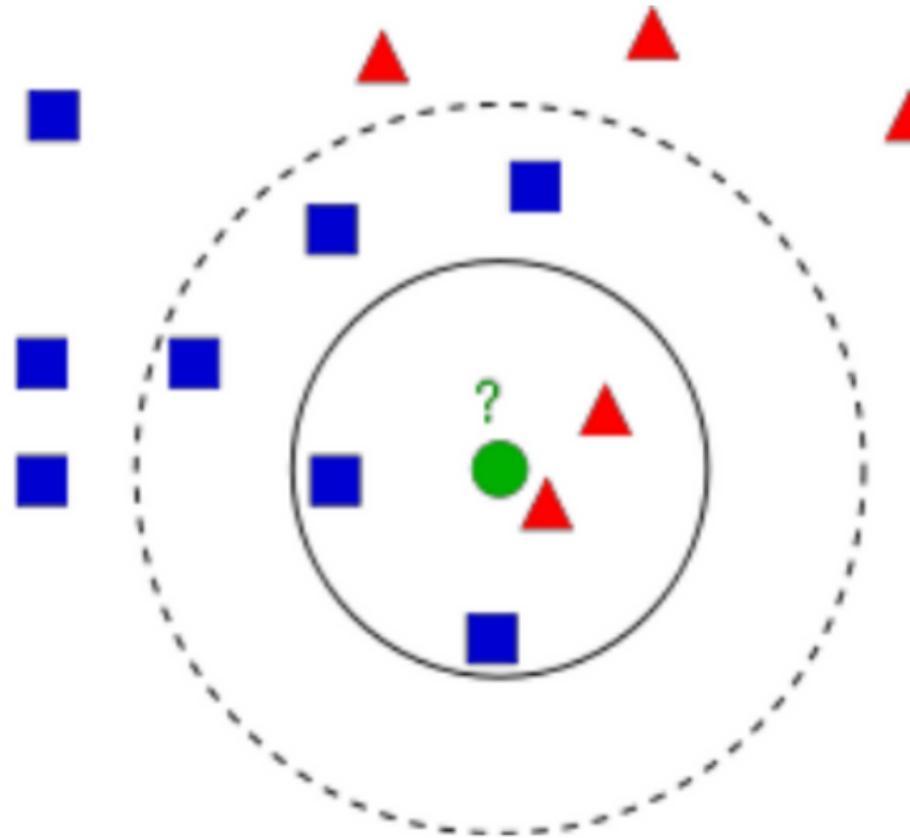


# THUẬT TOÁN K-NEAREST NEIGHBORS

*What is KNN algorithm?*



# Nội dung chính

## *Chương 1: Thuật toán KNN*

*Giới thiệu tổng quan về thuật toán K-nearest neighbors*

## *Chương 2: Ứng dụng vào bài toán*

*Áp dụng KNN đưa ra dự đoán quyết định mua sản phẩm của người tiêu dùng dựa trên bộ dữ liệu quảng cáo MXH.*

# Cơ sở lý thuyết

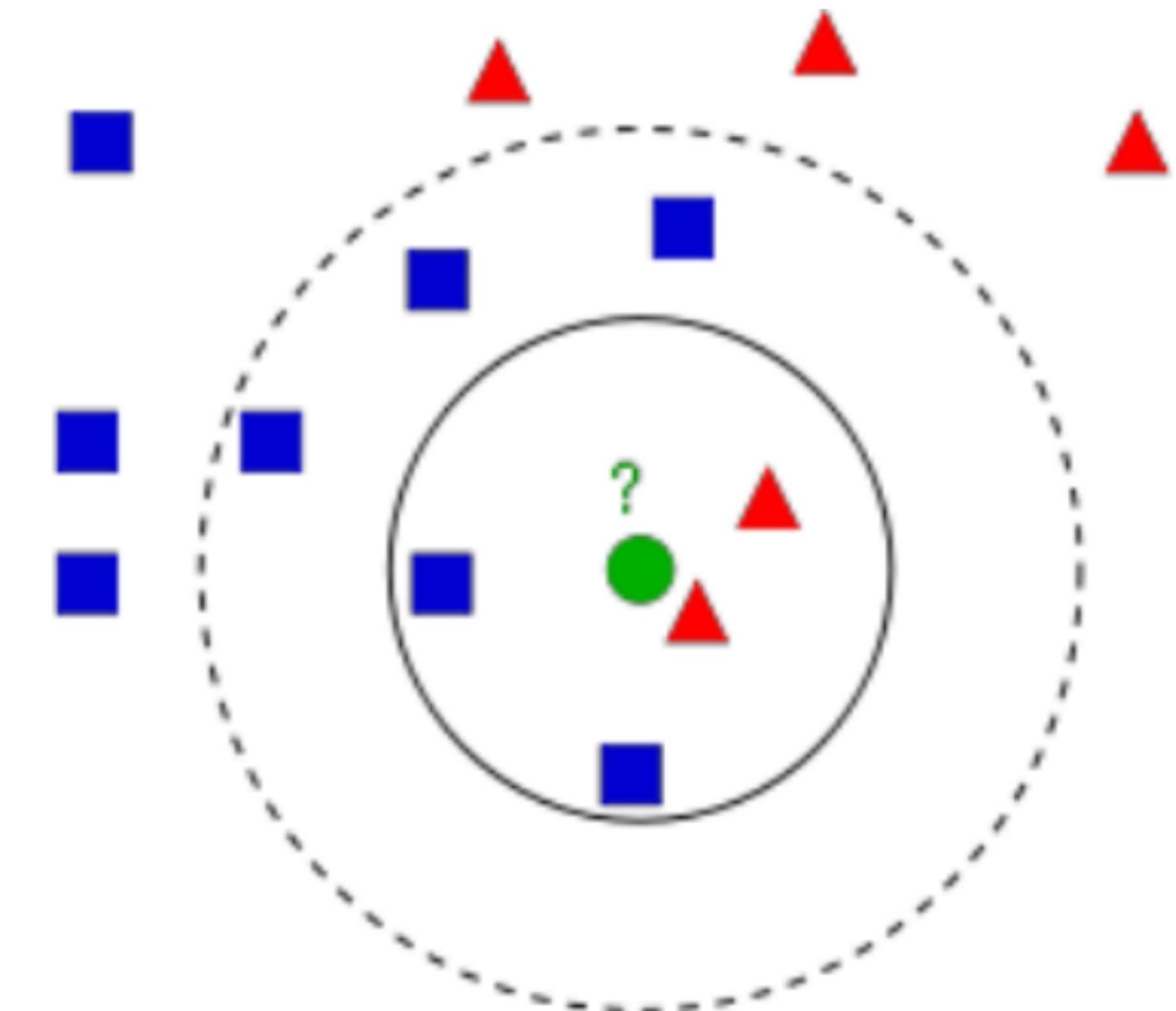
## Định nghĩa



*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.*

# Quy trình làm việc của KNN

- 1 Xác định tham số  $K =$  số láng giềng gần nhất
- 2 Tính khoảng cách đối tượng cần phân lớp
- 3 Sắp xếp khoảng cách & xác định  $K$  láng giềng gần nhất
- 4 Lấy tất cả các lớp của  $K$  láng giềng gần nhất
- 5 Dựa vào phần lớn lớp của  $K$  xác định lớp cho đối tượng



# Khoảng cách trong không gian vector

## Distance functions

Euclidean

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

Manhattan

$$\sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$\left( \sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

# Khoảng cách trong không gian vector

## 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  $x = 0$ .
- $f(ax) = |a|f(x), \forall a \in R$ .
- $f(x_1) + f(x_2) \geq f(x_1 + x_2), \forall x_1, x_2 \in R$

# Một số norm thường dùng

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

Khoảng cách Euclidean là norm 2 khi  $p = 2$ :

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

Với  $p$  là một số không nhỏ hơn 1 bất kỳ:

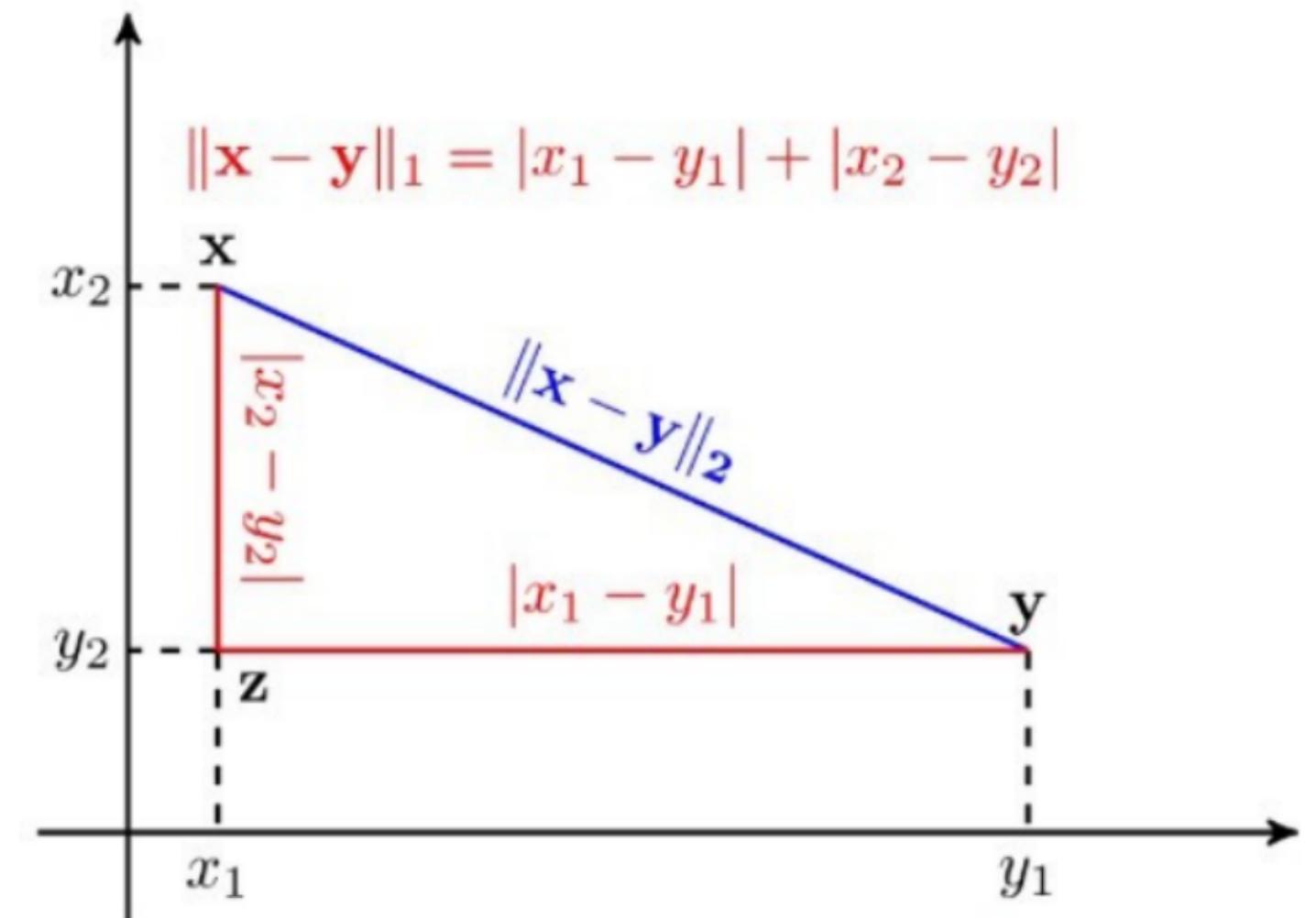
$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{\frac{1}{p}} \quad (2)$$

Khi  $p = 1$  chúng ta có:

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

Khi  $p > \infty$ , ta có norm  $p$ :

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



# Thuật toán KNN trong Python

- **Nạp thư viện “sklearn”**

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

- **Xây dựng, đào tạo và dự đoán**

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

# Đo lường KNN

## • Đo lường bằng 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.

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

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

# Đo lường KNN

- **Accuracy**

Là tỉ lệ giữa số điểm được dự đoán, phân lớp đúng và tổng số điểm trong tập dữ liệu kiểm thử.

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

- **Precision**

Precision trả lời cho câu hỏi trong các trường hợp được dự báo là positive thì có bao nhiêu trường hợp là đúng?

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

# Đo lường KNN

- **Recall**

Recall trả lời cho câu hỏi trong tất cả các trường hợp Positive, bao nhiêu trường hợp đã được dự đoán chính xác?

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

- **F1 score**

Là sự kết hợp 2 chỉ số Precision và Recall để đánh giá độ tin cậy chung của mô hình.

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

# Đo lường KNN

- **Đo lường KNN bằng AUC**

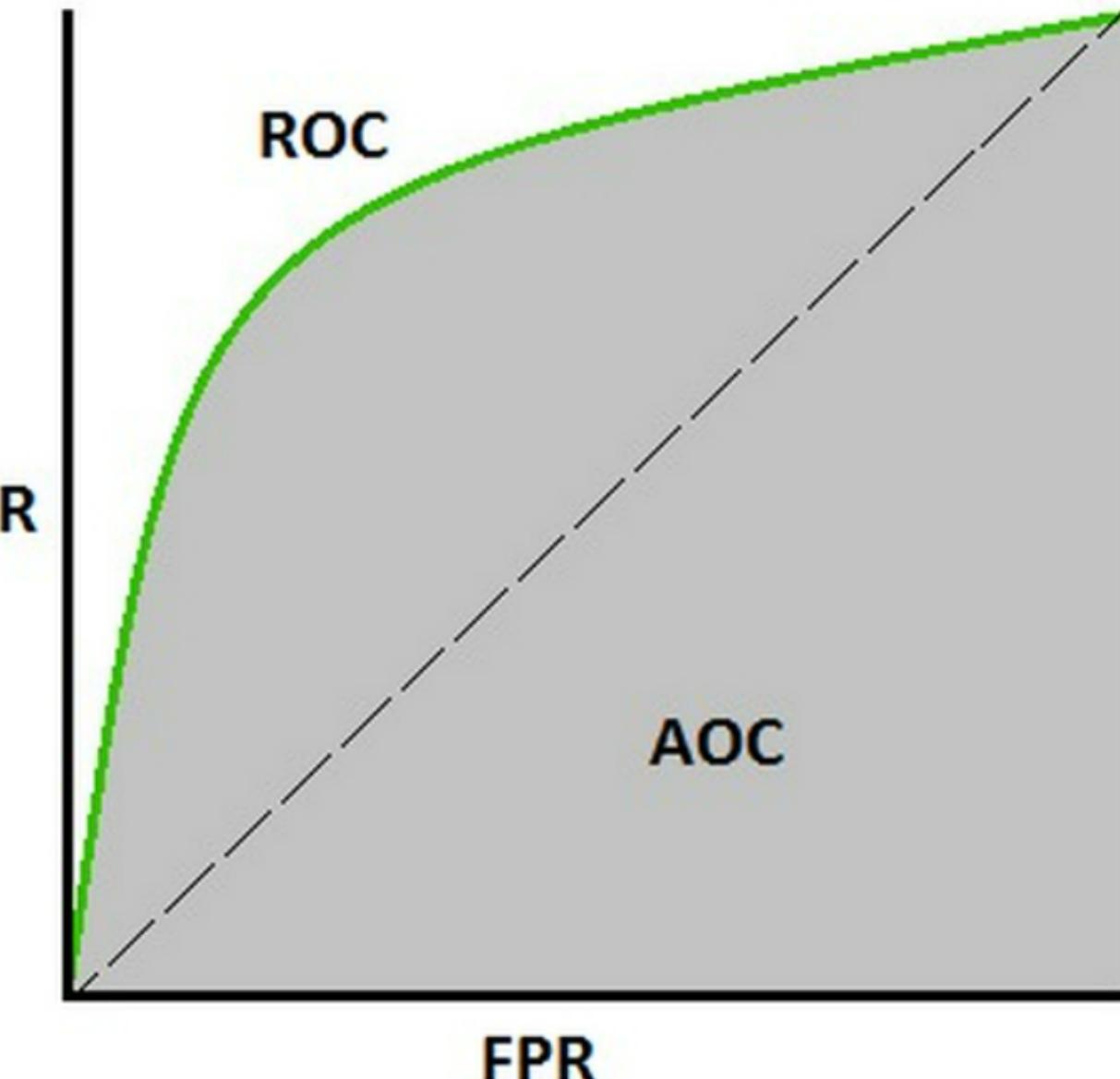
ROC là đường cong biểu diễn khả năng phân loại của một mô hình phân loại tại các ngưỡng threshold.

- TPR: Tỷ lệ các trường hợp phân loại đúng positive trên tổng số các trường hợp thực tế là positive.

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

- FPR: Tỷ lệ dự báo sai các trường hợp thực tế là negative ra positive trên tổng số các trường hợp thực tế là negative.

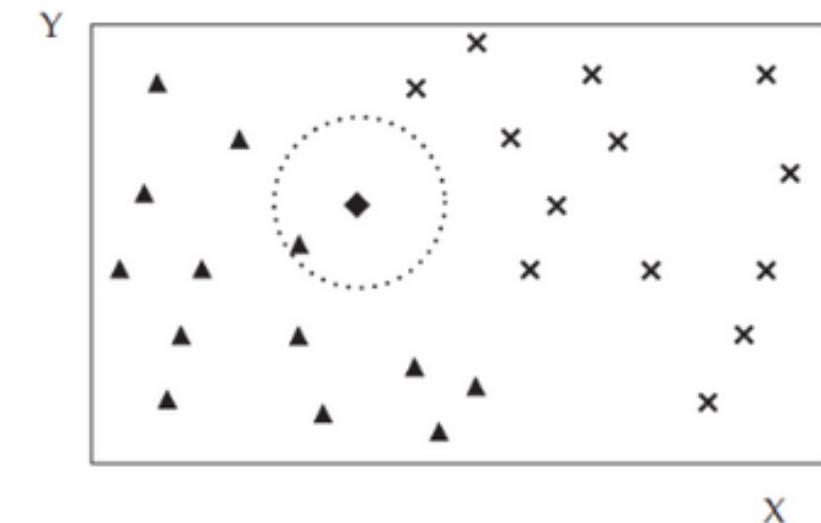
$$\text{FPR} = 1 - \text{specificity} = \frac{FP}{\text{total negative}}$$



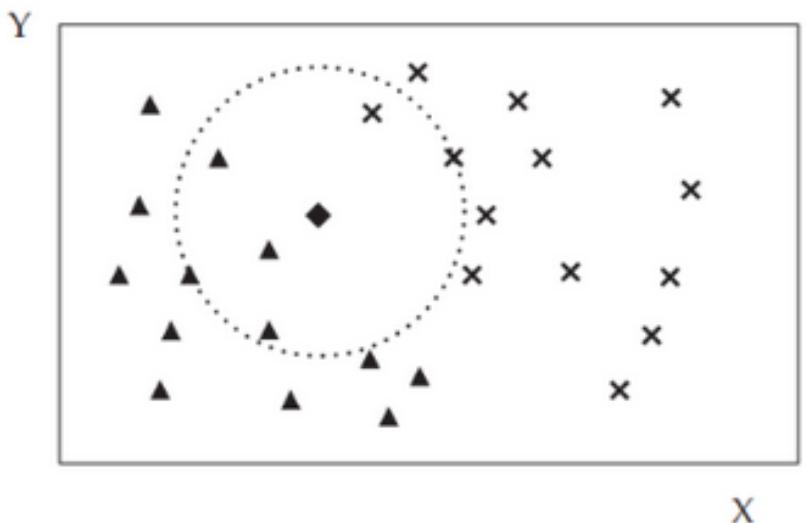
# Ứng dụng của thuật toán

- 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.

A. KNN With New Observation, K=1



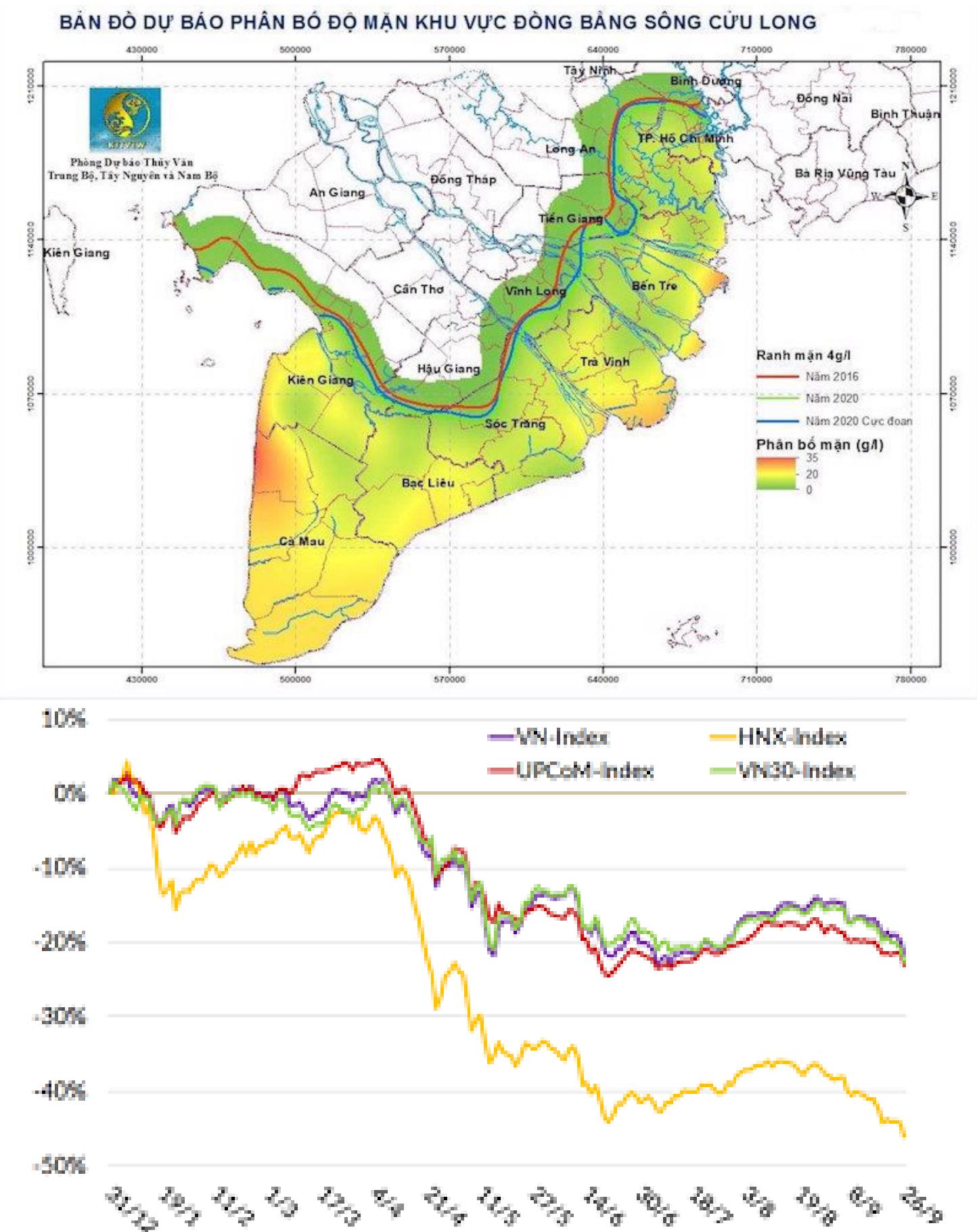
B. KNN With New Observation, K=5



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	?

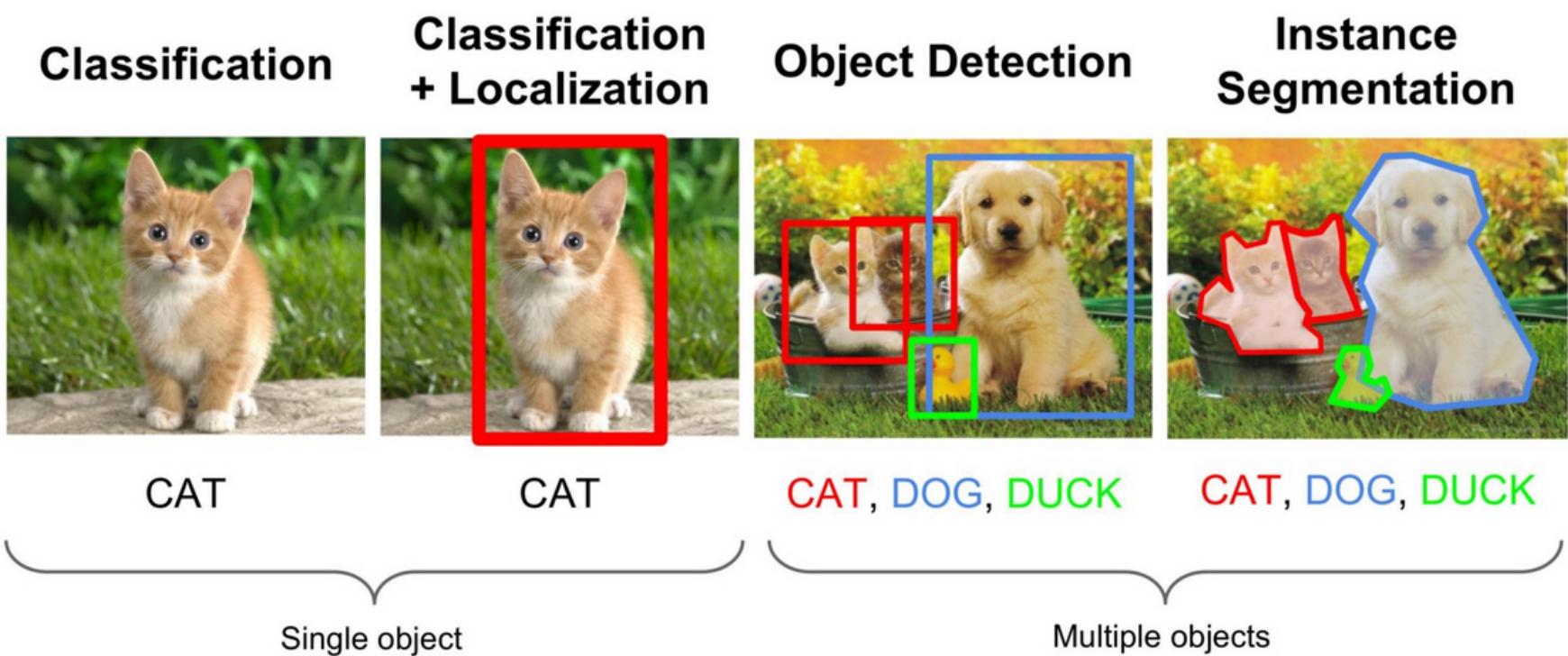
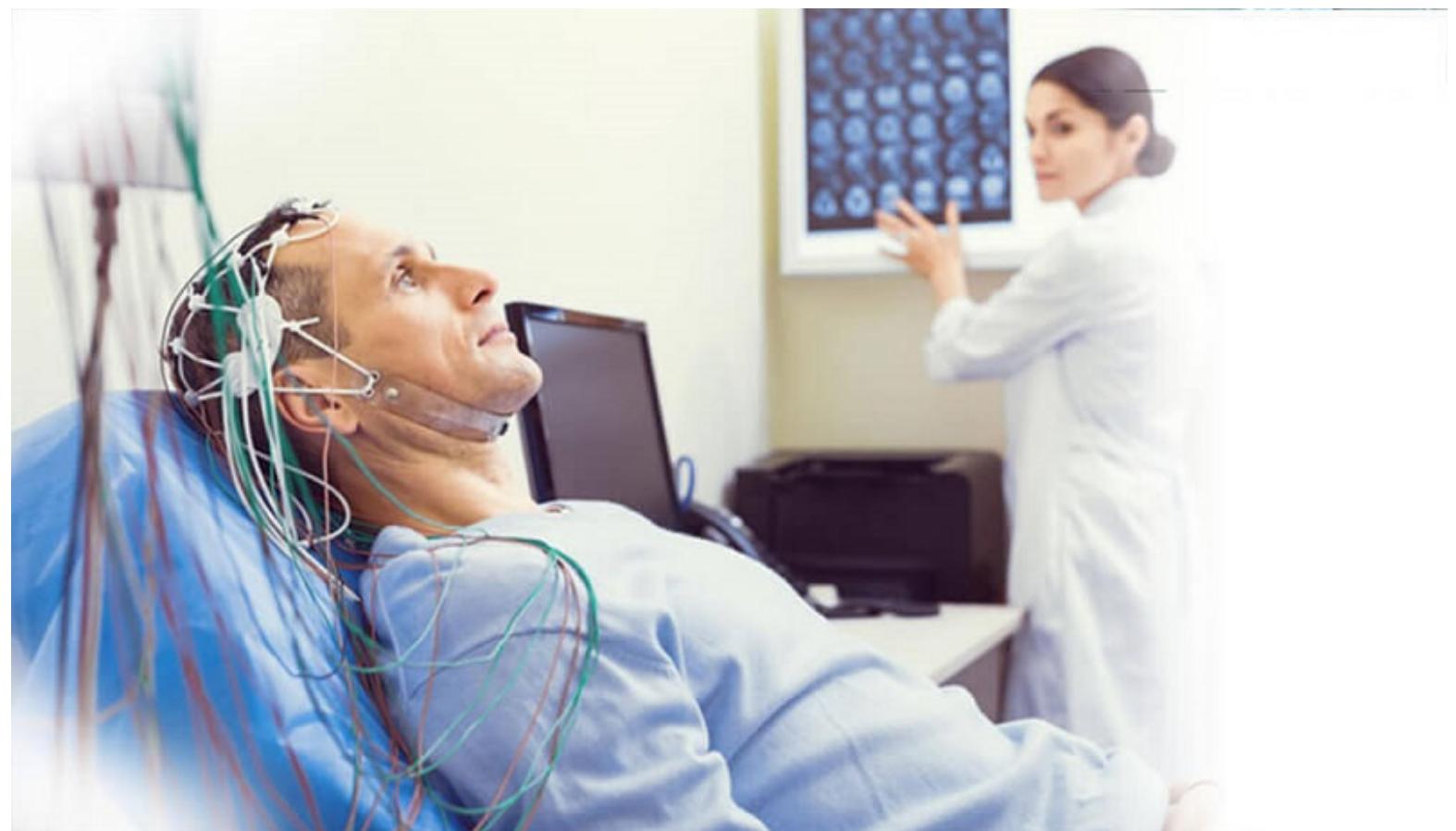
# Ứng dụng của thuật toán

- Lĩnh vực nông nghiệp: 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
- Lĩnh vực tài chính: Dự báo thị trường chứng khoán, dự đoán giá cổ phiếu, Điểm tín dụng, quản lý phê duyệt khoản vay, dự đoán phá sản, quản lý rủi ro tài chính, phân tích rửa tiền



# Ứng dụng của thuật toán

- Lĩnh vực y học: Dự đoán bệnh trạng, Xác định các yếu tố nguy cơ gây bệnh, ước tính chất trong cơ thể, phân tích dữ liệu biểu hiện gen vi mảng,...
- Thị giác máy tính: Thuật toán KNN được dùng để phân loại hình ảnh
- Nhận dạng các mẫu: hỗ trợ phát hiện và phát hiện các mẫu



# Ưu điểm và nhược điểm & vấn đề mở rộng

## 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.*

# Ưu điểm và nhược điểm & vấn đề mở rộng

## Nhược điểm



- Giá trị  $k$  cần được tính toán kỹ càng, các vấn đề về *overfitting*, không đạt hiệu quả
- Cần nhiều dung lượng và thời gian để thực hiện
- Chịu ảnh hưởng của “lời nguyền của chiều dữ liệu”
- Khó có thể áp dụng KNN lên tập dữ liệu có phân loại (*categorical features*), loại dữ liệu không phải là số

# CHƯƠNG II: ỨNG DỤNG THUẬT TOÁN KNN VÀO CASE STUDY

*Bài toán dự đoán khách hàng có mua hàng hay không?*



# Mô tả bài toán



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.

# Mô tả dữ liệu

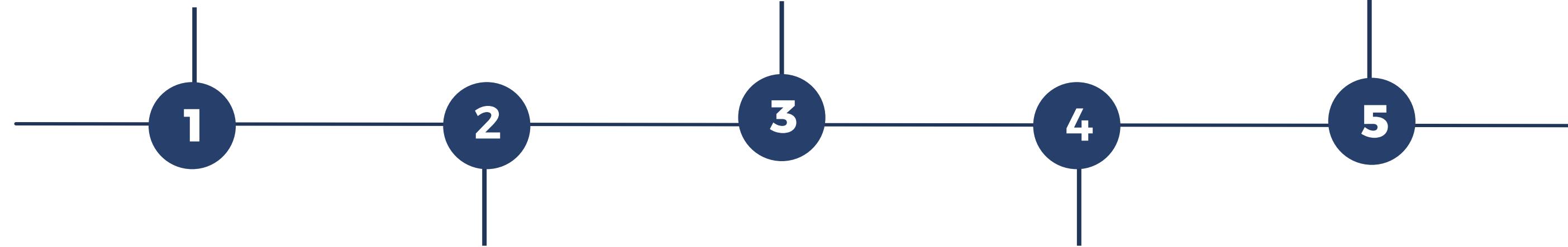
- Nam: 196
- Nữ: 204
- Độ tuổi: 18 - 60
- Thu nhập: 15000-150000

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

Tập dữ liệu gồm 400 dòng, 5 cột

# Quy trình đề xuất

**Thu thập  
dữ liệu**



**Xử lý tập  
dữ liệu**

**Đánh giá  
mô hình**

**Tiền xử lý  
dữ liệu**

**Xây dựng  
mô hình**

# Phân tích và khai thác dữ liệu

Kiểm tra xem dữ liệu có giá trị null và duplicate 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
```

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

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

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

```
0
```

## 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

## Tổng quan về dữ liệu

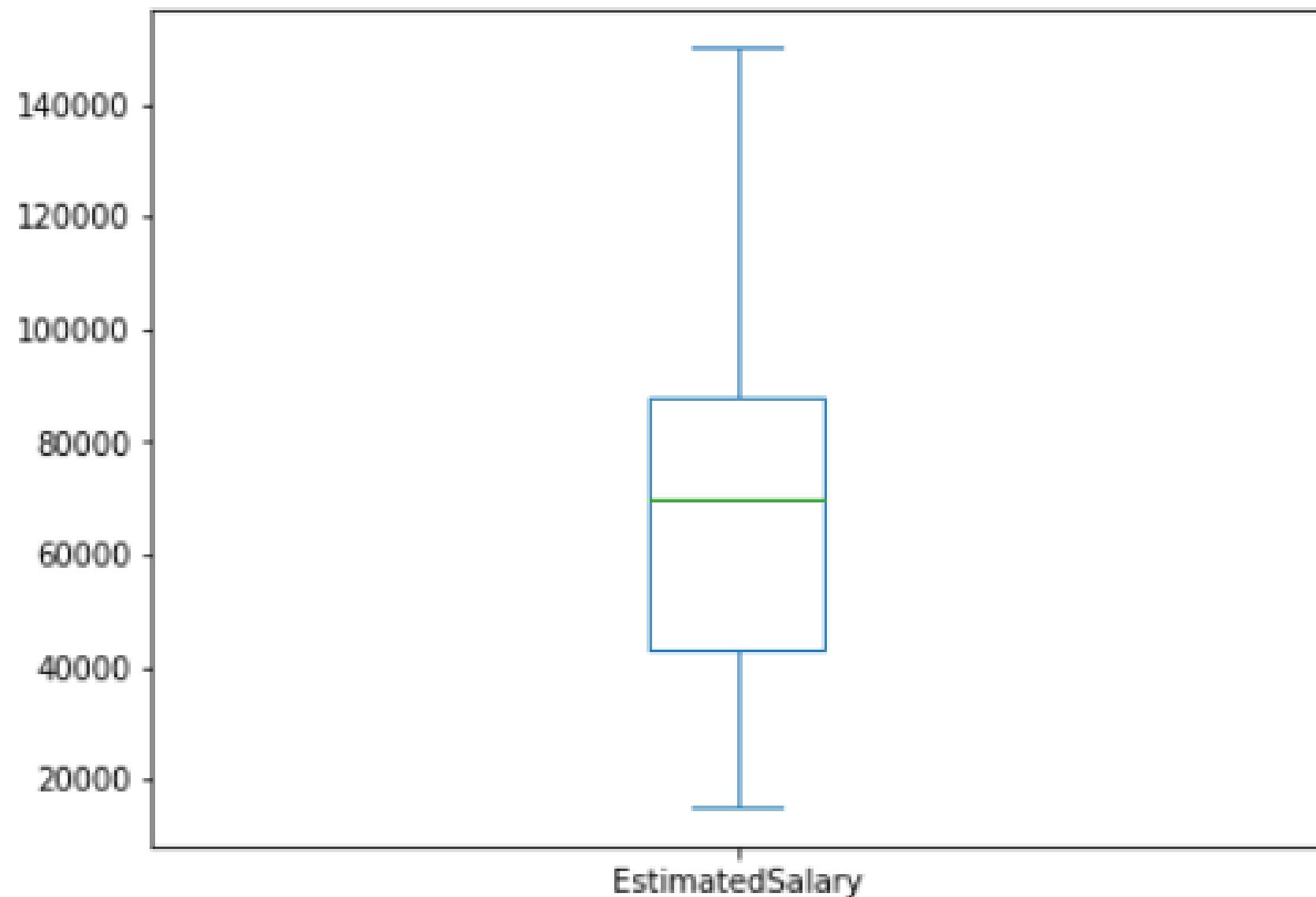
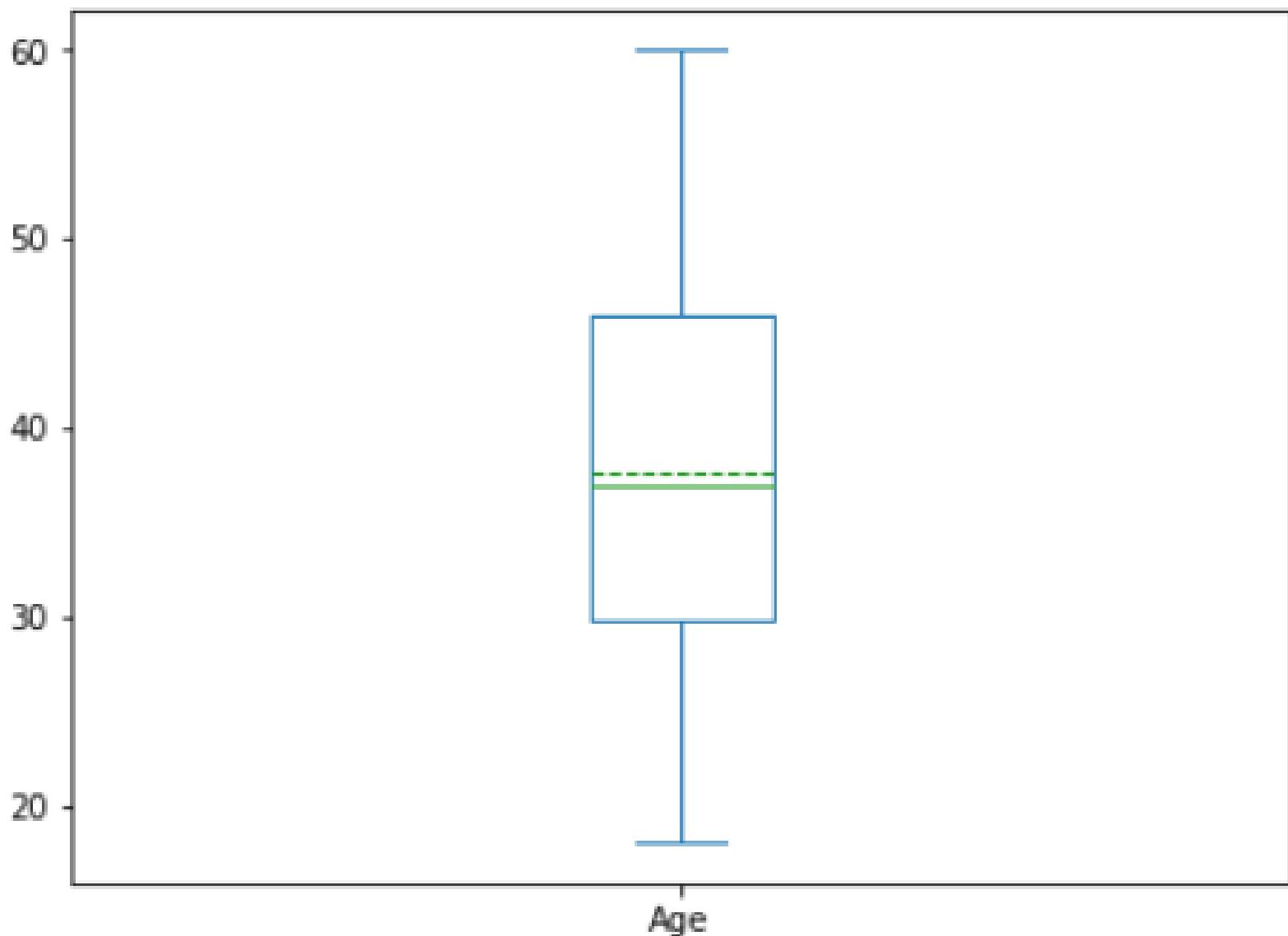
```
social_network_ads.describe()
```

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

## Kiểm tra giá trị ngoại lệ

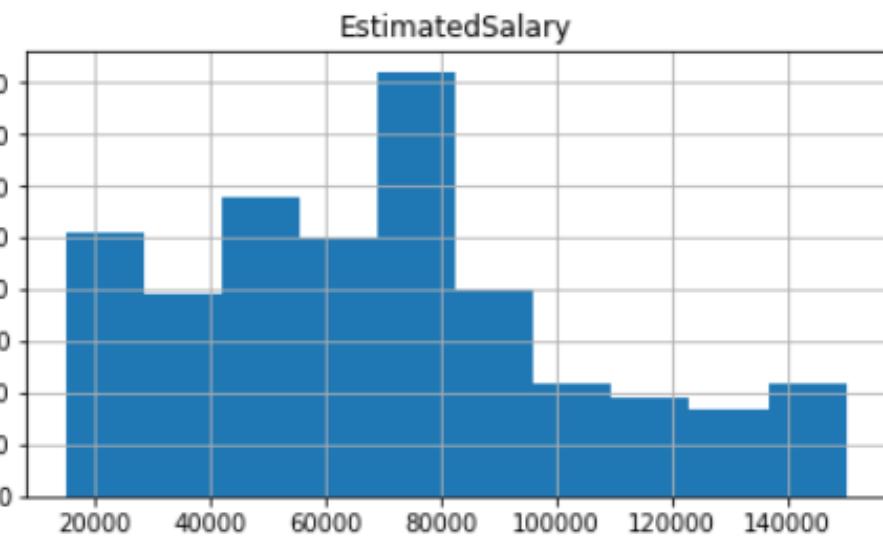
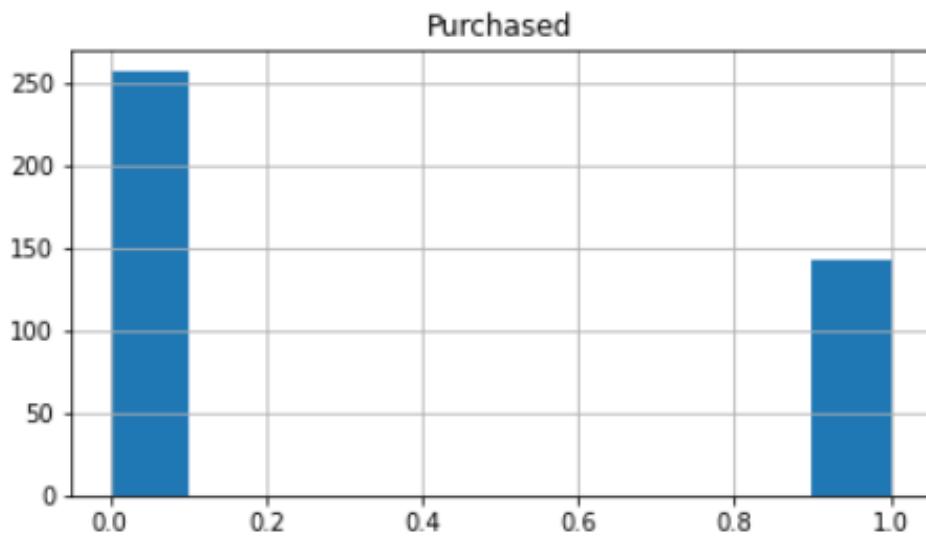
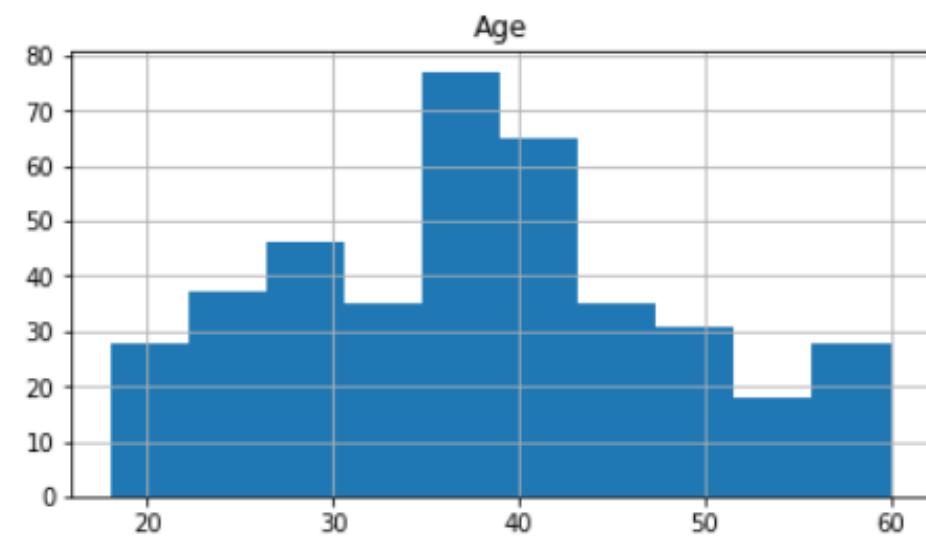
```
fig, ax = plt.subplots(1,2, figsize = (15, 5)) #1 dòng, 2 cột  
social_network_ads['Age'].plot.box(ax = ax[0], showmeans = True, meanline = True)  
#meanline: hiển thị đường trung bình  
social_network_ads['EstimatedSalary'].plot.box(ax = ax[1], showmeans = True, meanline = True)
```

<AxesSubplot:>

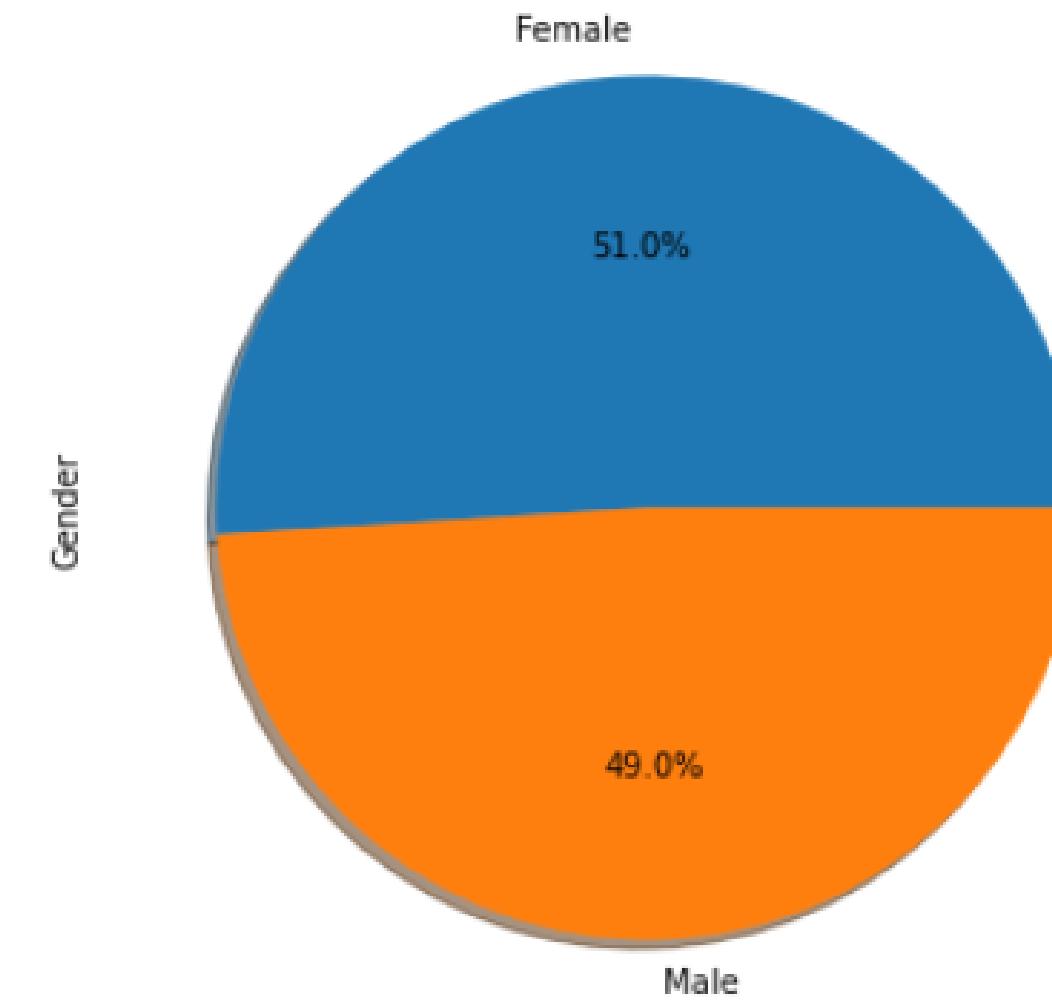


## Biểu đồ phân bổ của biến

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



```
<AxesSubplot:ylabel='Gender'>
```

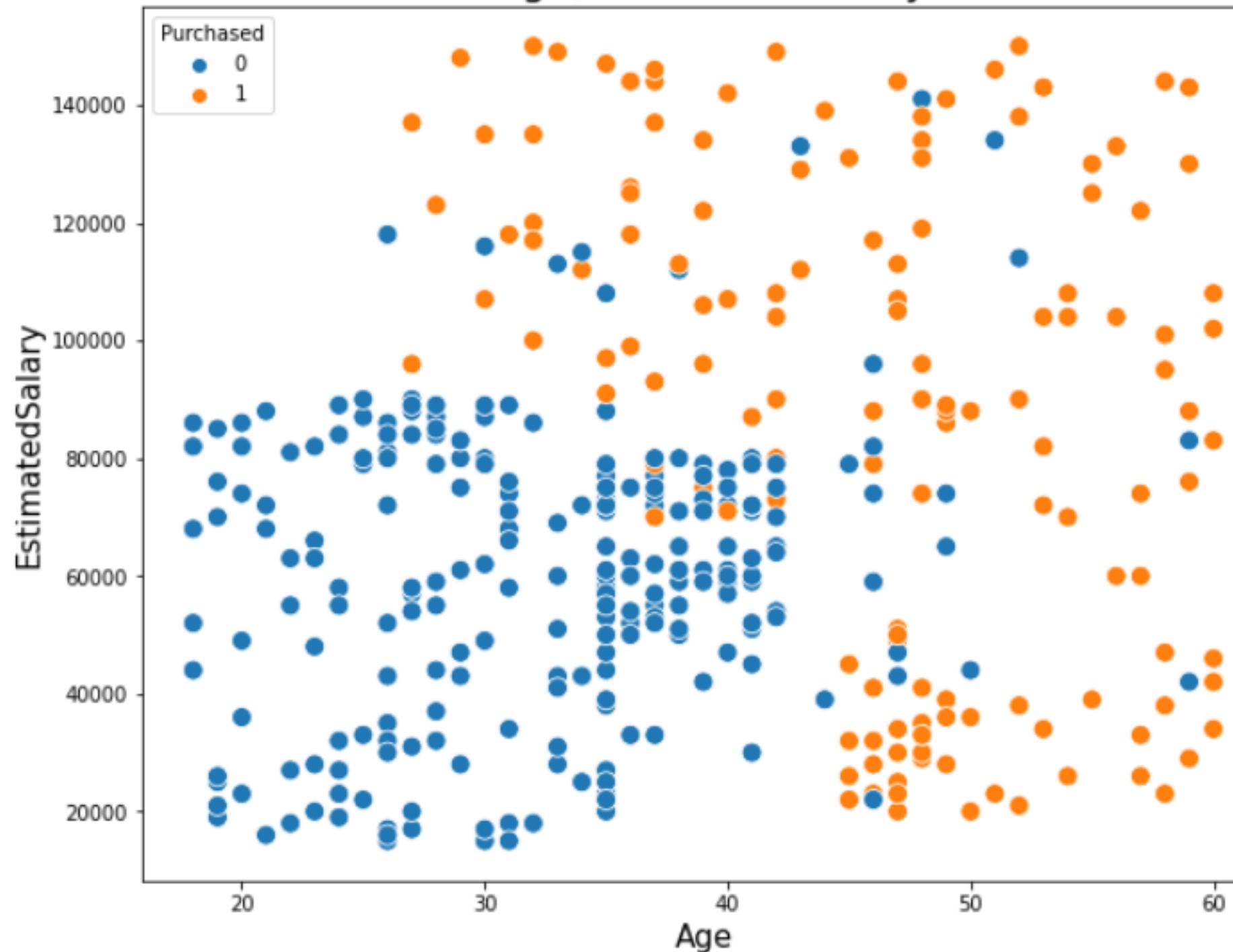


Biểu đồ tròn biến giới tính

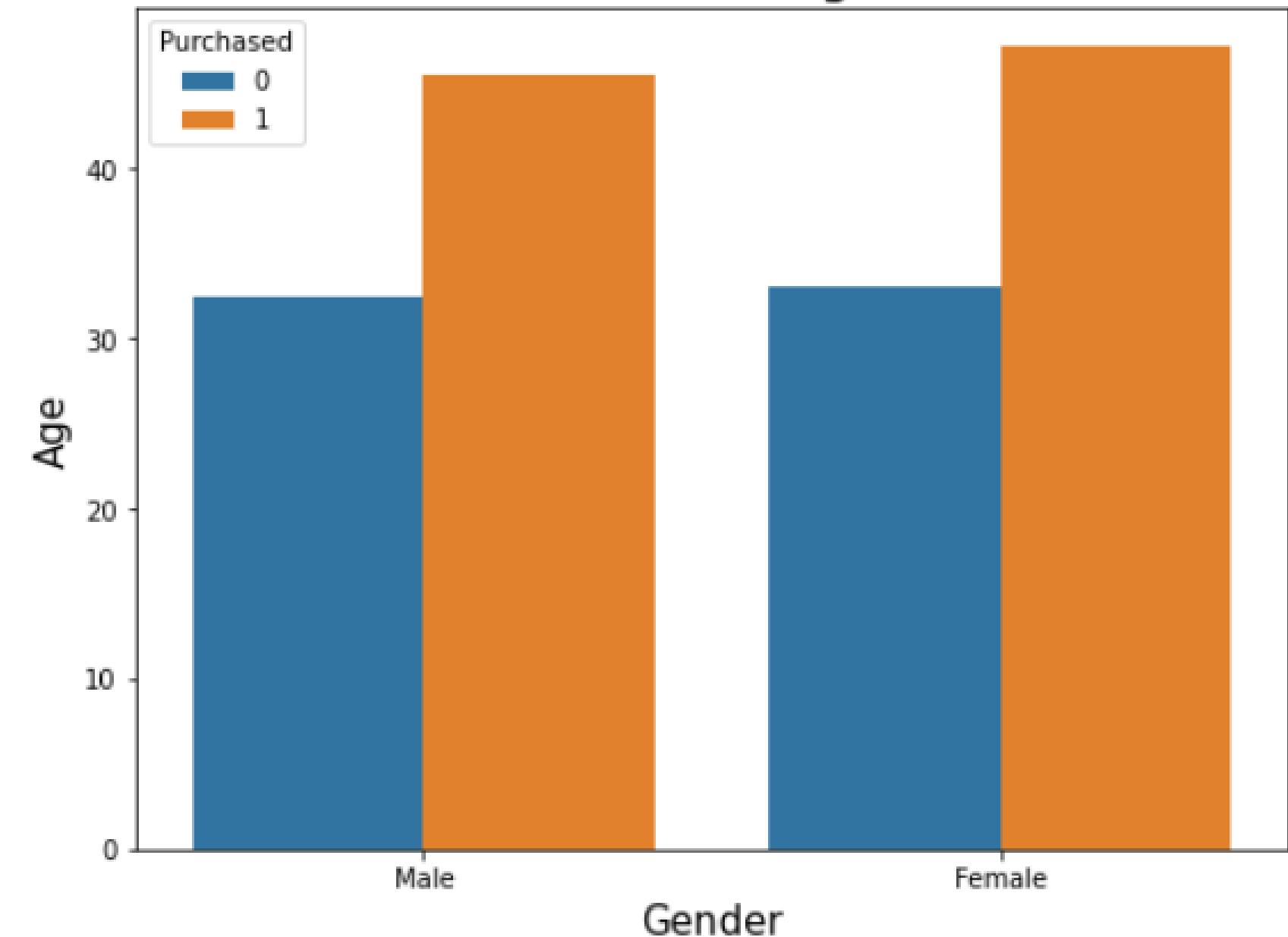
# Đồ thị phân tán thể hiện mối tương quan giữa các biến

```
plt.figure(figsize=(10,8))
plt.title('Scatter Plot of Age, EstimatedSalary 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);
```

Scatter Plot of Age, EstimatedSalary và Purchased



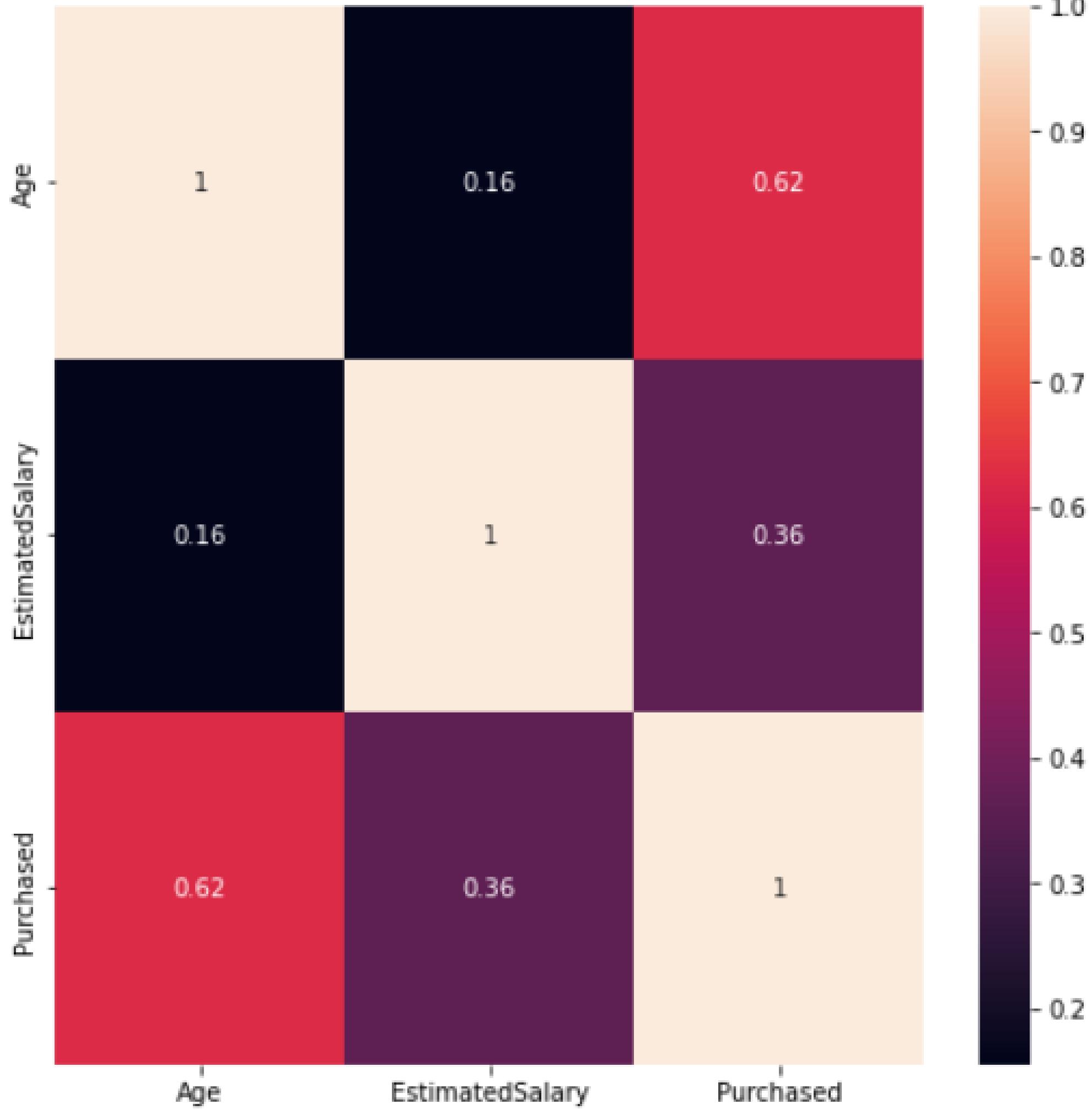
Distribution of Gender, Age và Purchased



Biến giới tính không ảnh hưởng đến Purchased

## Ma trận tương quan giữa các biến

Correlation matrix



# Tiền xử lý dữ liệu

## Chia biến

```
X = social_network_ads.iloc[:, :-1].values  
X  
  
array([[ 19, 19000],  
       [ 35, 20000],  
       [ 26, 43000],  
       [ 31, 31000],  
       [ 27, 27000],  
       [ 33, 33000],  
       [ 22, 22000],  
       [ 23, 23000],  
       [ 24, 24000],  
       [ 20, 20000],  
       [ 18, 18000],  
       [ 21, 21000],  
       [ 29, 29000],  
       [ 30, 30000],  
       [ 25, 25000],  
       [ 28, 28000],  
       [ 32, 32000],  
       [ 34, 34000],  
       [ 36, 36000],  
       [ 37, 37000],  
       [ 38, 38000],  
       [ 39, 39000],  
       [ 40, 40000],  
       [ 41, 41000],  
       [ 42, 42000],  
       [ 43, 43000],  
       [ 44, 44000],  
       [ 45, 45000],  
       [ 46, 46000],  
       [ 47, 47000],  
       [ 48, 48000],  
       [ 49, 49000],  
       [ 50, 50000],  
       [ 51, 51000],  
       [ 52, 52000],  
       [ 53, 53000],  
       [ 54, 54000],  
       [ 55, 55000],  
       [ 56, 56000],  
       [ 57, 57000],  
       [ 58, 58000],  
       [ 59, 59000],  
       [ 60, 60000],  
       [ 61, 61000],  
       [ 62, 62000],  
       [ 63, 63000],  
       [ 64, 64000],  
       [ 65, 65000],  
       [ 66, 66000],  
       [ 67, 67000],  
       [ 68, 68000],  
       [ 69, 69000],  
       [ 70, 70000],  
       [ 71, 71000],  
       [ 72, 72000],  
       [ 73, 73000],  
       [ 74, 74000],  
       [ 75, 75000],  
       [ 76, 76000],  
       [ 77, 77000],  
       [ 78, 78000],  
       [ 79, 79000],  
       [ 80, 80000],  
       [ 81, 81000],  
       [ 82, 82000],  
       [ 83, 83000],  
       [ 84, 84000],  
       [ 85, 85000],  
       [ 86, 86000],  
       [ 87, 87000],  
       [ 88, 88000],  
       [ 89, 89000],  
       [ 90, 90000],  
       [ 91, 91000],  
       [ 92, 92000],  
       [ 93, 93000],  
       [ 94, 94000],  
       [ 95, 95000],  
       [ 96, 96000],  
       [ 97, 97000],  
       [ 98, 98000],  
       [ 99, 99000],  
       [ 100, 100000],  
       [ 101, 101000],  
       [ 102, 102000],  
       [ 103, 103000],  
       [ 104, 104000],  
       [ 105, 105000],  
       [ 106, 106000],  
       [ 107, 107000],  
       [ 108, 108000],  
       [ 109, 109000],  
       [ 110, 110000],  
       [ 111, 111000],  
       [ 112, 112000],  
       [ 113, 113000],  
       [ 114, 114000],  
       [ 115, 115000],  
       [ 116, 116000],  
       [ 117, 117000],  
       [ 118, 118000],  
       [ 119, 119000],  
       [ 120, 120000],  
       [ 121, 121000],  
       [ 122, 122000],  
       [ 123, 123000],  
       [ 124, 124000],  
       [ 125, 125000],  
       [ 126, 126000],  
       [ 127, 127000],  
       [ 128, 128000],  
       [ 129, 129000],  
       [ 130, 130000],  
       [ 131, 131000],  
       [ 132, 132000],  
       [ 133, 133000],  
       [ 134, 134000],  
       [ 135, 135000],  
       [ 136, 136000],  
       [ 137, 137000],  
       [ 138, 138000],  
       [ 139, 139000],  
       [ 140, 140000],  
       [ 141, 141000],  
       [ 142, 142000],  
       [ 143, 143000],  
       [ 144, 144000],  
       [ 145, 145000],  
       [ 146, 146000],  
       [ 147, 147000],  
       [ 148, 148000],  
       [ 149, 149000],  
       [ 150, 150000],  
       [ 151, 151000],  
       [ 152, 152000],  
       [ 153, 153000],  
       [ 154, 154000],  
       [ 155, 155000],  
       [ 156, 156000],  
       [ 157, 157000],  
       [ 158, 158000],  
       [ 159, 159000],  
       [ 160, 160000],  
       [ 161, 161000],  
       [ 162, 162000],  
       [ 163, 163000],  
       [ 164, 164000],  
       [ 165, 165000],  
       [ 166, 166000],  
       [ 167, 167000],  
       [ 168, 168000],  
       [ 169, 169000],  
       [ 170, 170000],  
       [ 171, 171000],  
       [ 172, 172000],  
       [ 173, 173000],  
       [ 174, 174000],  
       [ 175, 175000],  
       [ 176, 176000],  
       [ 177, 177000],  
       [ 178, 178000],  
       [ 179, 179000],  
       [ 180, 180000],  
       [ 181, 181000],  
       [ 182, 182000],  
       [ 183, 183000],  
       [ 184, 184000],  
       [ 185, 185000],  
       [ 186, 186000],  
       [ 187, 187000],  
       [ 188, 188000],  
       [ 189, 189000],  
       [ 190, 190000],  
       [ 191, 191000],  
       [ 192, 192000],  
       [ 193, 193000],  
       [ 194, 194000],  
       [ 195, 195000],  
       [ 196, 196000],  
       [ 197, 197000],  
       [ 198, 198000],  
       [ 199, 199000],  
       [ 200, 200000],  
       [ 201, 201000],  
       [ 202, 202000],  
       [ 203, 203000],  
       [ 204, 204000],  
       [ 205, 205000],  
       [ 206, 206000],  
       [ 207, 207000],  
       [ 208, 208000],  
       [ 209, 209000],  
       [ 210, 210000],  
       [ 211, 211000],  
       [ 212, 212000],  
       [ 213, 213000],  
       [ 214, 214000],  
       [ 215, 215000],  
       [ 216, 216000],  
       [ 217, 217000],  
       [ 218, 218000],  
       [ 219, 219000],  
       [ 220, 220000],  
       [ 221, 221000],  
       [ 222, 222000],  
       [ 223, 223000],  
       [ 224, 224000],  
       [ 225, 225000],  
       [ 226, 226000],  
       [ 227, 227000],  
       [ 228, 228000],  
       [ 229, 229000],  
       [ 230, 230000],  
       [ 231, 231000],  
       [ 232, 232000],  
       [ 233, 233000],  
       [ 234, 234000],  
       [ 235, 235000],  
       [ 236, 236000],  
       [ 237, 237000],  
       [ 238, 238000],  
       [ 239, 239000],  
       [ 240, 240000],  
       [ 241, 241000],  
       [ 242, 242000],  
       [ 243, 243000],  
       [ 244, 244000],  
       [ 245, 245000],  
       [ 246, 246000],  
       [ 247, 247000],  
       [ 248, 248000],  
       [ 249, 249000],  
       [ 250, 250000],  
       [ 251, 251000],  
       [ 252, 252000],  
       [ 253, 253000],  
       [ 254, 254000],  
       [ 255, 255000],  
       [ 256, 256000],  
       [ 257, 257000],  
       [ 258, 258000],  
       [ 259, 259000],  
       [ 260, 260000],  
       [ 261, 261000],  
       [ 262, 262000],  
       [ 263, 263000],  
       [ 264, 264000],  
       [ 265, 265000],  
       [ 266, 266000],  
       [ 267, 267000],  
       [ 268, 268000],  
       [ 269, 269000],  
       [ 270, 270000],  
       [ 271, 271000],  
       [ 272, 272000],  
       [ 273, 273000],  
       [ 274, 274000],  
       [ 275, 275000],  
       [ 276, 276000],  
       [ 277, 277000],  
       [ 278, 278000],  
       [ 279, 279000],  
       [ 280, 280000],  
       [ 281, 281000],  
       [ 282, 282000],  
       [ 283, 283000],  
       [ 284, 284000],  
       [ 285, 285000],  
       [ 286, 286000],  
       [ 287, 287000],  
       [ 288, 288000],  
       [ 289, 289000],  
       [ 290, 290000],  
       [ 291, 291000],  
       [ 292, 292000],  
       [ 293, 293000],  
       [ 294, 294000],  
       [ 295, 295000],  
       [ 296, 296000],  
       [ 297, 297000],  
       [ 298, 298000],  
       [ 299, 299000],  
       [ 300, 300000],  
       [ 301, 301000],  
       [ 302, 302000],  
       [ 303, 303000],  
       [ 304, 304000],  
       [ 305, 305000],  
       [ 306, 306000],  
       [ 307, 307000],  
       [ 308, 308000],  
       [ 309, 309000],  
       [ 310, 310000],  
       [ 311, 311000],  
       [ 312, 312000],  
       [ 313, 313000],  
       [ 314, 314000],  
       [ 315, 315000],  
       [ 316, 316000],  
       [ 317, 317000],  
       [ 318, 318000],  
       [ 319, 319000],  
       [ 320, 320000],  
       [ 321, 321000],  
       [ 322, 322000],  
       [ 323, 323000],  
       [ 324, 324000],  
       [ 325, 325000],  
       [ 326, 326000],  
       [ 327, 327000],  
       [ 328, 328000],  
       [ 329, 329000],  
       [ 330, 330000],  
       [ 331, 331000],  
       [ 332, 332000],  
       [ 333, 333000],  
       [ 334, 334000],  
       [ 335, 335000],  
       [ 336, 336000],  
       [ 337, 337000],  
       [ 338, 338000],  
       [ 339, 339000],  
       [ 340, 340000],  
       [ 341, 341000],  
       [ 342, 342000],  
       [ 343, 343000],  
       [ 344, 344000],  
       [ 345, 345000],  
       [ 346, 346000],  
       [ 347, 347000],  
       [ 348, 348000],  
       [ 349, 349000],  
       [ 350, 350000],  
       [ 351, 351000],  
       [ 352, 352000],  
       [ 353, 353000],  
       [ 354, 354000],  
       [ 355, 355000],  
       [ 356, 356000],  
       [ 357, 357000],  
       [ 358, 358000],  
       [ 359, 359000],  
       [ 360, 360000],  
       [ 361, 361000],  
       [ 362, 362000],  
       [ 363, 363000],  
       [ 364, 364000],  
       [ 365, 365000],  
       [ 366, 366000],  
       [ 367, 367000],  
       [ 368, 368000],  
       [ 369, 369000],  
       [ 370, 370000],  
       [ 371, 371000],  
       [ 372, 372000],  
       [ 373, 373000],  
       [ 374, 374000],  
       [ 375, 375000],  
       [ 376, 376000],  
       [ 377, 377000],  
       [ 378, 378000],  
       [ 379, 379000],  
       [ 380, 380000],  
       [ 381, 381000],  
       [ 382, 382000],  
       [ 383, 383000],  
       [ 384, 384000],  
       [ 385, 385000],  
       [ 386, 386000],  
       [ 387, 387000],  
       [ 388, 388000],  
       [ 389, 389000],  
       [ 390, 390000],  
       [ 391, 391000],  
       [ 392, 392000],  
       [ 393, 393000],  
       [ 394, 394000],  
       [ 395, 395000],  
       [ 396, 396000],  
       [ 397, 397000],  
       [ 398, 398000],  
       [ 399, 399000],  
       [ 400, 400000],  
       [ 401, 401000],  
       [ 402, 402000],  
       [ 403, 403000],  
       [ 404, 404000],  
       [ 405, 405000],  
       [ 406, 406000],  
       [ 407, 407000],  
       [ 408, 408000],  
       [ 409, 409000],  
       [ 410, 410000],  
       [ 411, 411000],  
       [ 412, 412000],  
       [ 413, 413000],  
       [ 414, 414000],  
       [ 415, 415000],  
       [ 416, 416000],  
       [ 417, 417000],  
       [ 418, 418000],  
       [ 419, 419000],  
       [ 420, 420000],  
       [ 421, 421000],  
       [ 422, 422000],  
       [ 423, 423000],  
       [ 424, 424000],  
       [ 425, 425000],  
       [ 426, 426000],  
       [ 427, 427000],  
       [ 428, 428000],  
       [ 429, 429000],  
       [ 430, 430000],  
       [ 431, 431000],  
       [ 432, 432000],  
       [ 433, 433000],  
       [ 434, 434000],  
       [ 435, 435000],  
       [ 436, 436000],  
       [ 437, 437000],  
       [ 438, 438000],  
       [ 439, 439000],  
       [ 440, 440000],  
       [ 441, 441000],  
       [ 442, 442000],  
       [ 443, 443000],  
       [ 444, 444000],  
       [ 445, 445000],  
       [ 446, 446000],  
       [ 447, 447000],  
       [ 448, 448000],  
       [ 449, 449000],  
       [ 450, 450000],  
       [ 451, 451000],
```

# Chia tập train, test 8:2

```
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)
```

# Standard Scaler

```
#Before Standard Scaler  
x_train
```

```
[ 21, 106000 ],  
[ 19, 57000 ],  
[ 8, 72000 ],
```

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

```
#After StandardScaler  
X_train
```

```
array([[ 1.92295008e+00,   2.14601566e+00],  
       [ 2.02016082e+00,   3.78719297e-01],  
       [-1.38221530e+00,  -4.32498705e-01],  
       [-1.18779381e+00,  -1.01194013e+00],
```

# Xây dựng mô hình

Mô hình KNN khi  
chưa điều chỉnh

Độ chính xác: 0.95

```
[53] from sklearn.neighbors import KNeighborsClassifier  
     classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)  
     classifier.fit(X_train, y_train)
```

▼ KNeighborsClassifier  
KNeighborsClassifier()

```
[54] y_pred = classifier.predict(X_test)
```

\*\* evaluate our model using the confusion matrix\*\*

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

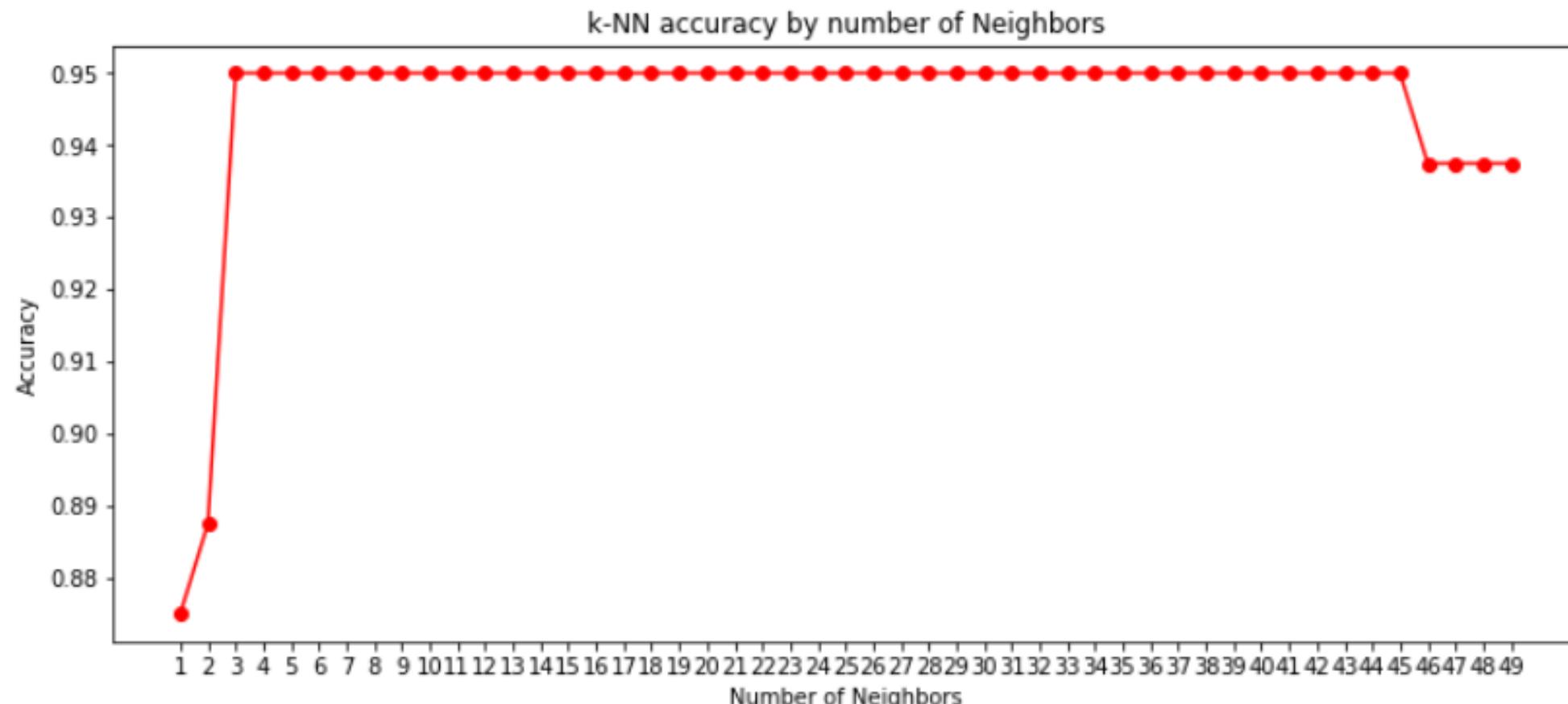
array([[55, 3],  
 [ 1, 21]])

```
[ ] ac
```

0.95

# Mô hình KNN khi k thay đổi

```
#khởi tạo chuỗi
scoreListknn = []
#Khởi tạo huấn luyện mô hình knn với vòng lặp thay đổi k từ 1->30
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=(12,5))
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))
```



## Đánh giá lại model khi sử dụng phương pháp tìm k tối ưu

```
# Đánh giá model
from sklearn.metrics import classification_report
KNclassifier = KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p = 2)
KNclassifier.fit(X_train, y_train)

preds = KNclassifier.predict(X_test)
print(classification_report(y_test, preds))
```

	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

## Phương pháp tìm k tối ưu bằng best\_params

```
#Tìm k tối ưu bằng thuộc tính best_params_
#Đầu tiên mình phải nạp hàm GridSearchCV
from sklearn.model_selection import GridSearchCV
#Tiến hành chia tập huấn luyện và tập kiểm tra
knn_grid = GridSearchCV(estimator = KNeighborsClassifier(), param_grid={'n_neighbors': np.arange(1,50)},cv=5)
knn_grid.fit(X_train,y_train)
knn_grid.best_params_
#tham khảo
https://datasciencebasic.com/?p=134
{'n_neighbors': 9}
```

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

0.95

# Mô hình KNN với lời nguyền của chiều dữ liệu

Thực hiện thay đổi chiều dữ liệu bằng phương pháp  
PCA (Principle Component Analysis)

```
from sklearn.neighbors import KNeighborsClassifier  
classifier = KNeighborsClassifier(n_neighbors = 5, metric = 'minkowski', p = 2)  
classifier.fit(X_train, y_train)
```

```
[ ] classifier  
[ ] classifier()
```

```
[ ] 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)
```

```
[ ] ac
```

0.95

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

Độ chính xác: 0.95

# Mô hình KNN với lời nguyền của chiều dữ liệu

Áp dụng PCA với số chiều dữ liệu bằng 1

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

```
▼ KNeighborsClassifier
KNeighborsClassifier(n_neighbors=9)
```

```
y_pred = classifier.predict(X_test_pca)
```

```
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)
```

KNN's Accuracy: 0.875

Độ chính xác: 0.875

## Mô hình KNN với lời nguyền của chiều dữ liệu



Kết luận: n\_features = 2 cho kết quả mô hình tốt nhất.

# Mô hình khác: Decision tree

## 3/ Huấn luyện

### 1/ Import thư viện

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

### 2/ Thiết lập Decision tree

```
[ ] from sklearn.model_selection import train_test_split
[ ] X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.2, random_state=3)
```

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

```
(280, 2)
(280,)
```

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

```
(120, 2)
(120,)
```

# Mô hình khác: Decision tree

## 4/ Chạy mô hình

```
purchasedTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
purchasedTree # it shows the default parameters
```

```
▼      DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

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

```
▼      DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

## 5/ Dự đoán

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

```
[ ] print (predTree [0:5])
print (y_testset [0:5])
```

```
[0 1 1 0 0]
[0 1 1 0 0]
```

## Mô hình khác: Decision tree

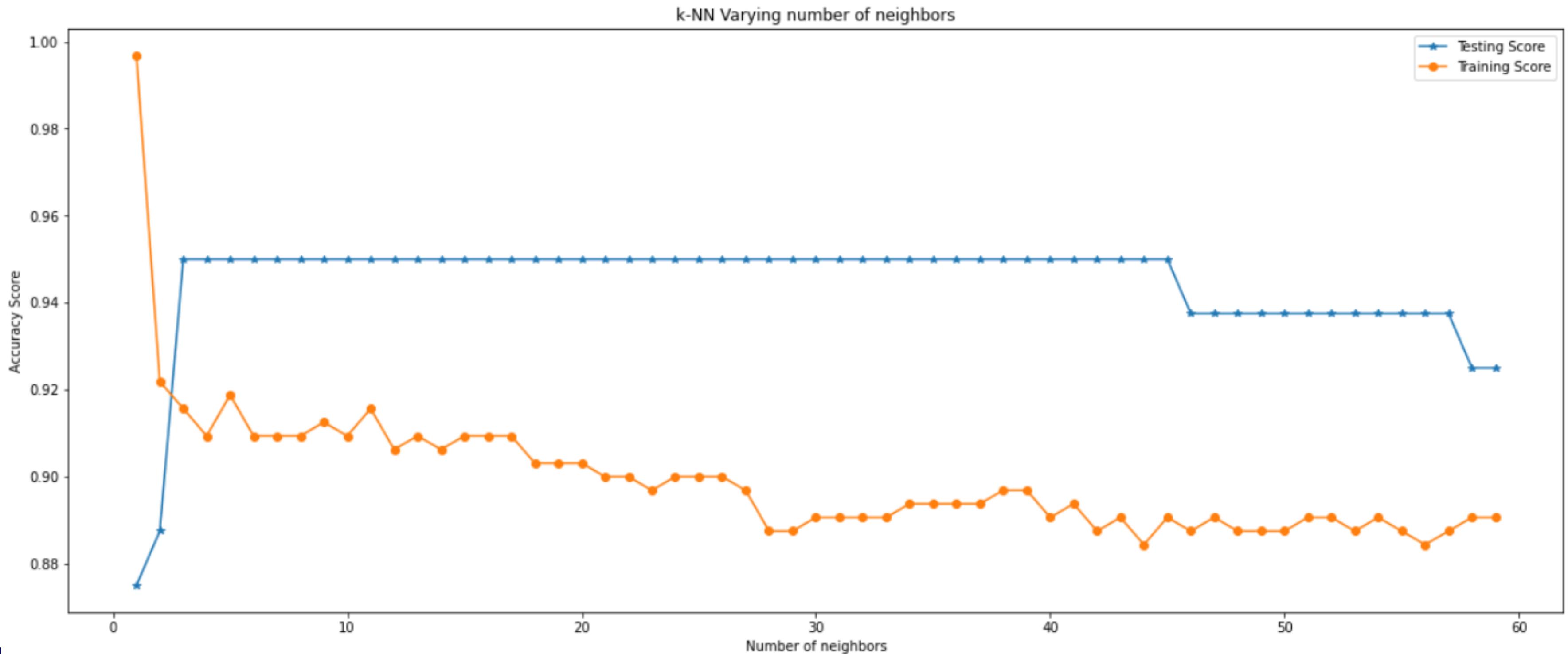
### 6/ Đánh giá

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

DecisionTrees's Accuracy: 0.925

# Đánh giá & Kết luận

Đánh giá mối tương quan của mô hình trong trường hợp số lượng láng giềng K thay đổi



# Đánh giá & Kết luận

## Kết luận:

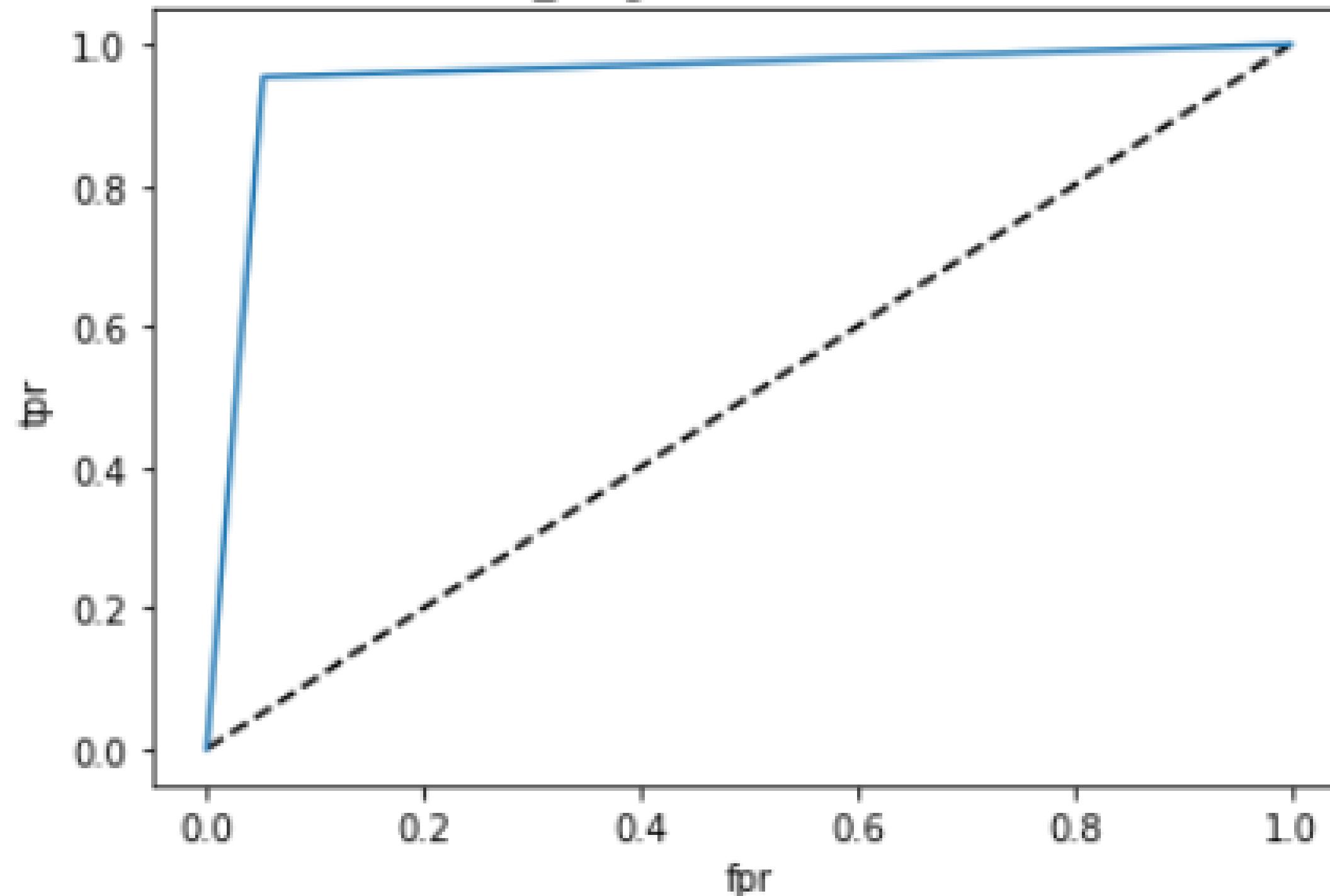
Sử dụng 1 láng giềng, mỗi điểm trong tập dữ liệu tập huấn đều có sự ảnh hưởng rõ rệt trong những dự đoán dẫn đến một dự đoán không chắc chắn.

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.

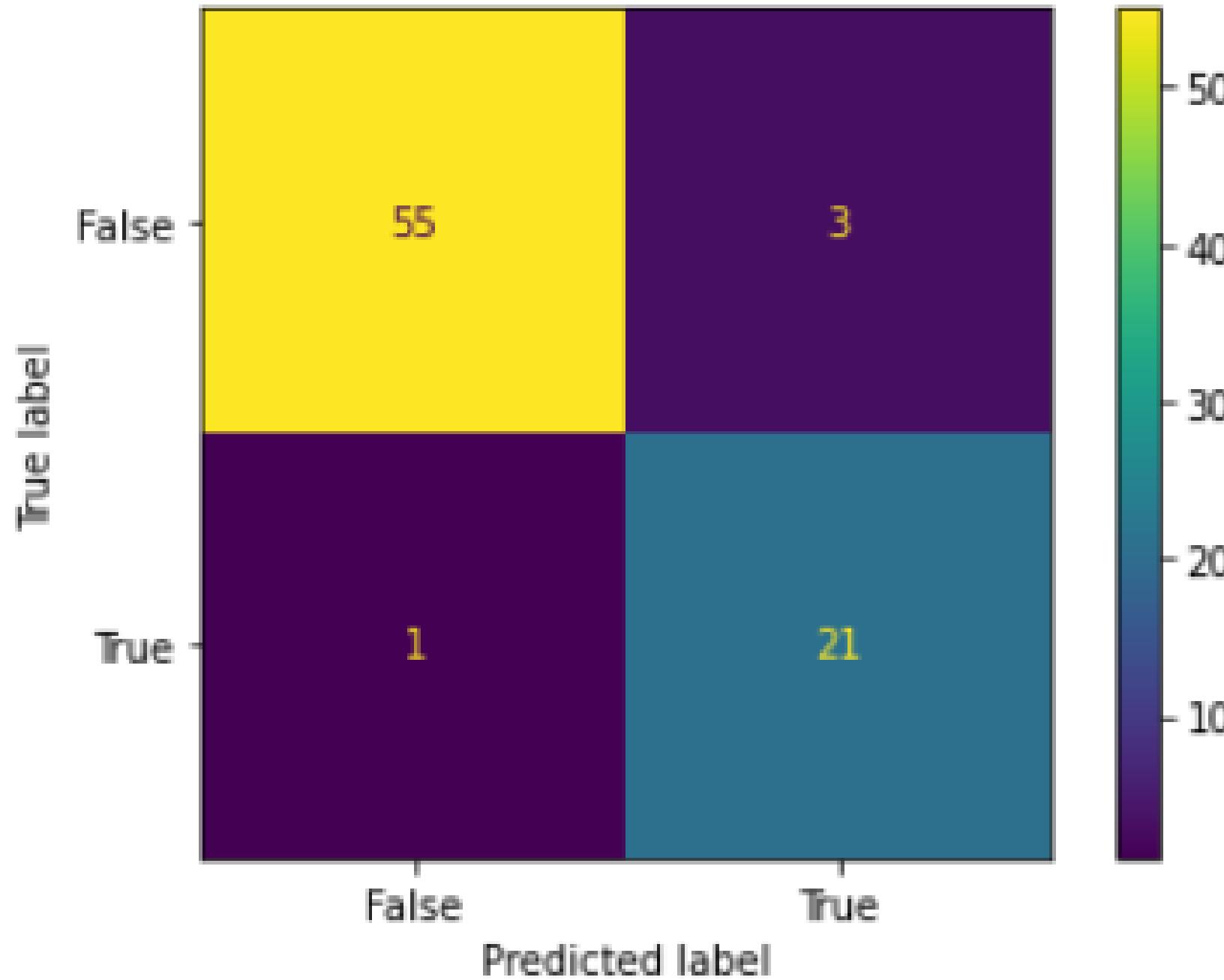
## Mô hình KNN với K tối ưu ( $n_{neighbor} = 9$ )

Knn( $n_{neighbors}=9$ ) ROC curve



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.

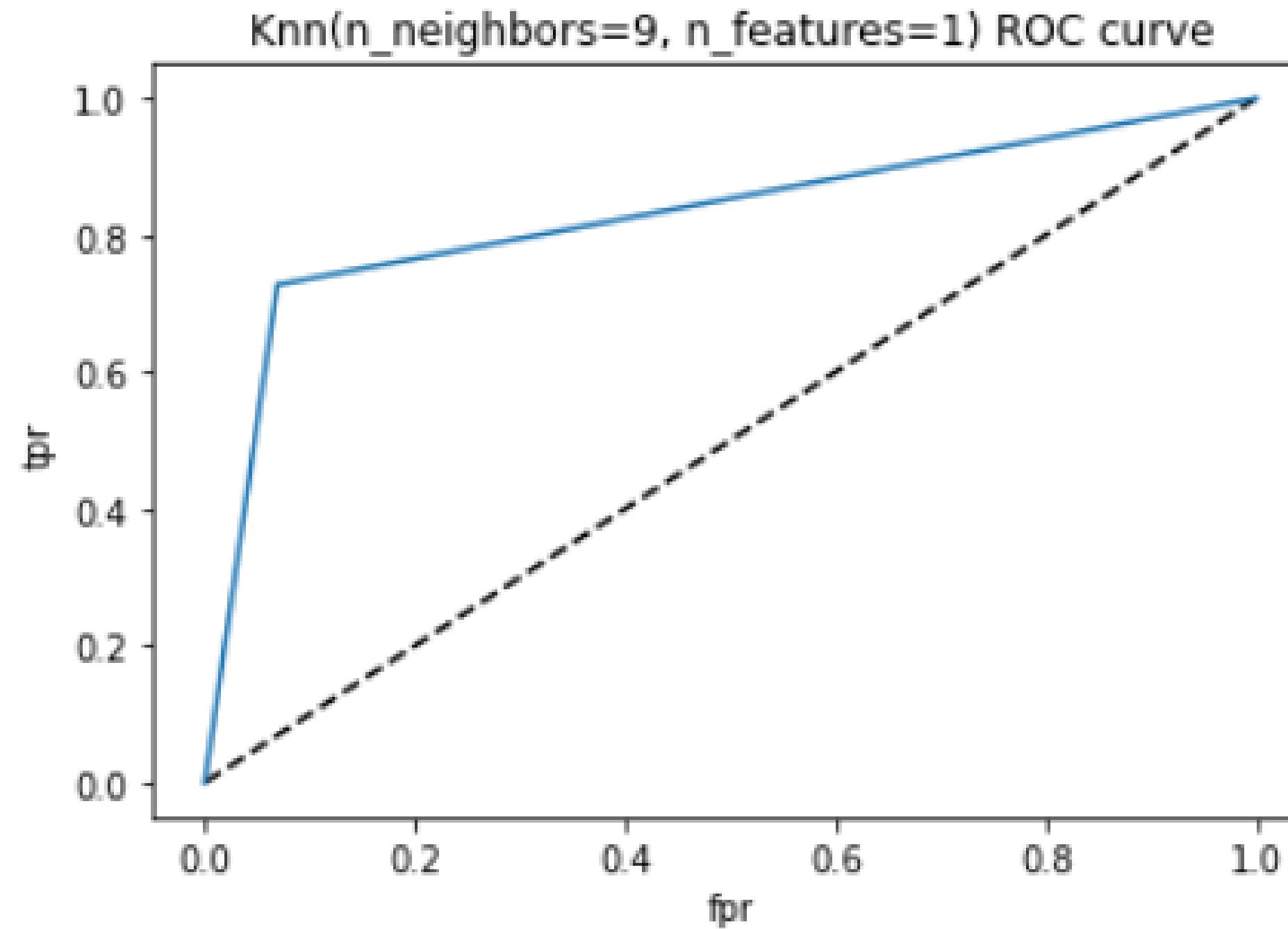
# Sử dụng ma trận nhầm lẫn dự đoán độ chính xác mô hình



Mô hình hoạt động tương đối chính xác, ít gặp phải lỗi.

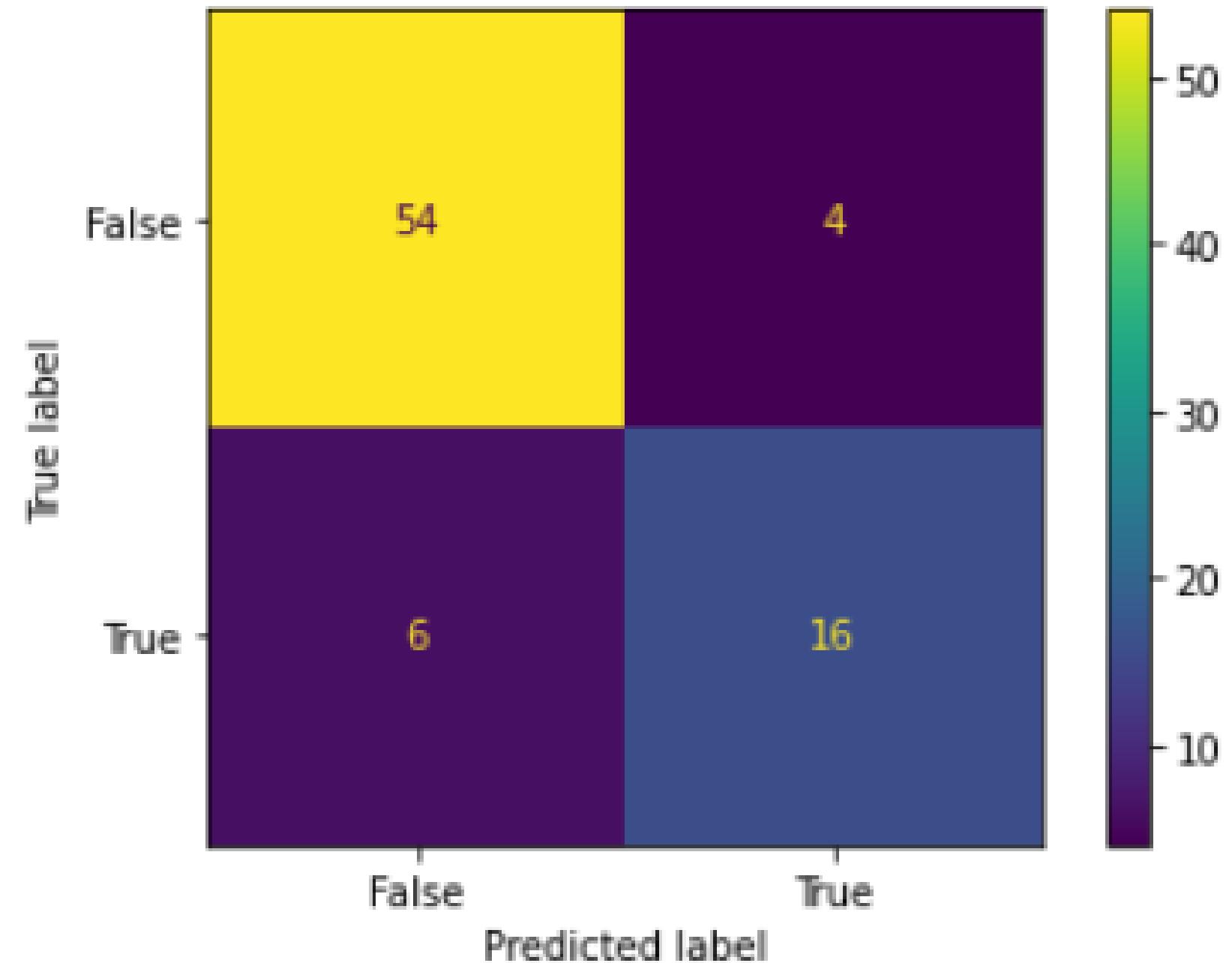
	precision	recall	f1-score	support
0	0.98	0.95	0.96	58
1	0.88	0.95	0.91	22
accuracy			0.95	80
macro avg	0.93	0.95	0.94	80
weighted avg	0.95	0.95	0.95	80

## Mô hình KNN với lời nguyền dữ liệu (`n_features = 1`)



ROC AUC của mô hình đạt gần 0.83 tương ứng với điểm fpr thấp và tpr cao ứng trên đồ thị,

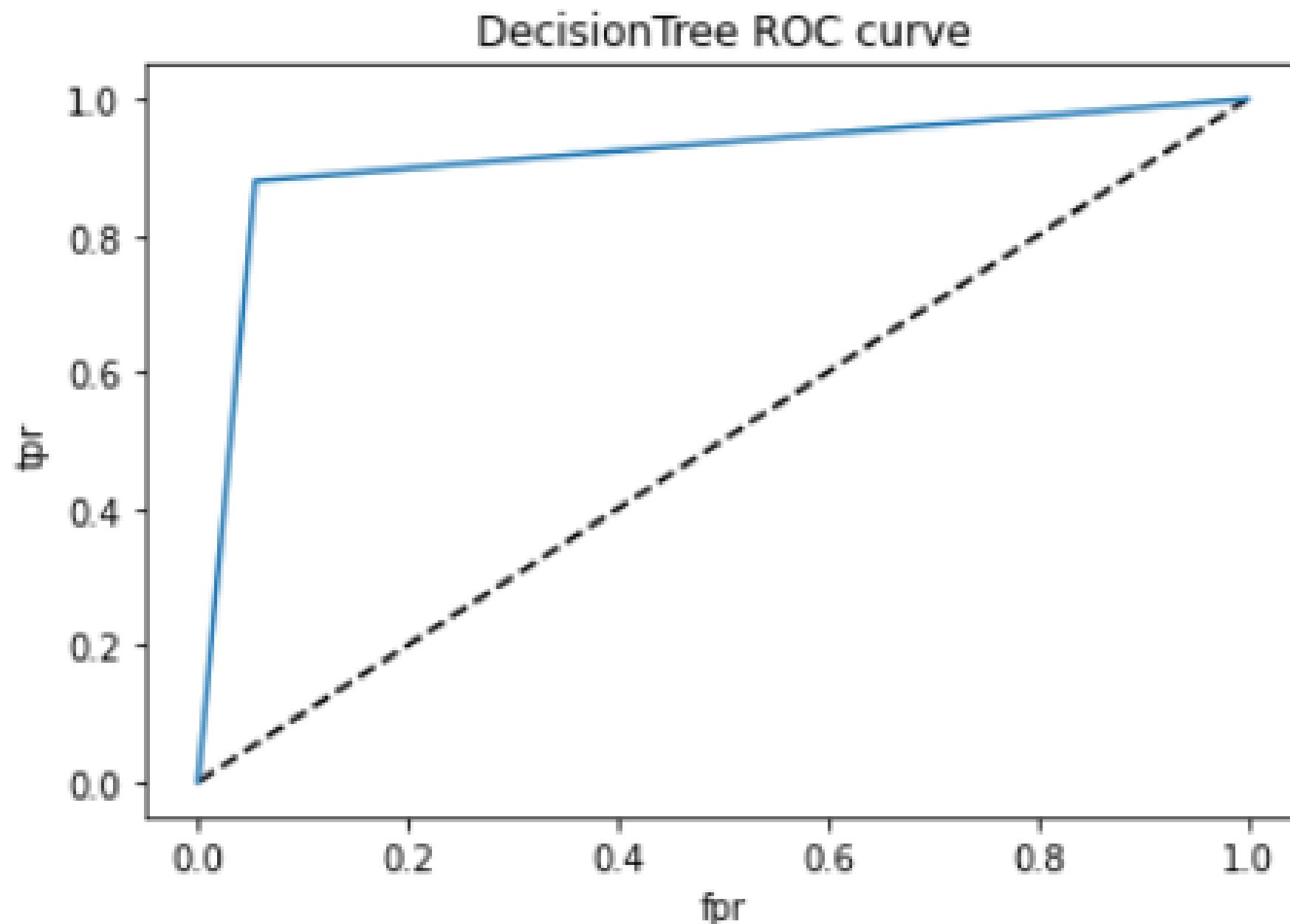
## Sử dụng ma trận nhầm lẫn dự đoán độ chính xác mô hình



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

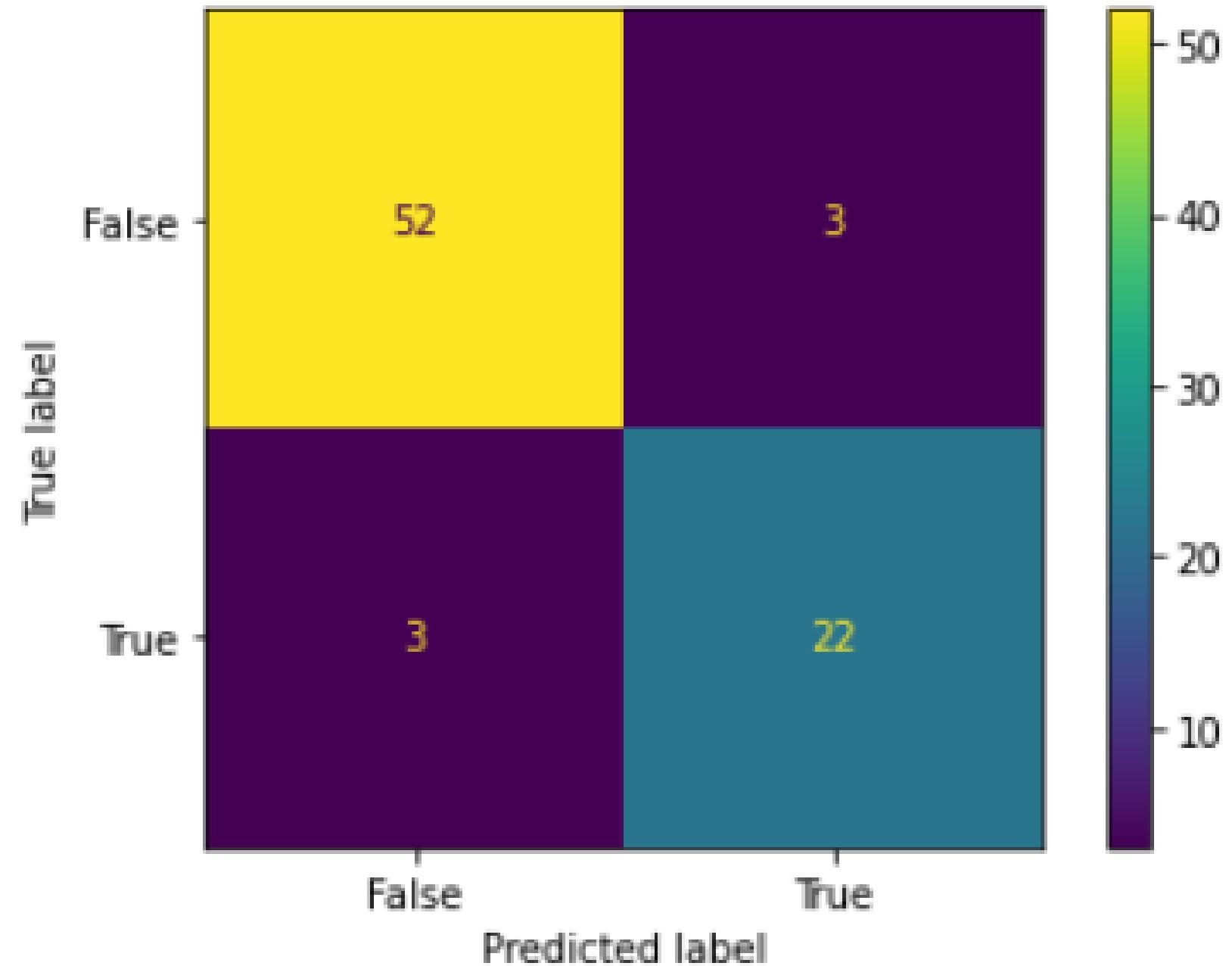
	precision	recall	f1-score	support
class 0	0.90	0.93	0.92	58
class 1	0.80	0.73	0.76	22
accuracy			0.88	80
macro avg	0.85	0.83	0.84	80
weighted avg	0.87	0.88	0.87	80

## Mô hình Decision Tree



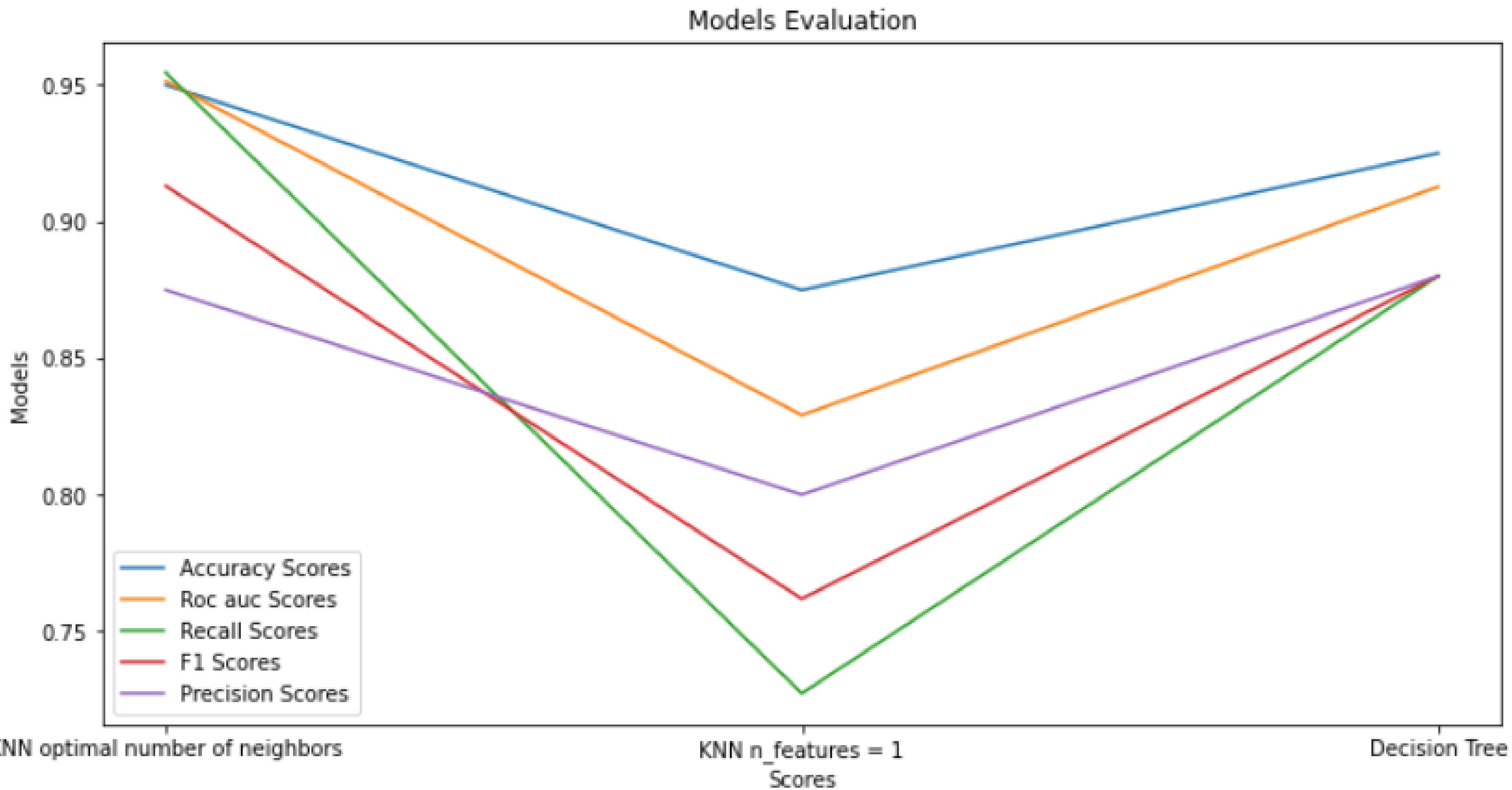
ROC AUC của mô hình đạt tầm 0.91 tương ứng với điểm fpr thấp và tpr cao trên ứng trên đồ thị => Mô hình Decision Tree hoạt động khá tốt

# Sử dụng ma trận nhầm lẫn dự đoán độ chính xác mô hình



	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

## So sánh các mô hình và lựa chọn





# THANK YOU FOR LISTENING

