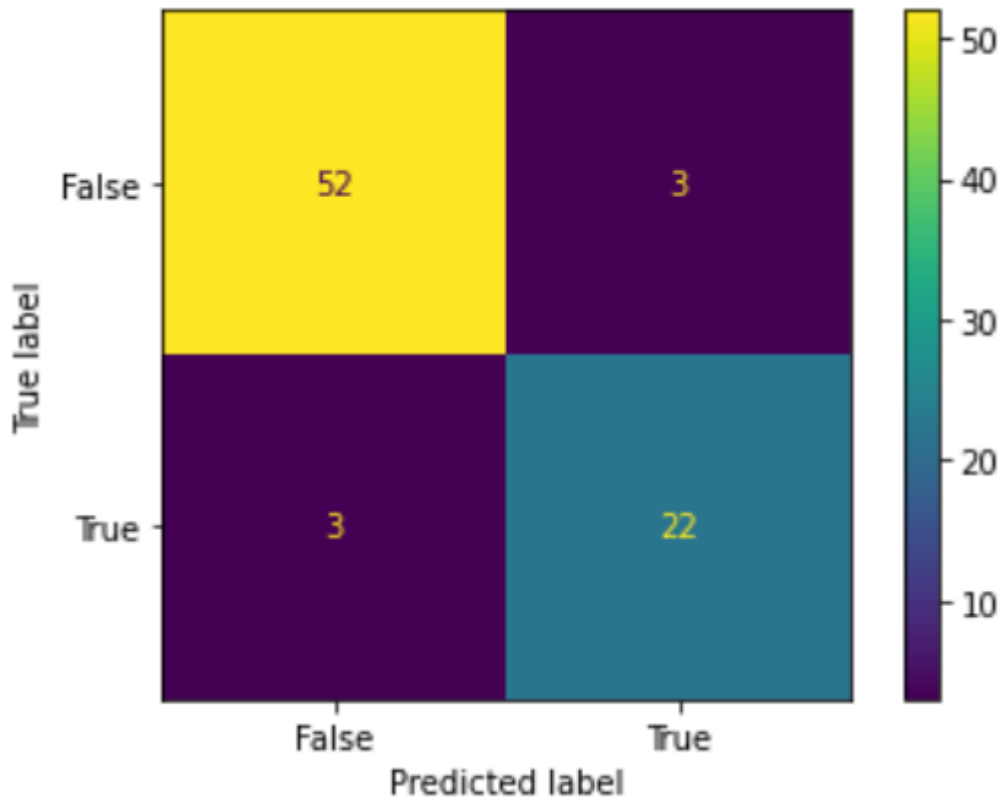
```

confusion_matrix2 = metrics.confusion_matrix(y_testset, predTree)

cm2_display = metrics.ConfusionMatrixDisplay(confusion_matrix =
confusion_matrix2, display_labels = [False, True])

cm2_display.plot()
plt.show()

```



*Hình 35. Ma trận nhầm lẫn mô hình decision tree*

```

from sklearn.metrics import classification_report
target_names = ['class 0', 'class 1']
print(classification_report(y_testset, predTree,
target_names=target_names))

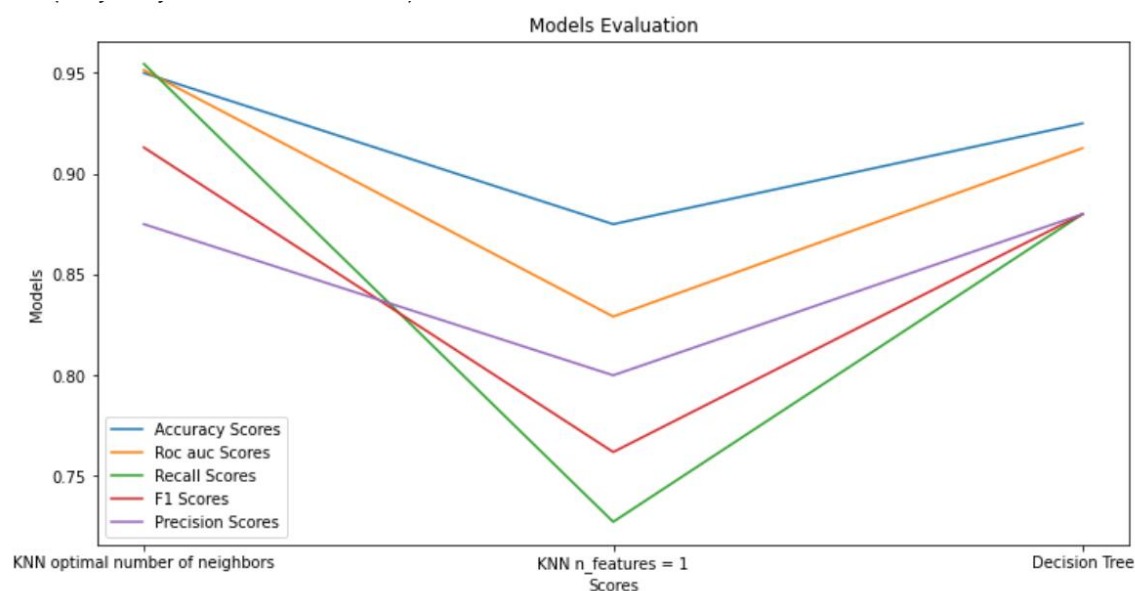
```

	precision	recall	f1-score	support
class 0	0.95	0.95	0.95	55
class 1	0.88	0.88	0.88	25
accuracy			0.93	80
macro avg	0.91	0.91	0.91	80
weighted avg	0.93	0.93	0.93	80

#### 2.6.4 So sánh các mô hình và lựa chọn

Để lựa chọn mô hình nào là tốt nhất cho bài toán, nhóm tiến hành biểu diễn kết quả đánh giá của từng mô hình dưới dạng một biểu đồ như sau:

```
acc_scores = [acc, acc1, acc2]
roc_scores = [roc, roc1, roc2]
rec_scores = [rec, rec1, rec2]
f1s_scores = [f1s, f1s1, f1s2]
pre_scores = [pre, pre1, pre2]
models = ['KNN optimal number of neighbors', 'n features = 1',
'Decision Tree']
plt.figure(figsize = (12,6))
plt.plot(models, acc_scores, label='Accuracy Scores')
plt.plot(models, roc_scores, label='Accuracy Scores')
plt.plot(models, rec_scores, label='Recall Scores')
plt.plot(models, f1s_scores, label='F1 Scores')
plt.plot(models, pre_scores, label='Precision Scores')
plt.xlabel('Scores')
plt.ylabel('Models')
plt.legend()
plt.title('Models Evaluation')
```



Hình 36. Biểu đồ so sánh chỉ số đo lường

Dựa vào biểu đồ, ta dễ dàng nhận thấy mô hình KNN với lời nguyên dữ liệu  $n\_features = 1$  cho kết quả hoạt động kém hiệu quả hơn hẳn so với 2 mô hình còn lại. Ngoài ra có thể khẳng định mô hình hoạt động có hiệu quả nhất với biểu hiện xuất sắc nhất trong cả 3 mô hình là mô hình KNN với K tối ưu  $n\_neighbors = 9$  với độ chính xác accuracy ( $0.95 > 0.925 > 0.88$ ), độ nhạy recall ( $0.954 > 0.88 > 0.727$ ) mặc dù có sự chênh lệch trong tỉ lệ dự đoán đúng precision ( $0.875 < 0.88 > 0.8$ ) nhưng không ảnh hưởng lớn vì điểm F1 - Một trung bình hài hòa Precision và Recall giữa 2 mô hình cho kết quả so sánh là  $0.913 > 0.88 > 0.76$ .

K-Nearest neighbors là một thuật toán đơn giản nhưng mang lại hiệu quả được sử dụng để giải quyết các bài toán phân loại Classifications. Hiệu quả và chất lượng của thuật toán K-Nearest neighbors phụ thuộc nhiều vào việc chọn tham số K láng giềng  $n\_neighbors$  và cả lời nguyên dữ liệu  $n\_features$ . Vì vậy, để tối ưu mô hình cần thử nghiệm với các giá trị đầu vào K láng giềng khác nhau cũng như với các chiều dữ liệu trước và chọn lựa mô hình thực sự hiệu quả, phù hợp với bài toán.



## Phụ lục: Báo cáo quá trình làm việc nhóm

### BÁO CÁO LÀM VIỆC NHÓM

STT	Họ và tên	Phụ trách công việc	Đánh giá
1	Vũ Thị Phương Anh (NT)	<ul style="list-style-type: none"><li>- Lên cấu trúc đề tài</li><li>- Ưu nhược điểm của thuật toán</li><li>- Quy trình đề xuất cho bài toán</li><li>- Mô hình KNN với lời nguyên của chiều dữ liệu</li><li>- Chạy lại code tổng hợp</li></ul>	100%
2	Đinh Thị Thúy An	<ul style="list-style-type: none"><li>- Tìm hiểu về KNN</li><li>- Thuật toán KNN trong R/Python</li><li>- Mô tả dữ liệu bài toán</li><li>- Mô hình KNN với sự thay đổi của hệ số k</li><li>- Tóm tắt chương 2</li><li>- Thuyết trình</li></ul>	100%
3	Nguyễn Thị Thanh Bình	<ul style="list-style-type: none"><li>- Tìm hiểu về KNN</li><li>- Cơ sở lý thuyết của thuật toán</li><li>- Đo lường KNN</li><li>- Mô hình khác có thể áp dụng vào bài toán</li><li>- Tổng hợp word</li><li>- Thuyết trình</li></ul>	100%
4	Lê Thị Thùy Linh	<ul style="list-style-type: none"><li>- Tìm hiểu về KNN</li><li>- Ứng dụng của thuật toán KNN</li><li>- Mô tả bài toán</li><li>- Phân tích và khai thác dữ liệu EDA</li><li>- Lời cảm ơn, báo cáo làm việc nhóm</li></ul>	100%

		- Thiết kế slide	
5	Mai Lê Ngọc Trâm	<ul style="list-style-type: none"> <li>- Tìm hiểu về KNN</li> <li>- Giới thiệu thuật toán</li> <li>- Thử nghiệm mô hình KNN với sự thay đổi của hệ số <math>p</math></li> <li>- Kết luận về bài toán</li> <li>- Tóm tắt chương 1</li> <li>- Thiết kế slide</li> </ul>	100%

### QUY TRÌNH LÀM VIỆC

Thời gian	Công việc triển khai
06/02/2023 - 11/02/2023	<ul style="list-style-type: none"> <li>- Bầu nhóm trưởng</li> <li>- Chốt thuật toán</li> <li>- Tìm hiểu về KNN, bài toán cho thuật toán</li> </ul>
13/02/2023 - 18/02/2023	<ul style="list-style-type: none"> <li>- Chốt cấu trúc đề tài</li> <li>- Brainstorm dataset cho bài toán</li> <li>- Phân công công việc chương I (Cơ sở lý thuyết, giới thiệu, ứng dụng và ưu nhược điểm của KNN)</li> </ul>
20/02/2023 - 25/02/2023	<ul style="list-style-type: none"> <li>- Chốt dataset bài toán</li> <li>- Cả nhóm nhận xét và bạn phụ trách tiến hành chỉnh sửa các nội dung phân công ở chương I</li> <li>- Phân công công việc chương II (Mô tả bài toán, dữ liệu, quy trình đề xuất, đo lường KNN, EDA, mô hình)</li> </ul>
27/02/2023 - 05/03/2023	<ul style="list-style-type: none"> <li>- Trình bày lại nội dung đã chỉnh sửa ở chương I, tiến hành duyệt lần nữa</li> <li>- Trình bày nội dung chương II, cả nhóm tiến hành</li> </ul>

	nhận xét và bạn phụ trách chỉnh sửa
06/03/2023 - 11/03/2023	<ul style="list-style-type: none"> <li>- Duyệt lại nội dung chương II</li> <li>- Tóm tắt chương I, chương II, lời cảm ơn, báo cáo, tổng hợp</li> <li>- Thiết kế slide</li> </ul>

## Tài liệu tham khảo:

- Nguyễn Hà Anh Dũng, Nguyễn Quang Long, Nguyễn Thị Thảo (2016). *Phân lớp dữ liệu số bằng giải thuật KNN*, Báo cáo bài tập lớn, Trường Đại Học Công Nghiệp Hà Nội. [\[Crossref\]](#)
- Nguyễn Nhật Quang (2020). Tổng quan về thuật toán láng giềng gần k-NN. [\[Crossref\]](#)
- Vũ Hữu Tiệp (2017). *Machine Learning cơ bản*. [\[Crossref\]](#)
- Hafidh Fikri (2019). Loan Approval Prediction. [\[Crossref\]](#)
- Mario Caesar (2022). Loan Prediction w/ Various ML Models. [\[Crossref\]](#)
- Cássia Sampaio (2022). Guide to the K-Nearest Neighbors Algorithm in Python and Scikit-Learn. [\[Crossref\]](#)
- Sarang Anil Gokte (2023). Most Popular Distance Metrics Used in KNN and When to Use Them. [\[Crossref\]](#)
- Sadegh Bafandeh Imandoust And Mohammad Bolandraftar (2013). *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background*, Department of Economics, Payame Noor University, Tehran, Iran. [\[Crossref\]](#)
- Utsav Mishra (2022). An Introduction to K-nearest Neighbor (KNN) Algorithm. [\[Crossref\]](#)
- Aishwarya Singh (2023). A Practical Introduction to K-Nearest Neighbors Algorithm for Regression (with Python code). [\[Crossref\]](#)
- Mai Phạm (2020). Thuật toán K láng giềng gần nhất (K-Nearest Neighbor - KNN) là gì? [\[Crossref\]](#)
- Why is Nearest Neighbor a Lazy Algorithm? [\[Crossref\]](#)
- Hoàng Thu Thảo (2021). Encoding categorical features in Machine learning. [\[Crossref\]](#)
- IBM (2023). What is the k-nearest neighbors algorithm?. [\[Crossref\]](#)
- Badreesh Shetty (2022). What Is the Curse of Dimensionality? [\[Crossref\]](#)
- Huy Nguyễn (2021). K-Nearest Neighbors (KNN). [\[Crossref\]](#)
- NominSim (2012). Value of k in k nearest neighbor algorithm. [\[Crossref\]](#)
- A. Aylin Tokuç (2022). K-Nearest Neighbors and High Dimensional Data. [\[Crossref\]](#)
- Nguyễn Thị Hợp (2019). KNN (K-Nearest Neighbors) #1. [\[Crossref\]](#)
- Javatpoint (2023). K-Nearest Neighbor(KNN) Algorithm for Machine Learning. [\[Crossref\]](#)

Kevin Coombes (2018). How can I use KNN for mixed data (categorical and numerical)? [[Crossref](#)]

Vibhu Singh (2018). K-Nearest Neighbors Algorithm: Steps to Implement in Python. [[Crossref](#)]

Pham Dinh Khanh (2020). Đánh giá mô hình phân loại trong ML. [[Crossref](#)]

Scikit learn (2022). KNeighborsClassifier. [[Crossref](#)]

Andrew Bruce, Peter C. Bruce, Peter Gedeck (2017). *Practical Statistics for Data Scientists 50 Essential Concepts Using R and Python*.