Thiết kế và lựa chọn đặc trưng (Feature engineering and selection)

TS. Nguyễn Vinh Tiệp

NỘI DUNG

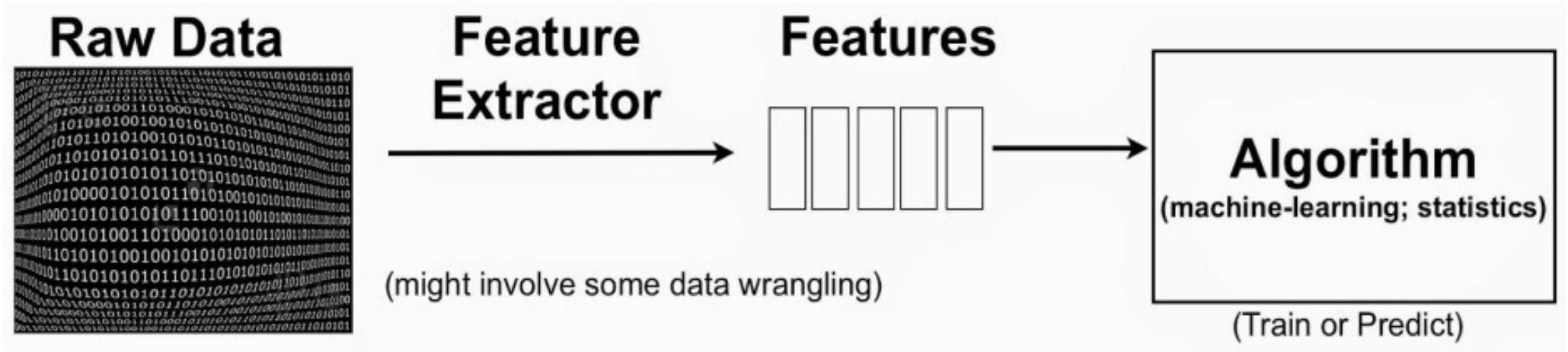
- Thiết kế đặc trưng và biến đổi dữ liệu
- Tiền xử lý dữ liệu
- Kỹ thuật thiết kế đặc trưng
- Tăng tốc độ thiết kế đặc trưng
- Kỹ thuật lựa chọn đặc trưng và công cụ

Thiết kế đặc trưng

Giới thiệu

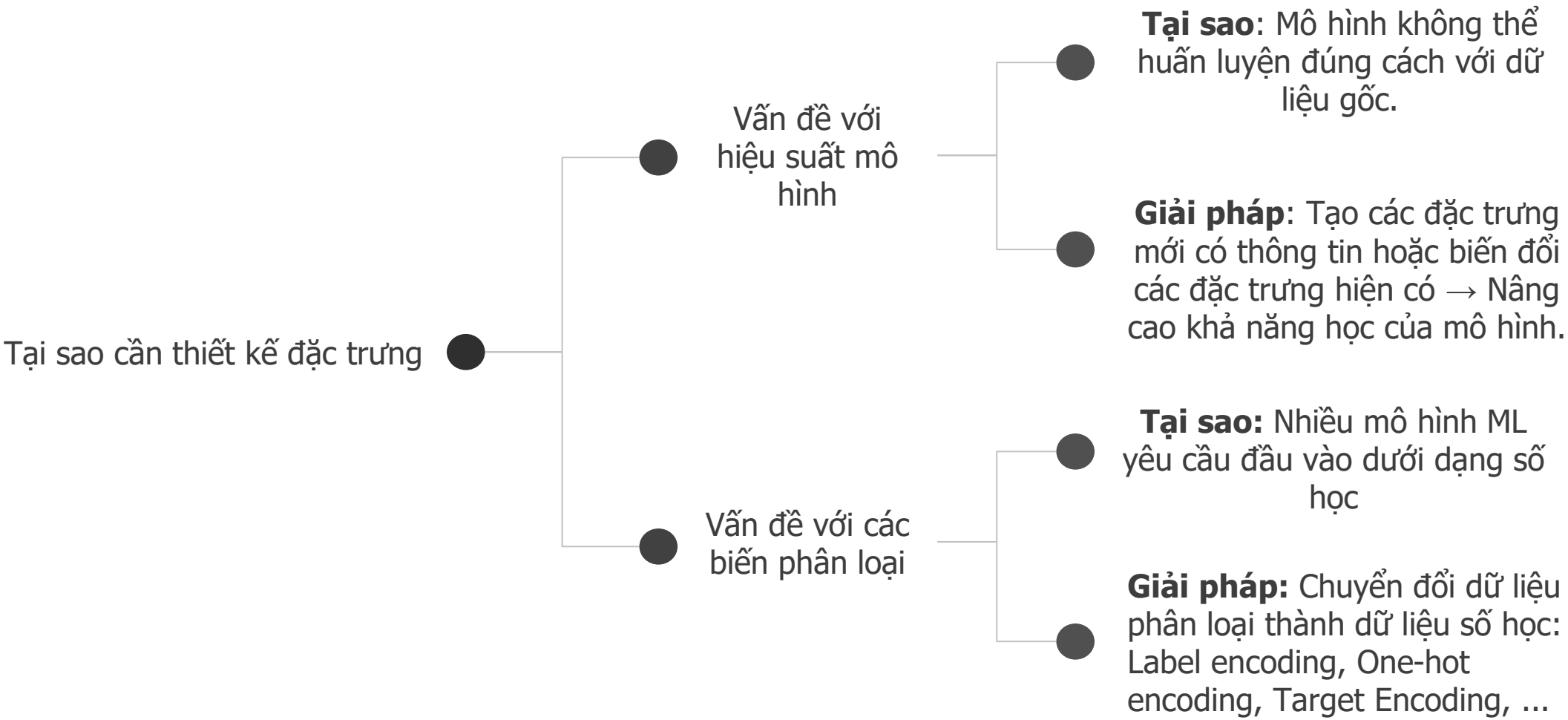
Thiết kế đặc trưng là gì

- **Thiết kế đặc trưng (Feature Engineering – Feature Extraction)** là quá trình sử dụng kiến thức về lĩnh vực để trích xuất các đặc trưng (đặc điểm, thuộc tính) từ dữ liệu gốc → cải thiện hiệu suất các mô hình máy học (ML models)

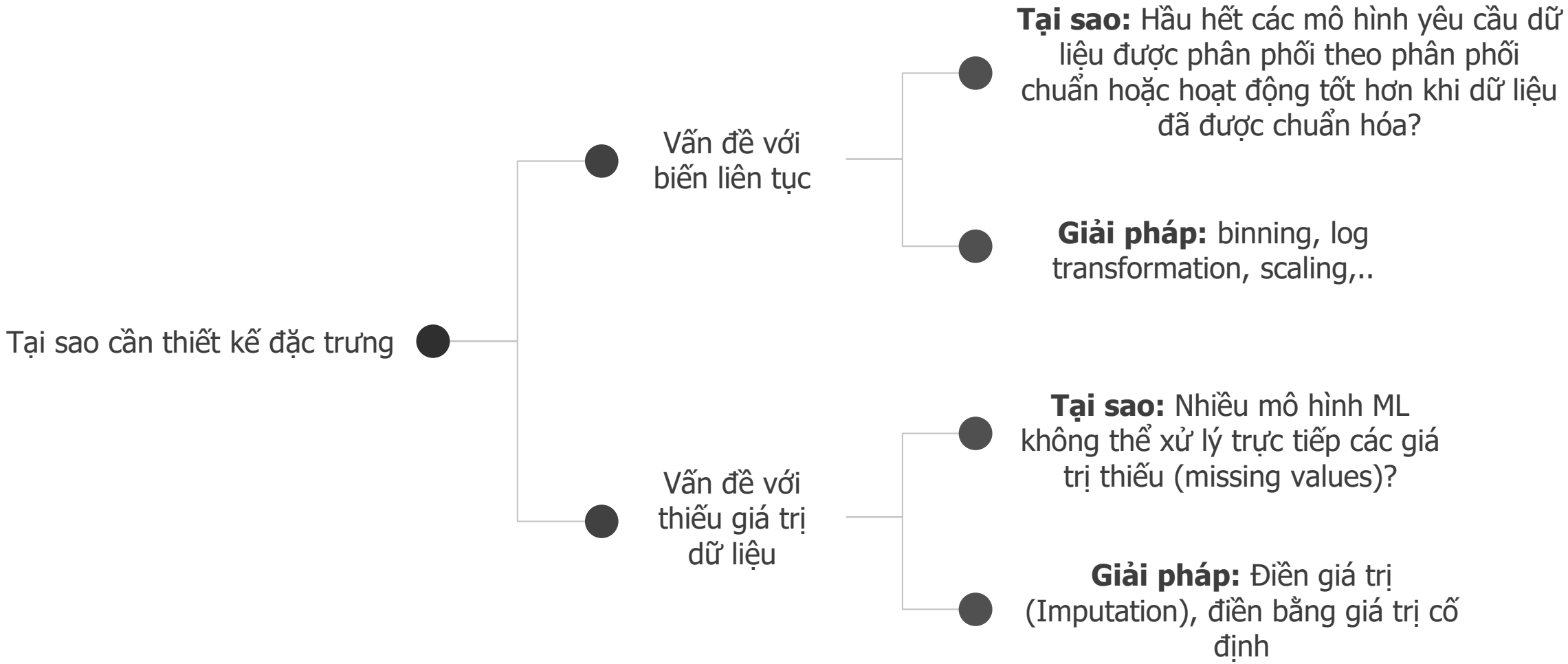


https://adataanalyst.com/machine-learning/comprehensive-guide-feature-engineering/

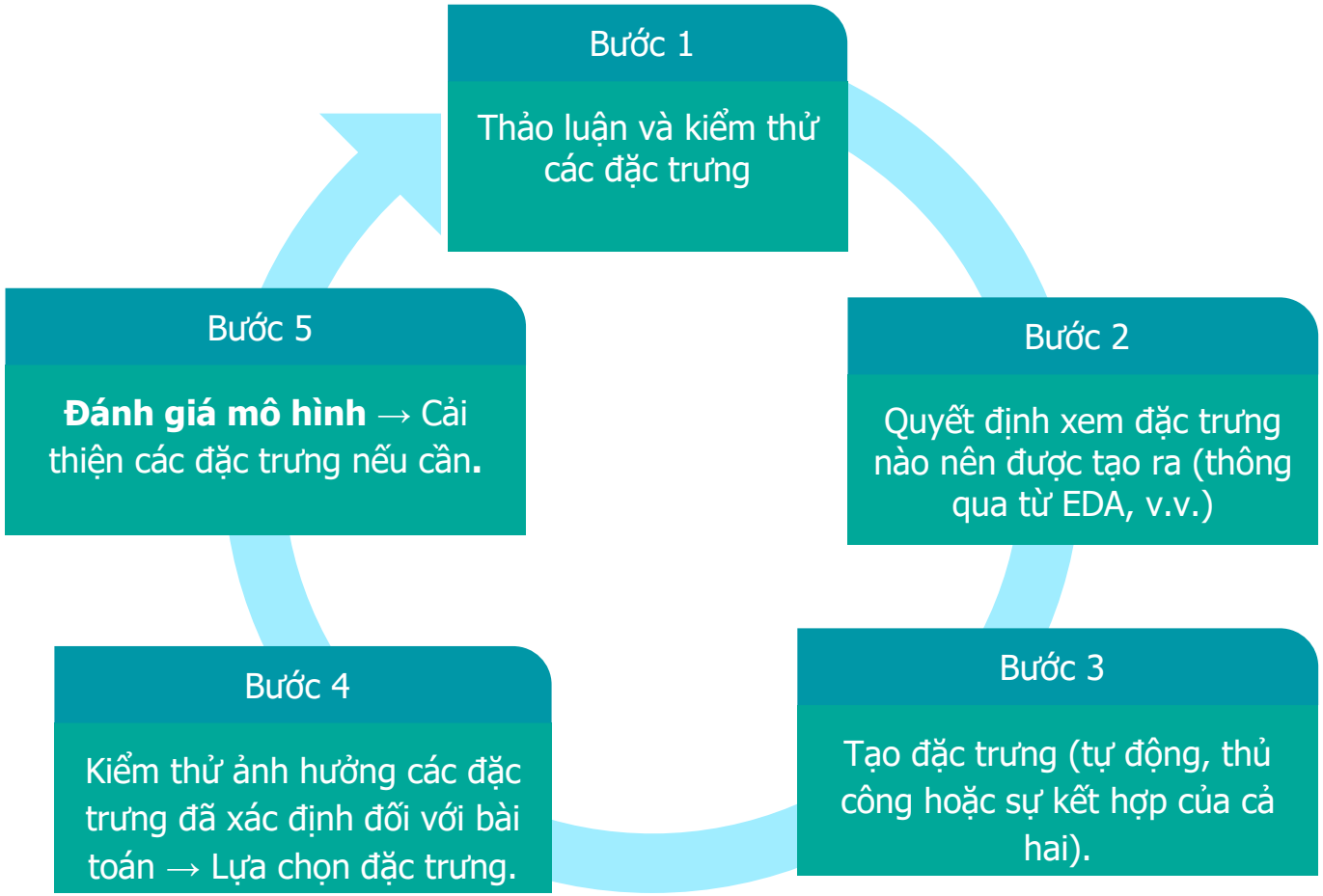
Tại sao cần thiết kế đặc trưng



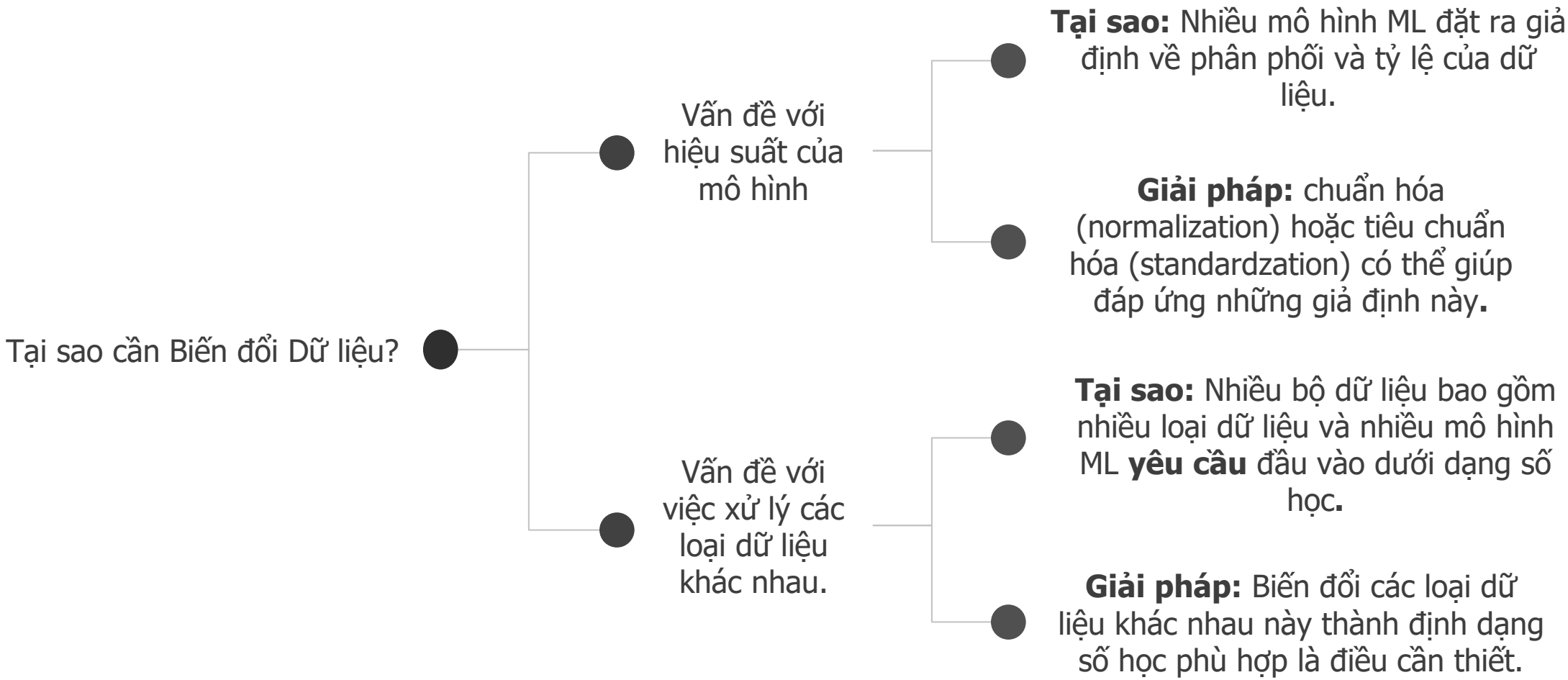
Tại sao cần thiết kế đặc trưng



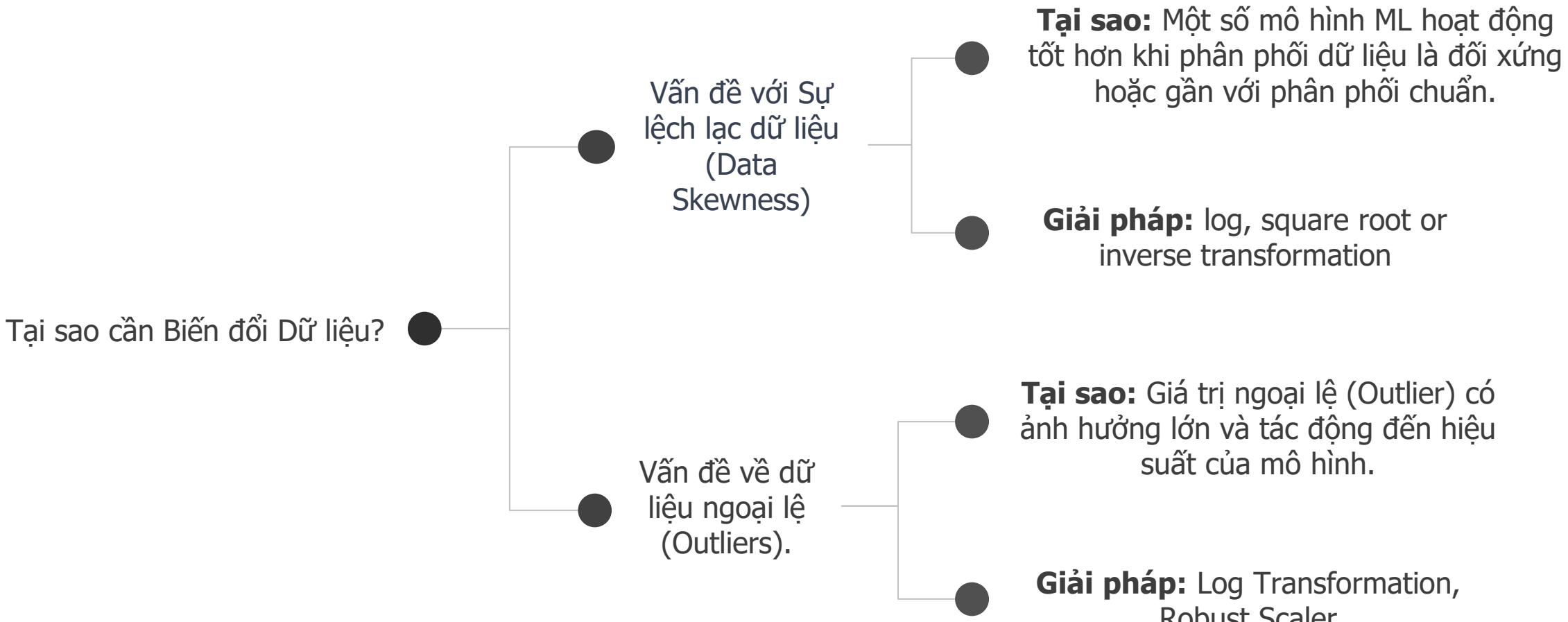
Quá trình lặp



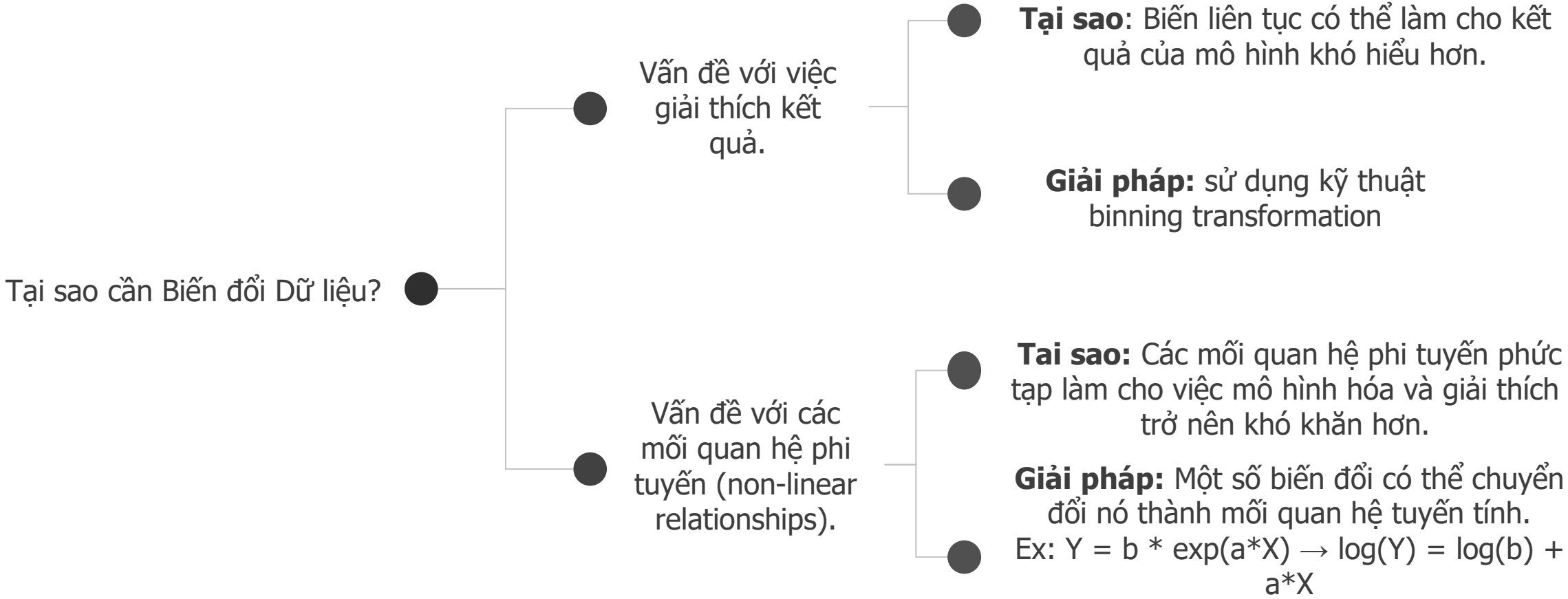
Tại sao cần Biến đổi Dữ liệu?



Tại sao cần Biến đổi Dữ liệu?

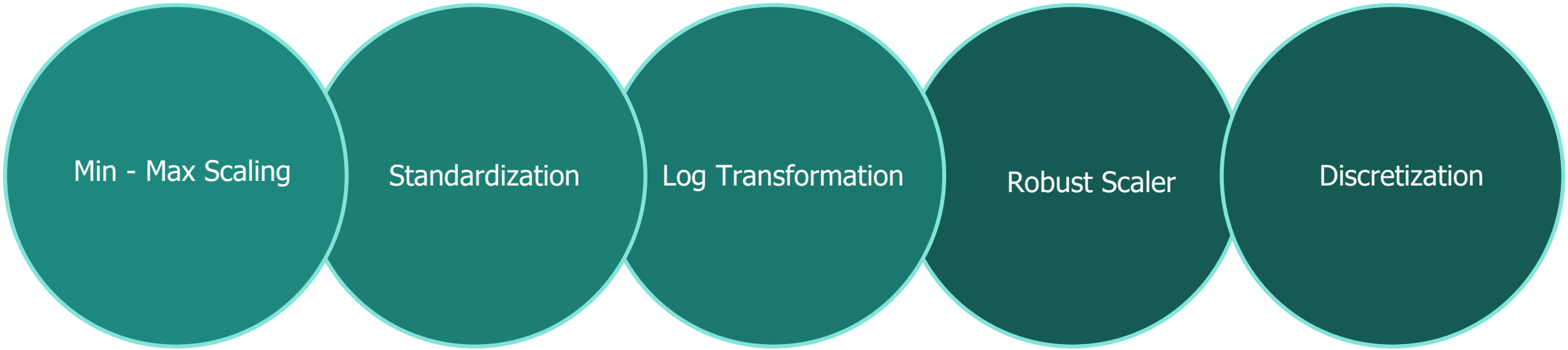


Tại sao cần Biến đổi Dữ liệu?



Biến đổi dữ liệu – Data Transformation

Dữ liệu dạng số



Kỹ thuật Min-Max Scaling

- Normalization (Min-Max Scaling): Chuẩn hóa (Normalization) tỷ lệ lại dữ liệu thành một phạm vi cố định, thường là phạm vi từ [0, 1].

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	0.0	0.0	0.0	0.0	0.0
1	109.0	9.0	25.0	549.0	44.0
2	43.0	3576.0	0.0	6715.0	49.0
3	0.0	1283.0	371.0	3329.0	193.0
4	303.0	70.0	151.0	565.0	2.0

Trước khi áp dụng Min-Max Scaling

```
X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min
```

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.007608	0.000302	0.001064	0.024500	0.001823
2	0.003001	0.119948	0.000000	0.299670	0.002030
3	0.000000	0.043035	0.015793	0.148563	0.007997
4	0.021149	0.002348	0.006428	0.025214	0.000083

Sau khi áp dụng Min-Max Scaling

Ví dụ

```
from sklearn.preprocessing import MinMaxScaler
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']


features = df[col_names]

scaler = MinMaxScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features = pd.DataFrame(features, columns = col_names)
scaled_features.head()
```

Yes

Sử dụng thư viện Scikit-lean

No

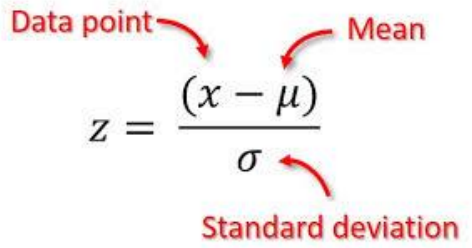
```
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
scaled_features = df.copy()

for col in col_names:
    scaled_features[col] =  (scaled_features[col] - scaled_features[col].min())/(scaled_features[col].max() + scaled_features[col].min())


scaled_features[col_names].head()
```

Kỹ thuật Standardization (Z-score scaling)

- Điều này đặc biệt hữu ích cho các thuật toán mà giả định rằng các đặc trưng tuân theo phân phối Gaussian hoặc khi các đặc trưng có các đơn vị và tỷ lệ khác nhau.



	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	0.0	0.0	0.0	0.0	0.0
1	109.0	9.0	25.0	549.0	44.0
2	43.0	3576.0	0.0	6715.0	49.0
3	0.0	1283.0	371.0	3329.0	193.0
4	303.0	70.0	151.0	565.0	2.0

Trước khi áp dụng Standard Scaling

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	-0.337025	-0.284274	-0.287317	-0.273736	-0.266098
1	-0.173528	-0.278689	-0.245971	0.209267	-0.227692
2	-0.272527	1.934922	-0.287317	5.634034	-0.223327
3	-0.337025	0.511931	0.326250	2.655075	-0.097634
4	0.117466	-0.240833	-0.037590	0.223344	-0.264352

Sau khi áp dụng Standard Scaling

Kỹ thuật Standardization (Z-score scaling)

```
from sklearn.preprocessing import StandardScaler
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']


features = df[col_names]

scaler = StandardScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features = pd.DataFrame(features, columns = col_names)
scaled_features.head()
```

Yes

Sử dụng thư viện scikit-learn

No

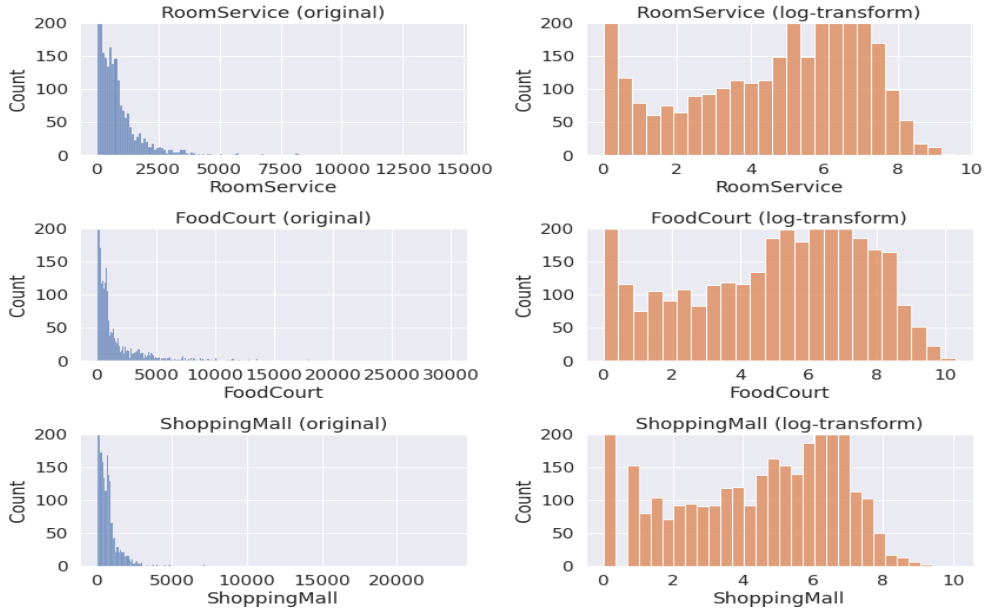
```
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
scaled_features = df.copy()

for col in col_names:
    scaled_features[col] = (scaled_features[col] - scaled_features[col].mean())/(scaled_features[col].std())


scaled_features[col_names].head()
```

Kỹ thuật Log Transformation

- Biến đổi logarithm (Log transformation) được sử dụng để giảm tác động của giá trị ngoại lệ (outliers) và xử lý phân phối dữ liệu bị lệch lạc (handle skewed).
- Bằng cách áp dụng hàm logarithm cho dữ liệu, nó nén phạm vi của dữ liệu và làm cho phân phối trở nên đối xứng hơn.



Biểu đồ histogram trước (bên trái) và sau (bên phải) biến đổi logarithm.

Discretization (Binning)

- **Rời rạc hóa** (quantization hoặc binning) là một kỹ thuật chuyển đổi biến liên tục thành các danh mục rời rạc bằng cách chia phạm vi của biến thành các khoảng (bins) hoặc khoảng cách cố định.
- Việc này có thể giúp xử lý dữ liệu nhiễu, đơn giản hóa mô hình hoặc tiết lộ các mẫu trong dữ liệu mà không rõ ràng với biến liên tục.

	PassengerId	Age
0	0001_01	39.0
1	0002_01	24.0
2	0003_01	58.0
3	0003_02	33.0
4	0004_01	16.0

Liên tục

	PassengerId	Age_group
0	0001_01	Age_31-50
1	0002_01	Age_18-25
2	0003_01	Age_51+
3	0003_02	Age_31-50
4	0004_01	Age_13-17

Rời rạc

Ví dụ

```
train['Age_group']=np.nan
train.loc[train['Age']<=12,'Age_group']='Age_0-12'
train.loc[(train['Age']>12) & (train['Age']<18),'Age_group']='Age_13-17'
train.loc[(train['Age']>=18) & (train['Age']<=25),'Age_group']='Age_18-25'
train.loc[(train['Age']>25) & (train['Age']<=30),'Age_group']='Age_26-30'
train.loc[(train['Age']>30) & (train['Age']<=50),'Age_group']='Age_31-50'
train.loc[train['Age']>50,'Age_group']='Age_51+'
```

```
# The first way
binned = pd.cut(train['Age'], bins = 6,labels = ['Age_0-12', 'Age_13-17', 'Age_18-25',
                                                 'Age_26-30', 'Age_31-50', 'Age_51+'])
# Second way
bins = pd.IntervalIndex.from_tuples([(0, 12), (12, 18), (18, 25), (25, 30), (30, 50), (50, 100)])
binned = pd.cut(train['Age'], bins = bins, labels = ['Age_0-12', 'Age_13-17', 'Age_18-25',
                                                 'Age_26-30', 'Age_31-50', 'Age_51+'])

pdcut_output = np.squeeze(np.array([list(binned)]).reshape(-1, 1))

train['Age_group'] =  pdcut_output
```

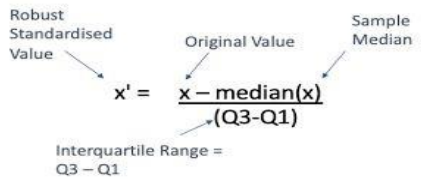
```
from sklearn.preprocessing import KBinsDiscretizer

data_age = np.array(train['Age'].fillna(0)).reshape(-1, 1)
age_group = KBinsDiscretizer(n_bins = 6, encode = "ordinal").fit_transform(data_age).squeeze()

train['Age_group'] =  age_group
```

Chuẩn hóa đặc trưng bằng Robust Scaler

- Robust scaling sử dụng giá trị trung vị và phạm vi giữa các phân vị (interquartile range) để tỷ lệ dữ liệu, làm cho nó ít nhạy cảm hơn đối với giá trị ngoại lệ so với Min-Max hoặc Z-score scaling.
- ***Điều này hữu ích khi bộ dữ liệu chứa giá trị ngoại lệ hoặc khi phân phối dữ liệu không đối xứng..***



	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	0.0	0.0	0.0	0.0	0.0
1	109.0	9.0	25.0	549.0	44.0
2	43.0	3576.0	0.0	6715.0	49.0
3	0.0	1283.0	371.0	3329.0	193.0
4	303.0	70.0	151.0	565.0	2.0

Trước Robust Scaler

	RoomService	FoodCourt	ShoppingMall	Spa	VRDeck
0	0.000000	0.000000	0.000000	0.000000	0.000000
1	2.319149	0.118421	0.925926	9.305085	0.956522
2	0.914894	47.052632	0.000000	113.813559	1.065217
3	0.000000	16.881579	13.740741	56.423729	4.195652
4	6.446809	0.921053	5.592593	9.576271	0.043478

Sau Robust Scaler

Example

```
from sklearn.preprocessing import RobustScaler
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']


features = df[col_names]

scaler = RobustScaler().fit(features.values)
features = scaler.transform(features.values)
scaled_features = pd.DataFrame(features, columns = col_names)
scaled_features.head()
```

Sử dụng scikit-learn

```
col_names = ['RoomService', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']
scaled_features = df.copy()

for col in col_names:
    q1 = scaled_features[col].quantile(0.25)
    median = scaled_features[col].quantile(0.5)
    q3 = scaled_features[col].quantile(0.75)
    iqr = q3 - q1
    scaled_features[col] = (scaled_features[col] - median)/iqr

scaled_features[col_names].head()
```

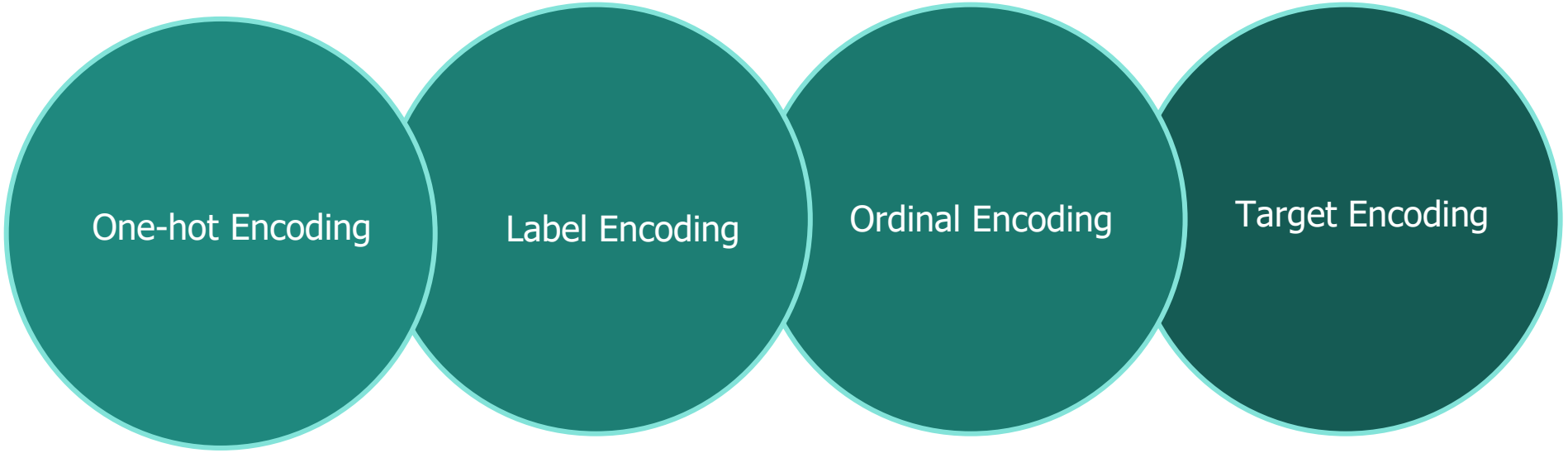
Sử dụng pandas

Kết luận

01	Min - Max Scaling	• x_norm = (x - x_min)/(x_max - x_min) • Khi đặc trưng phân bố gần đều trên một phạm vi cố định.
02	Standard Scaling	• x_norm = (x - mean(x))/std(x) • Phân phối của đặc trưng không chứa các giá trị ngoại lệ cực đoan, giả định rằng đặc trưng tuân theo phân phối Gaussian hoặc có các đơn vị và tỷ lệ khác nhau.
03	Log Transformation	• x_norm = log(x) or x_norm = log(1+x) • Giảm tác động của các giá trị ngoại lệ và xử lý các phân phối dữ liệu bị lệch.
04	Binning	• Chuyển đổi biến liên tục thành các danh mục rời rạc bằng cách chia phạm vi của biến thành các khoảng (bins). • Xử lý dữ liệu nhiễu, đơn giản hóa mô hình.
05	Robust Scaling	• x_norm = (x - median(x))/IQR • Làm cho nó ít nhạy cảm hơn đối với các giá trị ngoại lệ so với việc tỷ lệ Min-Max hoặc Z-score. • Hữu ích khi bộ dữ liệu chứa các giá trị ngoại lệ hoặc phân phối dữ liệu không đối xứng.

Chuyển hóa dữ liệu

Kiểu dữ liệu danh mục (Categorical data)



Kỹ thuật One-hot Encoding

- Đối với mỗi giá trị, sẽ được tạo một cột nhị phân mới, với giá trị 1 và 0 thể hiện sự có mặt hoặc vắng mặt của danh mục đó trong dữ liệu gốc.

	PassengerId	HomePlanet
0	0001_01	Europa
1	0002_01	Earth
2	0003_01	Europa
3	0003_02	Europa
4	0004_01	Earth

```
pd.get_dummies(df['HomePlanet'], prefix = 'is')
```

	is_Earth	is_Europa	is_Mars
0	0	1	0
1	1	0	0
2	0	1	0
3	0	1	0
4	1	0	0

Kỹ thuật Ordinal Encoding

- Mã hóa Ordinal (Ordinal encoding) gán các giá trị số nguyên cho biến phân loại dựa trên thứ tự hoặc xếp hạng của chúng..
- Nó phù hợp cho dữ liệu thứ tự (ordinal data), trong đó có sự thứ tự tự nhiên giữa các danh mục.

```
age_group_mapping = {'Age_0-12': 1, 'Age_13-17': 2, 'Age_18-25': 3,
                     'Age_26-30': 4, 'Age_31-50': 5, 'Age_51+': 6}

train['Age_group_encode'] = train['Age_group'].map(age_group_mapping)
train[['Age', 'Age_group', 'Age_group_encode']].head()
```

	Age	Age_group	Age_group_encode
0	39.0	Age_31-50	5.0
1	24.0	Age_18-25	3.0
2	58.0	Age_51+	6.0
3	33.0	Age_31-50	5.0
4	16.0	Age_13-17	2.0

Kỹ thuật Label Encoding

- Mã hóa nhãn (Label encoding) là một phương pháp khác để chuyển đổi biến phân loại thành giá trị số học bằng cách gán một số nguyên duy nhất cho mỗi loại.

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
le.fit(train['HomePlanet'].astype(str).fillna('Unknow').tolist() +
                  test['HomePlanet'].astype(str).fillna('Unknow').tolist())

train['HomePlanet_encode'] = le.transform(train['HomePlanet'])
test['HomePlanet_encode'] = le.transform(test['HomePlanet'])

train[['PassengerId', 'HomePlanet', 'HomePlanet_encode']].head()
```

	PassengerId	HomePlanet	HomePlanet_encode
0	0001_01	Europa	1
1	0002_01	Earth	0
2	0003_01	Europa	1
3	0003_02	Europa	1
4	0004_01	Earth	0

Use scikit-learn

```
train['HomePlanet_enc'],_ = train['HomePlanet'].factorize()
train[['PassengerId', 'HomePlanet', 'HomePlanet_enc']].head()
```

	PassengerId	HomePlanet	HomePlanet_enc
0	0001_01	Europa	0
1	0002_01	Earth	1
2	0003_01	Europa	0
3	0003_02	Europa	0
4	0004_01	Earth	1

Use pandas

Kỹ thuật Target Encoding

- Mã hóa mục tiêu (Target encoding) là loại mã hóa thay thế các nhãn của một đặc trưng bằng một số được tính từ mục tiêu (target).
- Phương pháp
  - Một phiên bản đơn giản và hiệu quả là áp dụng một phép tổng hợp theo nhóm như trung bình (mean). Xem example
  - Mã hóa mục tiêu với kỹ thuật làm mịn (smoothing). Xem example
  - Mã hóa mục tiêu Bayesian phân cấp (Hierarchical Bayesian Target Encoding). Xem example

Kỹ thuật Simple Target Encoding (Mean encoding)

- Khi áp dụng cho mục tiêu nhị phân, phương pháp này còn được gọi là "bin counting”.
- Các tên gọi khác mà bạn có thể gặp bao gồm: mã hóa xác suất (likelihood encoding), mã hóa tác động (impact encoding) và mã hóa "leave-one-out".

```
train["HomePlanet_target_en"] = train.groupby("HomePlanet")["Transported"].transform("mean")

train[["HomePlanet", "Transported", "HomePlanet_target_en"]].head(10)
```

	HomePlanet	Transported	HomePlanet_target_en
0	Europa	False	0.658846
1	Earth	True	0.423946
2	Europa	False	0.658846
3	Europa	False	0.658846
4	Earth	True	0.423946
5	Earth	True	0.423946

Target encoding with smoothing

Tại sao sử dụng mã hóa mục tiêu với kỹ thuật làm mịn?

Danh mục không xác đinh

Tạo ra một rủi ro đặc biệt về việc overfitting, điều này có nghĩa là chúng cần được huấn luyện trên một phần dữ liệu "mã hóa" độc lập.

+

Danh mục hiếm

Các giá trị tính bằng phép tổng hợp nhóm có thể không đại diện cho mẫu nào chúng ta có thể gặp trong tương lai → có thể làm tăng khả năng overfitting.

→

Target encoding with smoothing

Ý tưởng là kết hợp giá trị trung bình trong danh mục với giá trị trung bình tổng thể. Các danh mục hiếm thường có trọng số thấp hơn đối với giá trị trung bình của chúng, trong khi các danh mục bị thiếu chỉ đơn giản được gán giá trị trung bình tổng thể.

Mã giả - Pseudo code

In pseudocode:

```
encoding = weight * in_category + (1 - weight) * overall
```

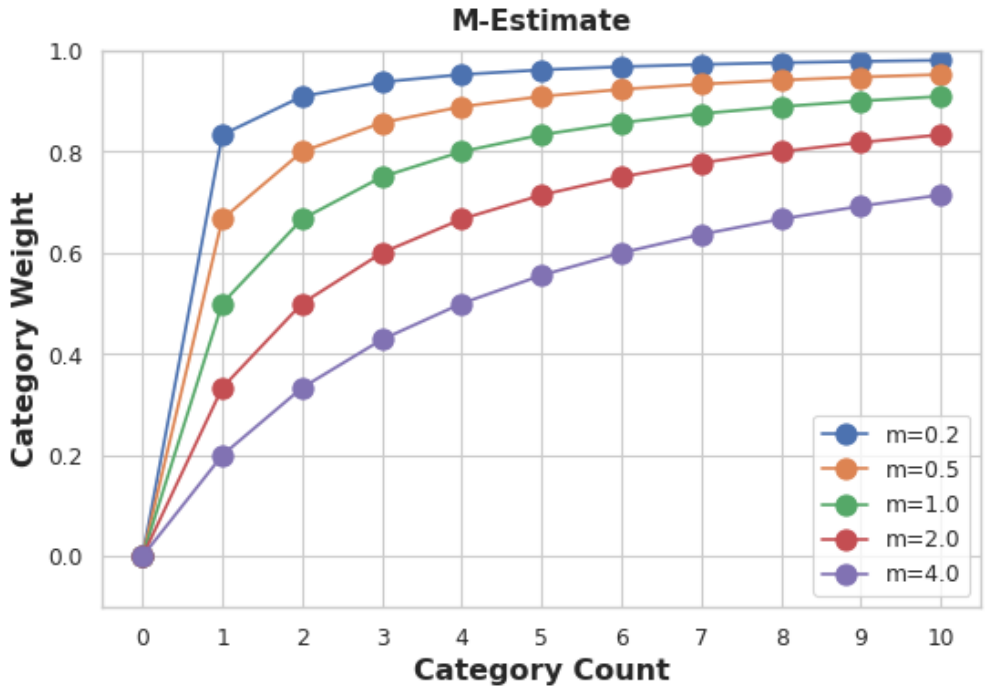
where `weight` is a value between 0 and 1 calculated from the category frequency.

An easy way to determine the value for `weight` is to compute an **m-estimate**:

```
weight = n / (n + m)
```

where `n` is the total number of times that category occurs in the data. The parameter `m` determines the "smoothing factor". Larger values of `m` put more weight on the overall estimate.

https://www.kaggle.com/code/ryanholbrook/target-encoding



https://www.kaggle.com/code/ryanholbrook/target-encoding

HỎI ĐÁP