CS116 – LẬP TRÌNH PYTHON CHO MÁY HỌC

Bài 07
LỰA CHỌN ĐẶC TRƯNG

TS. Nguyễn Vinh Tiệp

NỘI DUNG

1. Tại sao nên chọn đặc trưng?
2. Kỹ thuật lựa chọn đặc trưng
3. Công cụ lựa chọn đặc trưng

Tại sao nên chọn đặc trưng?

Tại sao lựa chọn đặc trưng ?

- Vấn đề với hiệu suất mô hình
  - **Lý do**: Noise và distraction gây ra bởi các đặc trưng không liên quan hoặc dư thừa
  - **Giải pháp:** Chỉ **chọn** các đặc trưng phù hợp nhất → **giảm** noise, distraction → **cải thiện** hiệu suất của mô hình, khái quát hóa tốt hơn
- Vấn đề với overfitting
  - **Lý do:** Mô hình quá phức tạp, hấp thụ **noise** trong dữ liệu đào tạo hơn là mô hình cơ bản
  - **Giải pháp: Loại bỏ** các đặc trưng không liên quan hoặc dư thừa → độ phức tạp của mô hình được **giảm** → tránh over-fitting

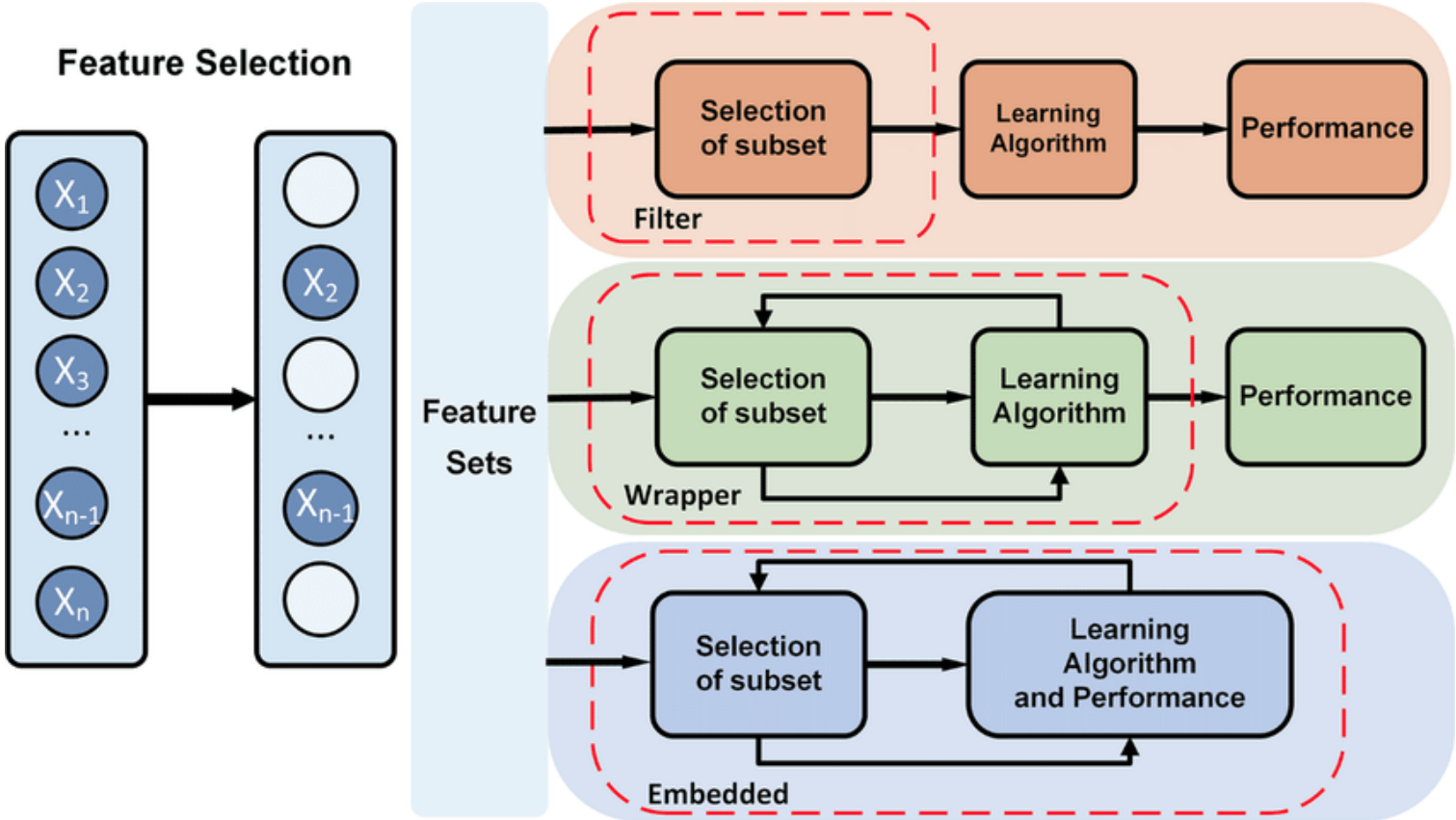
Tại sao nên chọn đặc trưng?

- Tại sao lựa chọn đặc trưng?
  - Vấn đề với thời gian đào tạo và chi phí tính toán
    - **Lý do:** Nhiều đặc trưng hơn, mô hình phức tạp → tốn kém về mặt tính toán và tốn thời gian
    - **Giải pháp:** Chỉ chọn các đặc trưng quan trọng nhất → thời gian đào tạo có thể giảm đáng kể
  - Vấn đề với khả năng diễn giải mô hình
    - **Lý do:** Các mô hình có quá nhiều đặc trưng thường khó diễn giải và giải thích hơn
    - **Giải pháp:** Chỉ sử dụng hầu hết các đặc trưng quan trọng → dễ dàng hơn để giải thích, giải thích hiểu việc ra quyết định của mô hình (ví dụ: tài chính, chăm sóc sức khỏe, lĩnh vực pháp lý)

Kỹ thuật lựa chọn đặc trưng

01	**Phương pháp Filter**	• Correlation coefficient: Pearson,.. • Variance Threshold • Missing value ratio; Mutual Information
02	**Phương pháp Wrapper**	• Forward Selection • Backward Elimination • Recursive Feature Elimination (RFE)
02	**Phương pháp Embedded**	• LASSO, Ridge Regression, Elastic Net • Tree-based: Random Forest, GBM
04	**Giảm chiều**	• Component/Factor based: Factor Analysis, PCA, ICA • Projection based: t-SNE, UMAP

Kỹ thuật lựa chọn đặc trưng



https://www.researchgate.net/publication/345579532_Computational_Diagnostic_Techniques_for_Electrocardiogram_Signal_Analysis

Phương pháp lọc

- Lựa chọn đặc trưng dựa trên bộ lọc áp dụng một chỉ số đã chọn để tìm các thuộc tính không liên quan và lọc ra dữ liệu dư thừa

Tương quan Pearson

```
# Threshold for removing correlated variables
threshold = 0.9

# Absolute value correlation matrix
corr_matrix = train.corr().abs()
corr_matrix.head()
```

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY
CNT_CHILDREN	1.000000	0.055960	0.036836	0.055732
AMT_INCOME_TOTAL	0.055960	1.000000	0.429317	0.491143
AMT_CREDIT	0.036836	0.429317	1.000000	0.797209
AMT_ANNUITY	0.055732	0.491143	0.797209	1.000000
AMT_GOODS_PRICE	0.035851	0.439981	0.986046	0.799121

```
# Upper triangle of correlations
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(np.bool))
upper.head()
```

	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE
CNT_CHILDREN	NaN	0.05596	0.036836	0.055732	0.035851
AMT_INCOME_TOTAL	NaN	NaN	0.429317	0.491143	0.439981
AMT_CREDIT	NaN	NaN	NaN	0.797209	0.986046
AMT_ANNUITY	NaN	NaN	NaN	NaN	0.799121
AMT_GOODS_PRICE	NaN	NaN	NaN	NaN	NaN

```
# Select columns with correlations above threshold
to_drop = [column for column in upper.columns if any(upper[column] > threshold)]

print('There are %d columns to remove.' % (len(to_drop)))
```

```
There are 584 columns to remove.
```

Ngưỡng phương sai

- Bộ chọn đặc trưng loại bỏ tất cả các đặc trưng có phương sai thấp

```
#------------ var -------------
from sklearn.feature_selection import VarianceThreshold
var_thresh = VarianceThreshold(0.8)
var_thresh.fit(x_train)
x_train = x_train.loc[:,var_thresh.variances_ > 0.8]
x_valid = x_valid.loc[:,var_thresh.variances_ > 0.8]
x_test  = x_test.loc[:,var_thresh.variances_ > 0.8]
```

Thiếu tỷ lệ giá trị

- Một lựa chọn tương đối đơn giản về lựa chọn đặc trưng, nếu bất kỳ cột nào có giá trị thiếu lớn hơn 75% (80%, 90%,..), chúng sẽ bị xóa.

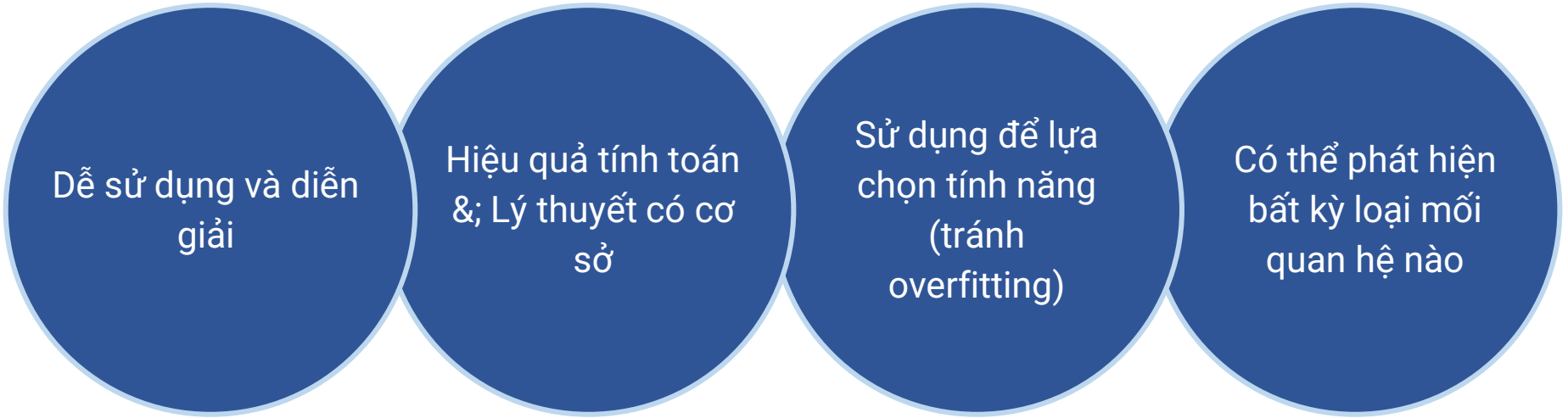
```
# Train missing values (in percent)
train_missing = (train.isnull().sum() / len(train)).sort_values(ascending = False)
train_missing.head()
```

```
# Identify missing values above threshold
train_missing = train_missing.index[train_missing > 0.75]
test_missing = test_missing.index[test_missing > 0.75]

all_missing = list(set(set(train_missing) | set(test_missing)))
print('There are %d columns with more than 75%% missing values' % len(all_missing))
```

```
There are 19 columns with more than 75% missing values
```

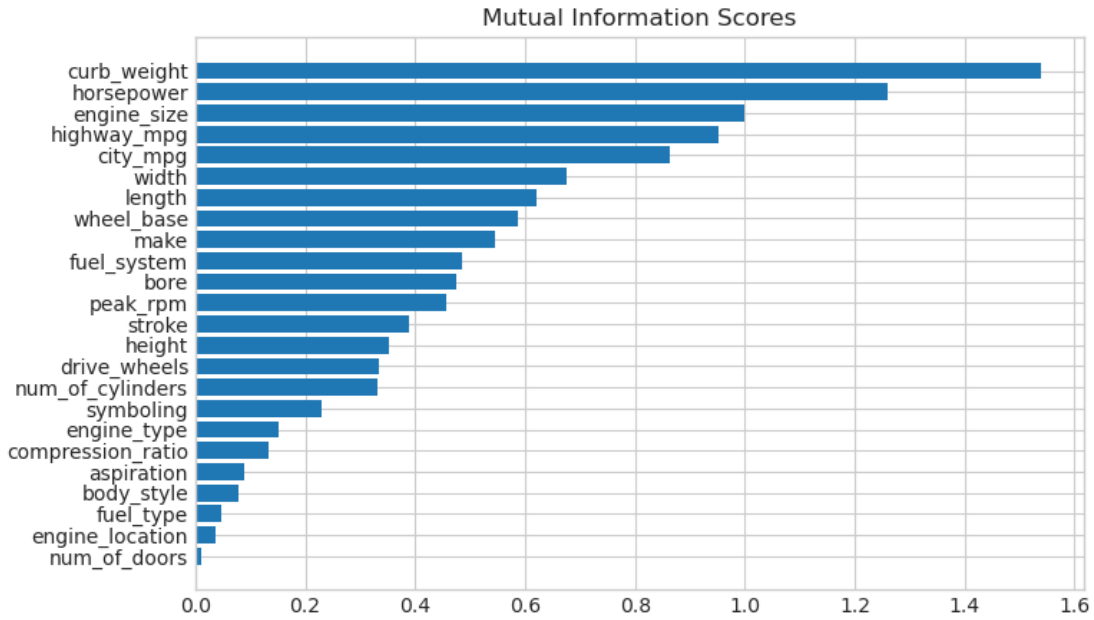
Thông tin tương hỗ (MI)

- Đo lượng thông tin thu được về một biến thông qua việc quan sát biến khác.
- Điểm MI là giá trị không âm
- Nó bằng 0 nếu và chỉ khi hai biến ngẫu nhiên độc lập và giá trị cao hơn có nghĩa là sự phụ thuộc cao hơn (mối quan hệ mạnh)



Điểm MI (Ví dụ)

```python
from sklearn.feature_selection import mutual_info_regression

def make_mi_scores(X, y, discrete_features):
    mi_scores = mutual_info_regression(X, y, discrete_features=discrete_features)
    mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
    mi_scores = mi_scores.sort_values(ascending=False)
    return mi_scores

mi_scores = make_mi_scores(X, y, discrete_features)
mi_scores[::3]  # show a few features with their MI scores
```



https://www.kaggle.com/code/ryanholbrook/mutual-information

Phương pháp Filter: Ưu điểm &; nhược điểm

Lợi thế

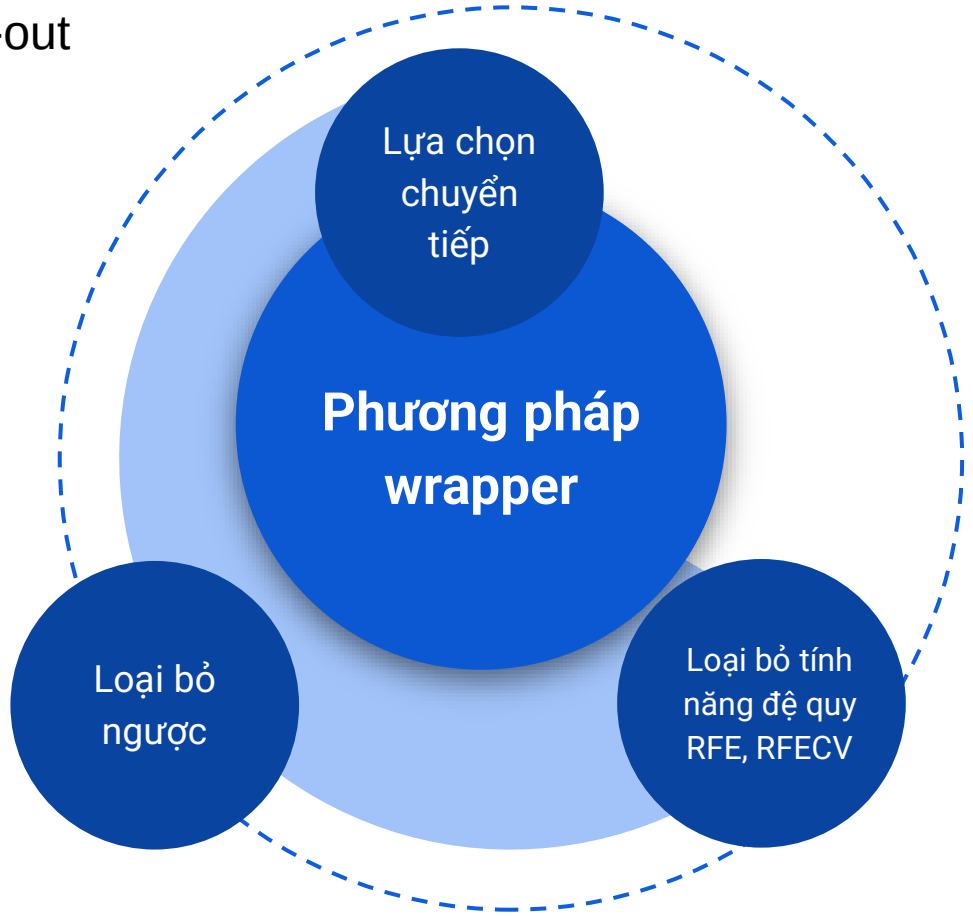
- Hiệu quả tính toán (không yêu cầu mô hình đào tạo)
- Dễ thực hiện và dễ hiểu
- Ít bị overfitting (không liên quan đến mô hình mục tiêu)

Khó khăn

- Thiếu sự tương tác giữa các đặc trưng
- Hạn chế trong việc xác định các tập hợp con tính năng tối ưu
- Nguy cơ đơn giản hóa quá mức

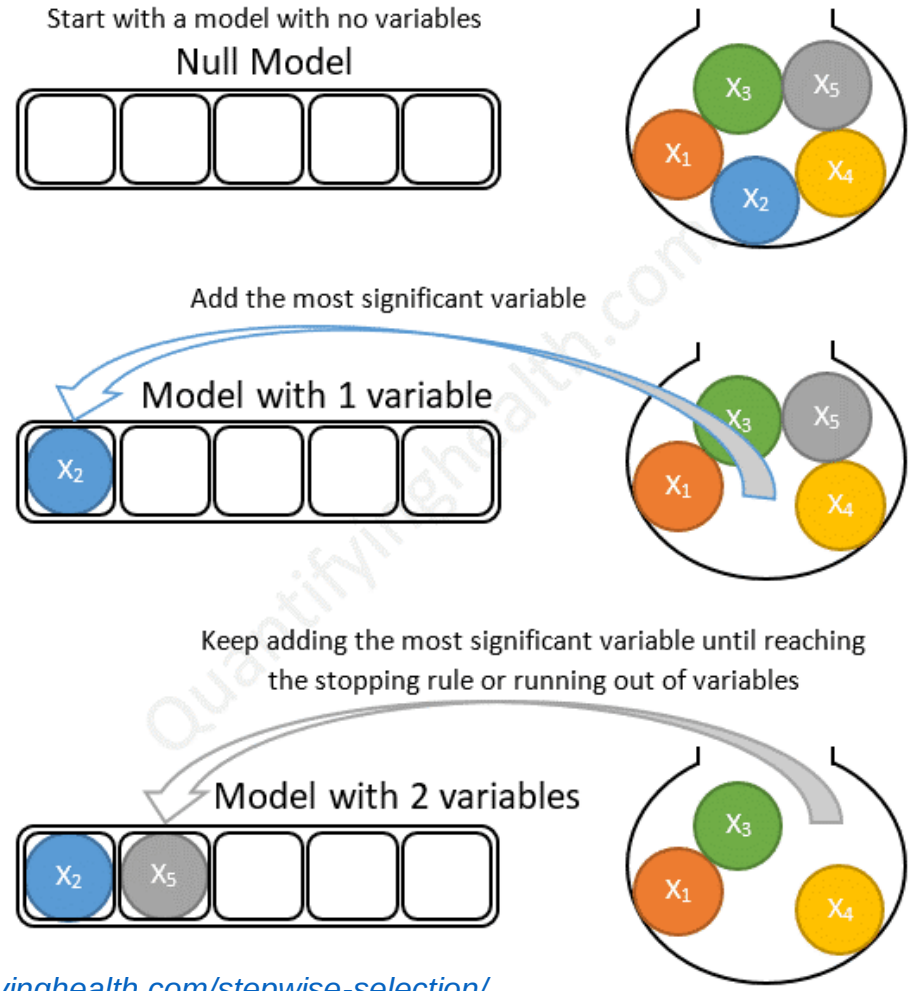
Phương pháp Wrapper

- Các phương pháp wrapper sử dụng mô hình dự đoán để ghi điểm các tập hợp con đặc trưng. Mỗi tập hợp con mới được sử dụng để đào tạo một mô hình, được thử nghiệm trên một tập hợp hold-out



Lựa chọn chuyển tiếp

Forward stepwise selection example with 5 variables:



https://quantifyinghealth.com/stepwise-selection/

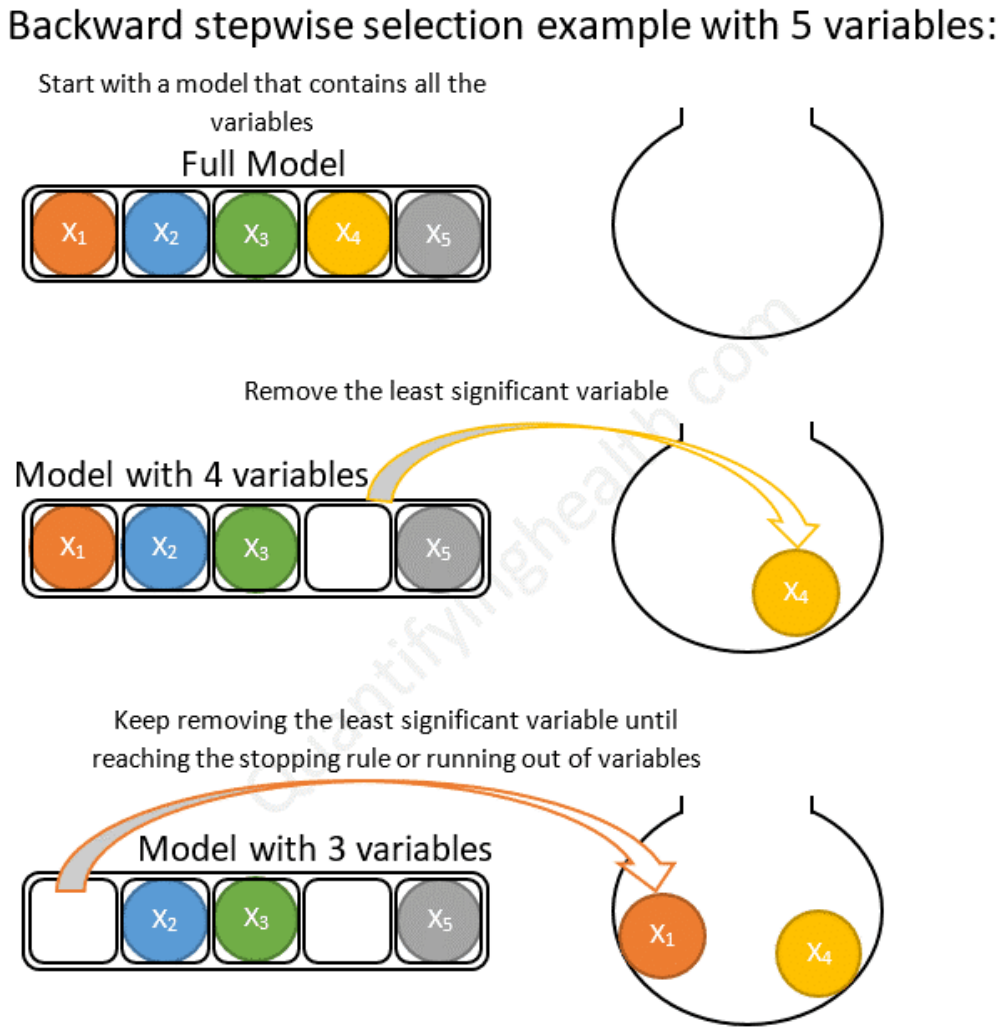
Lựa chọn chuyển tiếp: Bước 1

- Làm thế nào để xác định biến quan trọng nhất để thêm ở mỗi bước?
- Biến quan trọng nhất có thể được chọn để khi được thêm vào mô hình:

Lựa chọn chuyển tiếp: Bước 2

- Quy tắc dừng được thỏa mãn khi tất cả các biến còn lại cần xem xét có giá trị **p** lớn hơn một số ngưỡng được chỉ định, nếu được thêm vào mô hình
- Làm thế nào để xác định ngưỡng?

Loại bỏ ngược



https://quantifyinghealth.com/stepwise-selection/

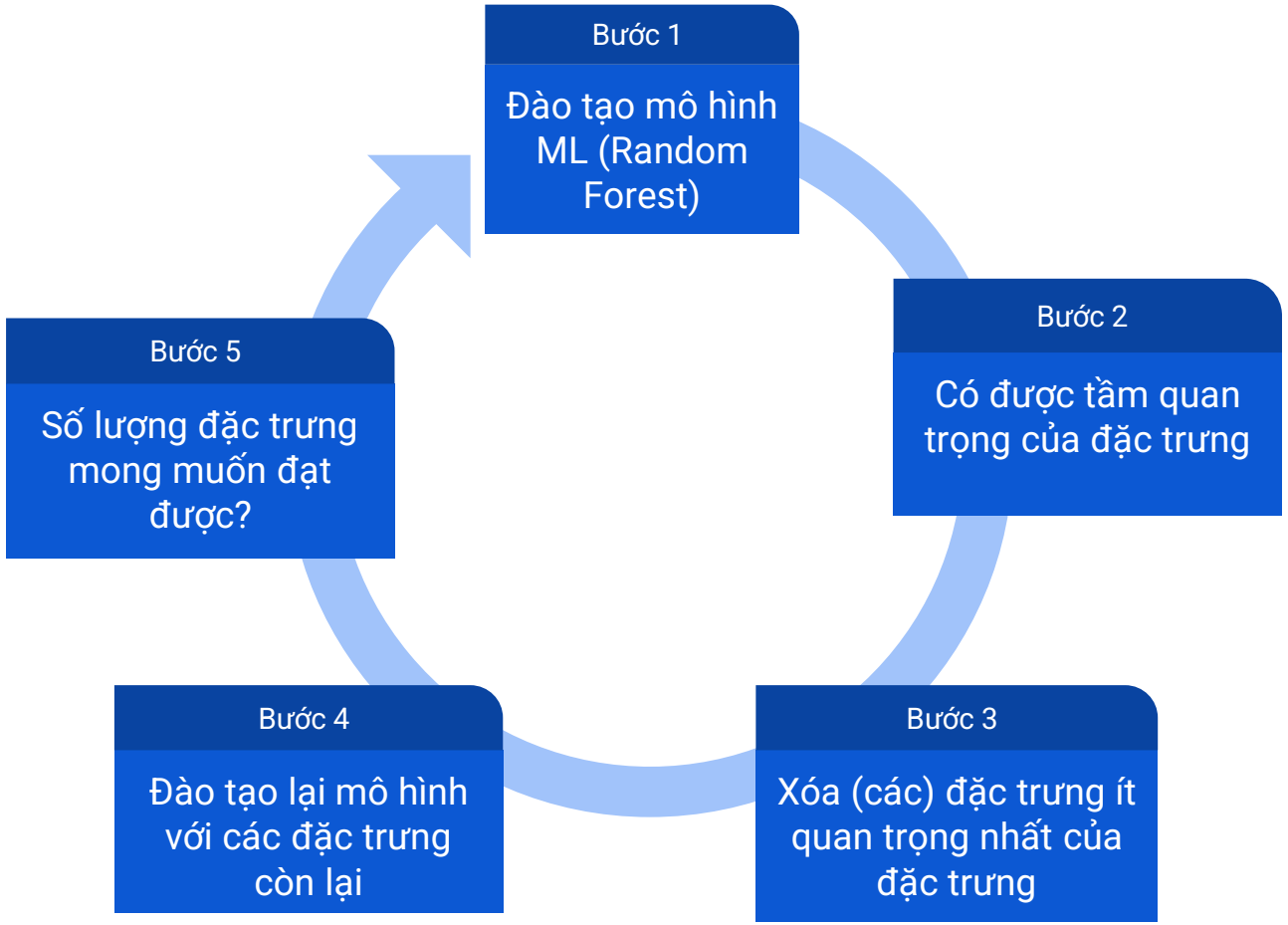
Loại bỏ ngược: Bước 1

- Làm thế nào để xác định biến ít quan trọng nhất để loại bỏ ở mỗi bước?
- Biến ít quan trọng nhất là một biến:

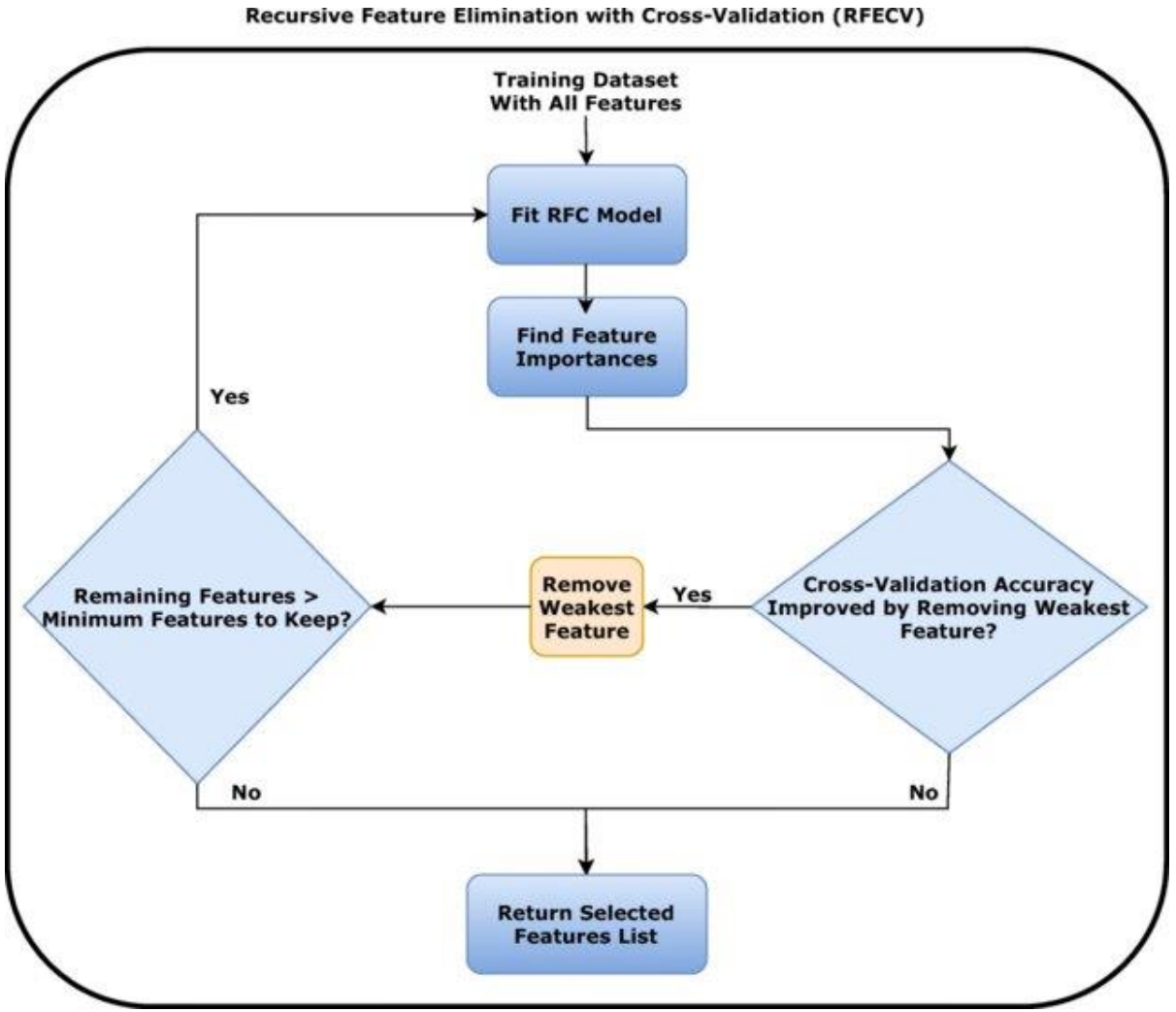
Loại bỏ ngược: Bước 2

- Quy tắc dừng được thỏa mãn khi tất cả các biến còn lại trong mô hình có giá trị **p** nhỏ hơn một số ngưỡng được chỉ định trước

Loại bỏ tính năng đệ quy (RFE)



RFE với Xác thực chéo



Phương pháp Wrapper: Advantages & Disadvantages

Lợi thế

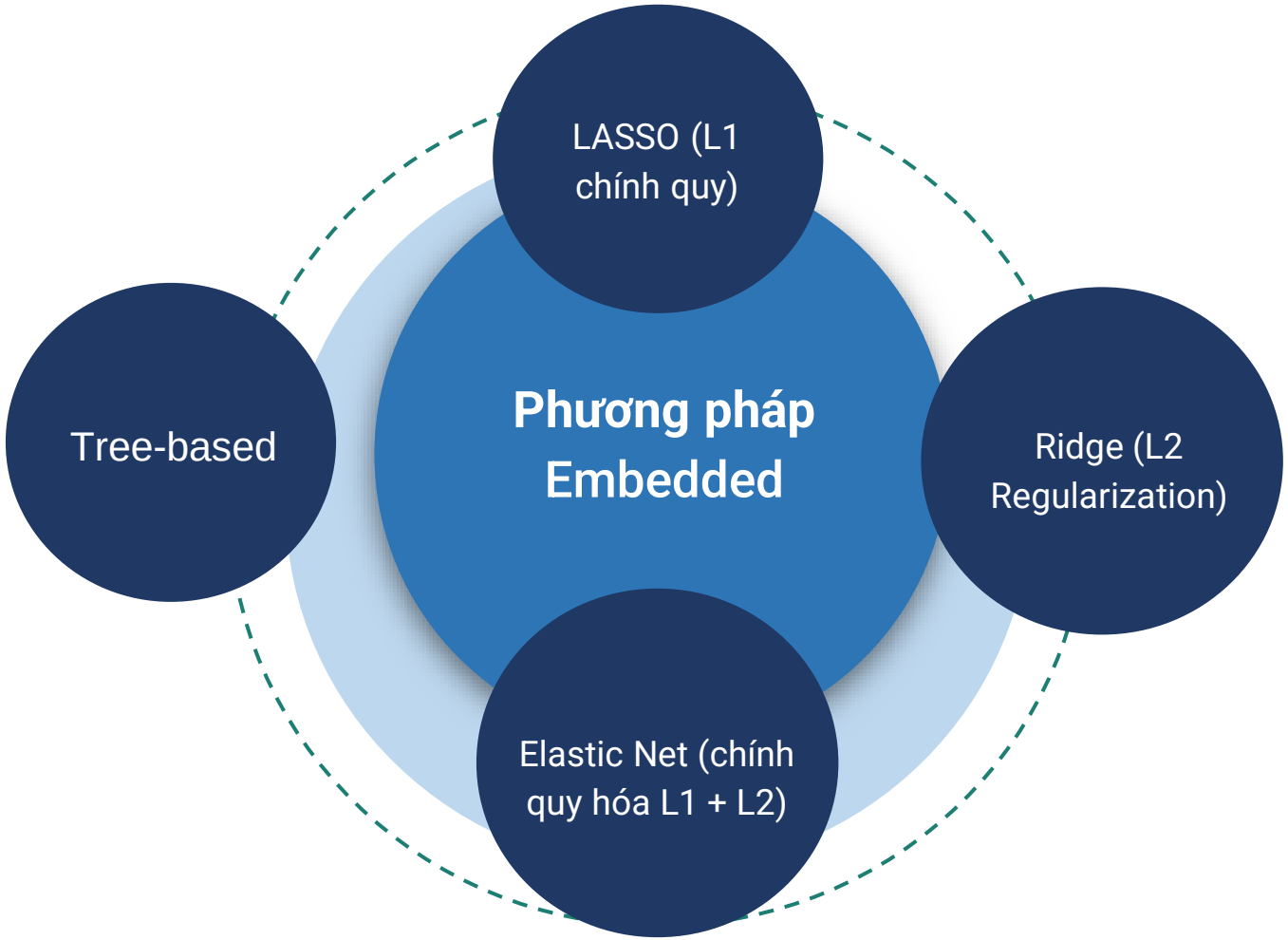
- Nắm bắt tương tác giữa các đặc trưng
- Tập hợp con đặc trưng tối ưu

Khó khăn

- Tính toán tốn kém, đặc biệt là đối với các bộ dữ liệu chiều cao (yêu cầu mô hình đào tạo nhiều lần)
- Dễ bị quá tải
- Phức tạp hơn để thực hiện và hiểu so với các phương pháp Filter.

Phương pháp Embedded

- Nắm bắt tất cả các nhóm kỹ thuật thực hiện lựa chọn đặc trưng như là một phần của quá trình xây dựng mô hình (kết hợp các đặc tính của phương pháp Filter và Wrapper)



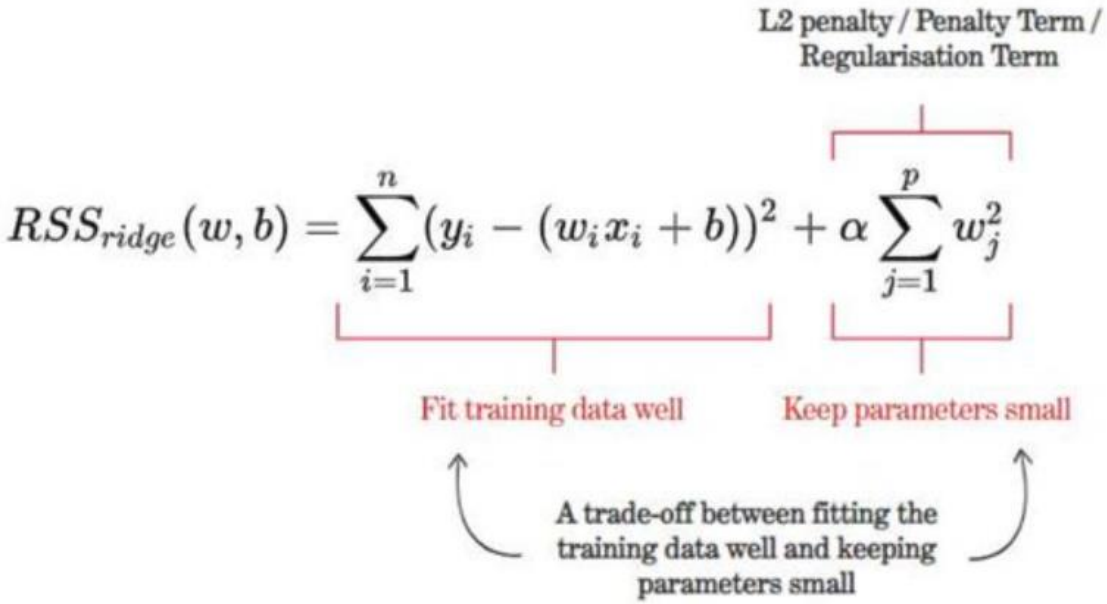
Hồi quy LASSO

- Mô hình tuyến tính sử dụng một hình phạt tương đương với giá trị tuyệt đối của độ lớn của các hệ số
- Một số hệ số có thể trở thành 0 → có thể thực hiện lựa chọn đặc trưng (giảm overfitting)

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Ridge Regression

- Thêm một hình phạt tương đương với bình phương độ lớn của các hệ số
- Không làm giảm (thu nhỏ) hệ số về 0 mà nó đưa các hệ số gần bằng 0 → không có hệ số nào bị loại bỏ
- Không tốt cho việc giảm đặc trưng nhưng sẽ giúp chọn các đặc trưng có liên quan.



Elastic Net

- Kết hợp quy tắc L1 + L2, cân bằng giữa các hình phạt Lasso và Ridge
- Lasso sẽ loại bỏ các đặc trưng và giảm sự phù hợp quá mức trong mô hình tuyến tính. Ridge sẽ làm giảm tác động của các đặc trưng không quan trọng trong việc dự đoán các giá trị mục tiêu.
- Siêu tham số alpha: tối ưu hóa bằng cách sử dụng xác thực chéo

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$$

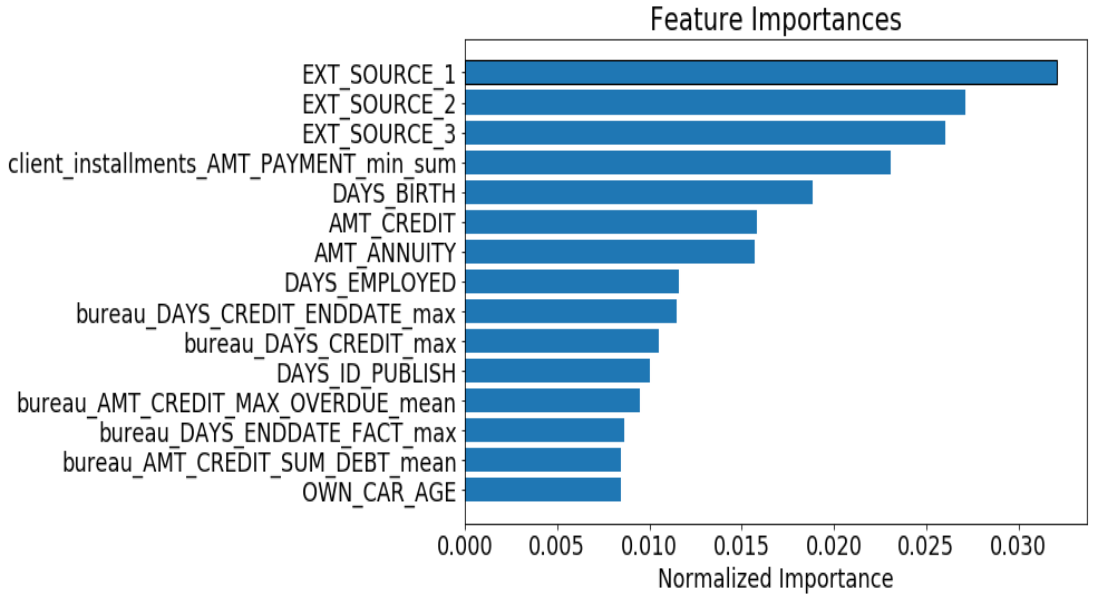
Tree-based

- Đối với mô hình dựa trên cây, các đặc trưng có tầm quan trọng bằng 0 hoàn toàn không được sử dụng để thực hiện bất kỳ phân tách nào
- Chúng ta có thể sử dụng tầm quan trọng của đặc trưng để loại bỏ các đặc trưng mà mô hình không coi là quan trọng.

```
# Find the features with zero importance
zero_features = list(feature_importances[feature_importances['importance'] == 0.0]['feature'])
print('There are %d features with 0.0 importance' % len(zero_features))
feature_importances.tail()
```

```
There are 271 features with 0.0 importance
```

	feature	importance
348	previous_loans_RATE_INTEREST_PRIMARY_sum	0.0
352	client_cash_NAME_CONTRACT_STATUS_XNA_count_nor...	0.0
635	previous_loans_NAME_CASH_LOAN_PURPOSE_Buying a...	0.0
353	previous_loans_PRODUCT_COMBINATION_POS mobile ...	0.0
843	EMERGENCYSTATE_MODE_Yes	0.0



https://www.kaggle.com/code/willkoehrsen/introduction-to-feature-selection

Phương pháp Embedded: Ưu điểm &; nhược điểm

Lợi thế

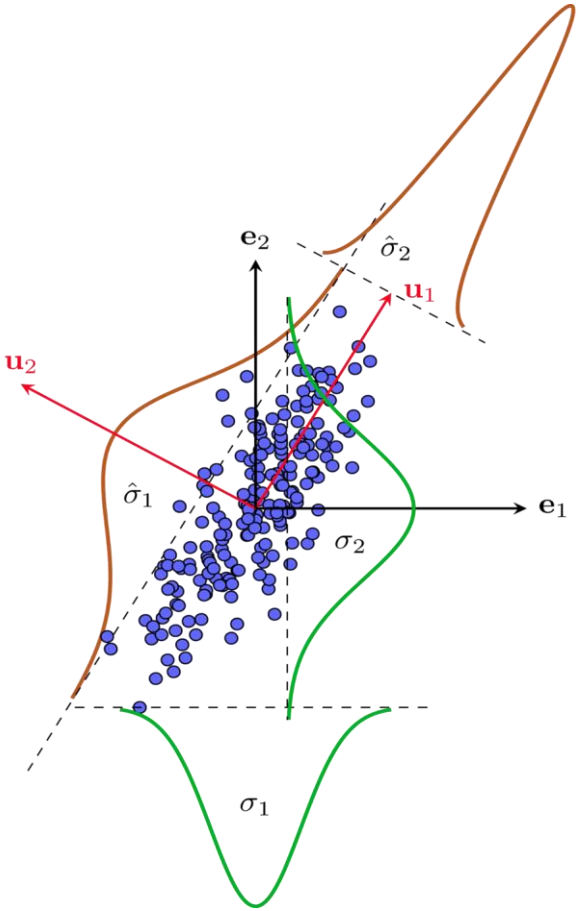
- Hiệu quả: hiệu quả tính toán hơn các phương pháp Wrapper
- Khái quát hóa tốt hơn: tính đến sự tương tác giữa các đặc trưng và tham số mô hình

Khó khăn

- Giới hạn ở một số kiểu máy nhất định có cơ chế lựa chọn đặc trưng tích hợp
- Ít diễn giải hơn so với các phương Filter
- Tiềm năng Overfitting: mô hình phức tạp và tập dữ liệu nhỏ.

Giảm chiều

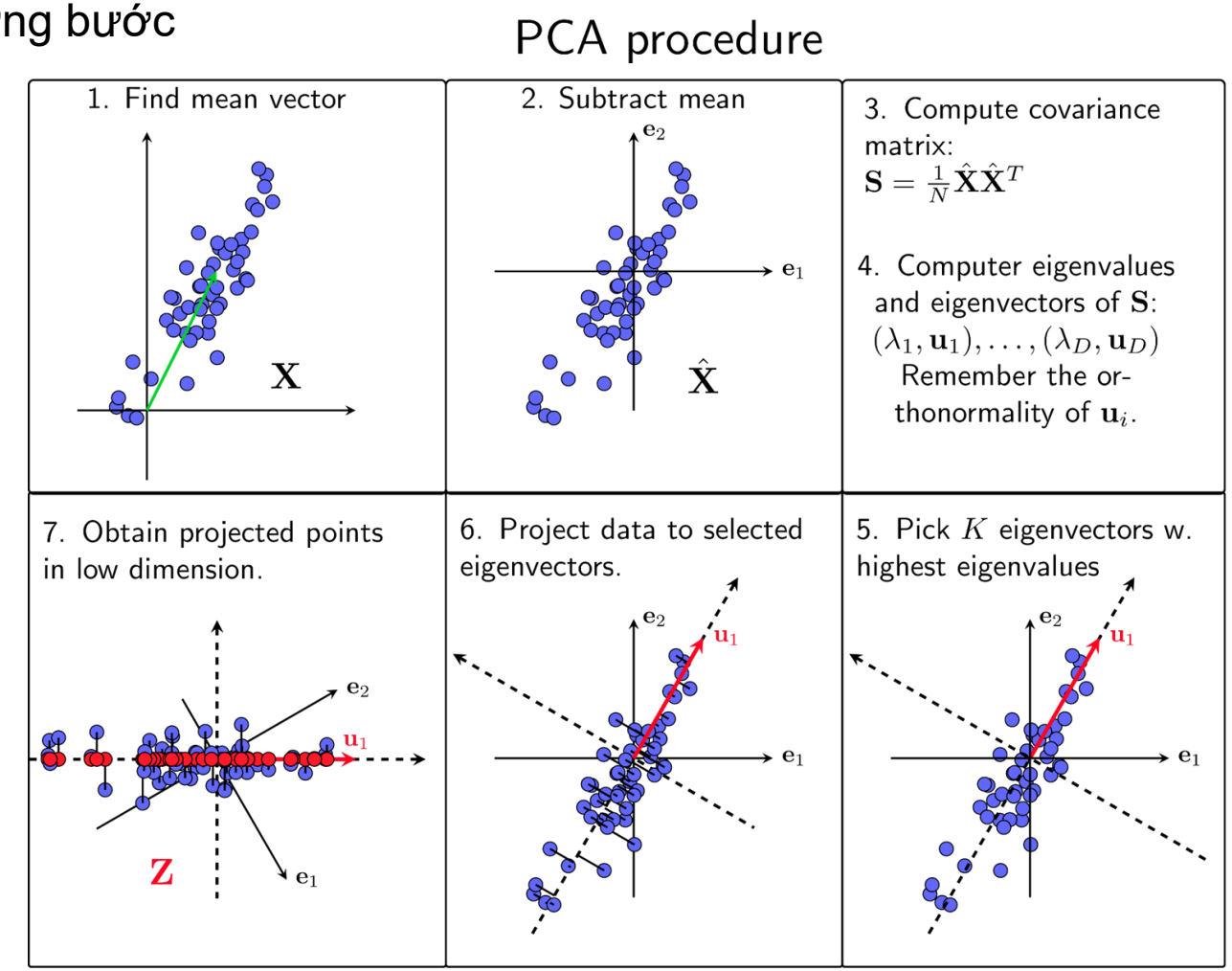
- Xử lý dữ liệu chiều cao bằng cách chuyển đổi các đặc trưng gốc thành không gian chiều thấp hơn
- Cải thiện hiệu suất mô hình bằng cách giảm nhiễu và dự phòng trong dữ liệu



https://machinelearningcoban.com/2017/06/15/pca/

Phân tích thành phần chính - PCA

- PCA từng bước



https://machinelearningcoban.com/2017/06/15/pca/

Giảm chiều: Ưu điểm và nhược điểm

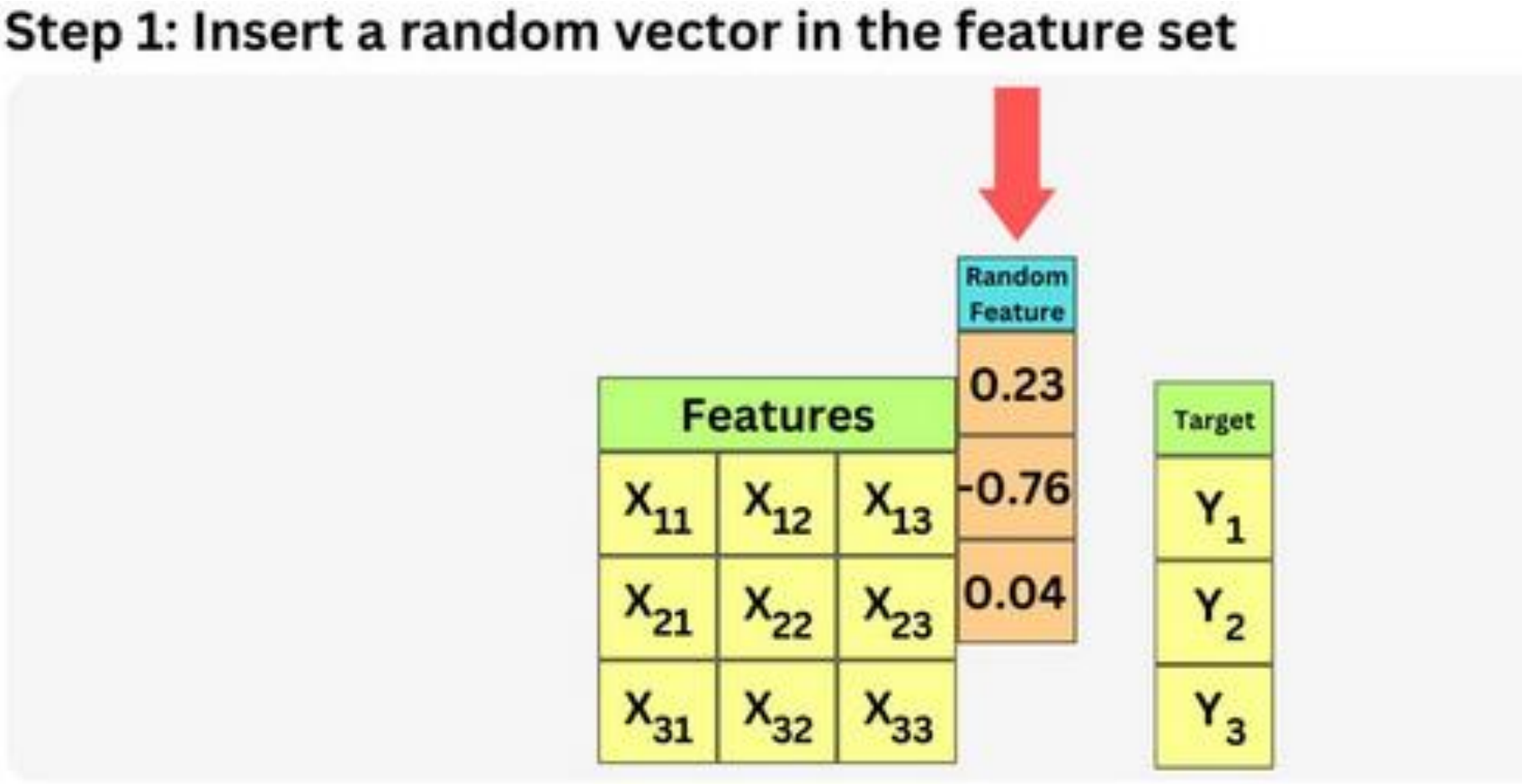
Lợi thế

- Hiệu quả tính toán
- Trực quan hóa: 2D hoặc 3D
- Loại bỏ tiếng ồn: loại bỏ các thành phần phương sai thấp, thường tương ứng với tiếng ồn

Khó khăn

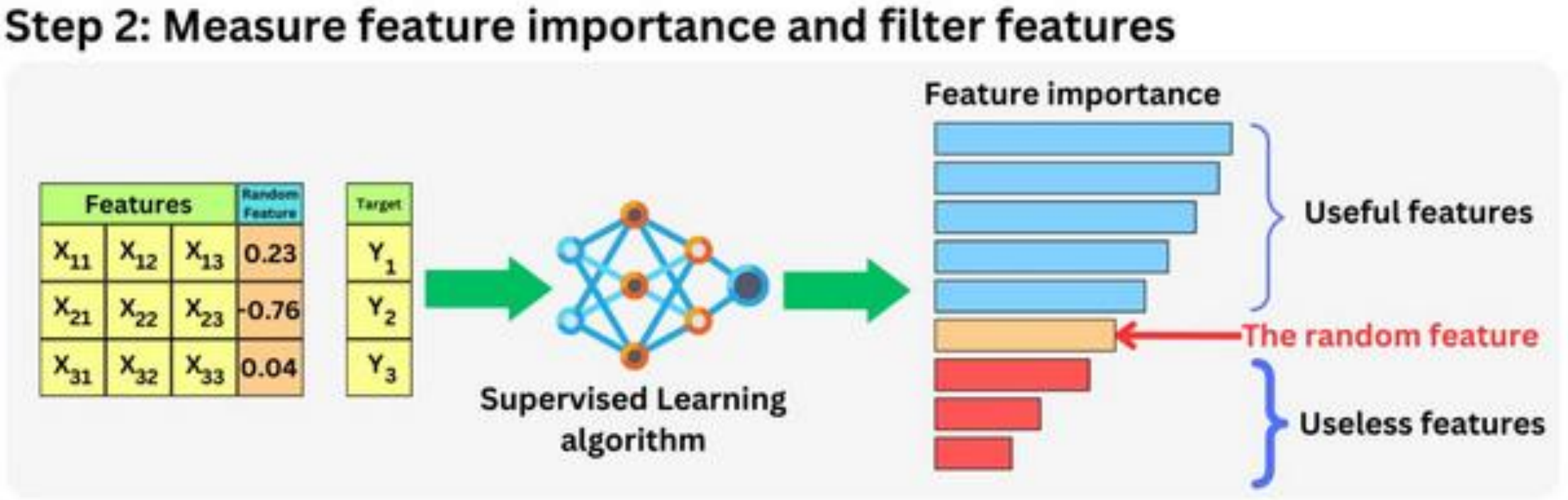
- Khả năng diễn giải: sự kết hợp của các đặc trưng ban đầu và thường khó giải thích
- Có thể không phù hợp với tất cả các loại dữ liệu
- Không có lựa chọn cụ thể: không cung cấp một tập hợp con cụ thể các đặc trưng quan trọng hơn

The Random Bar: Bước 1

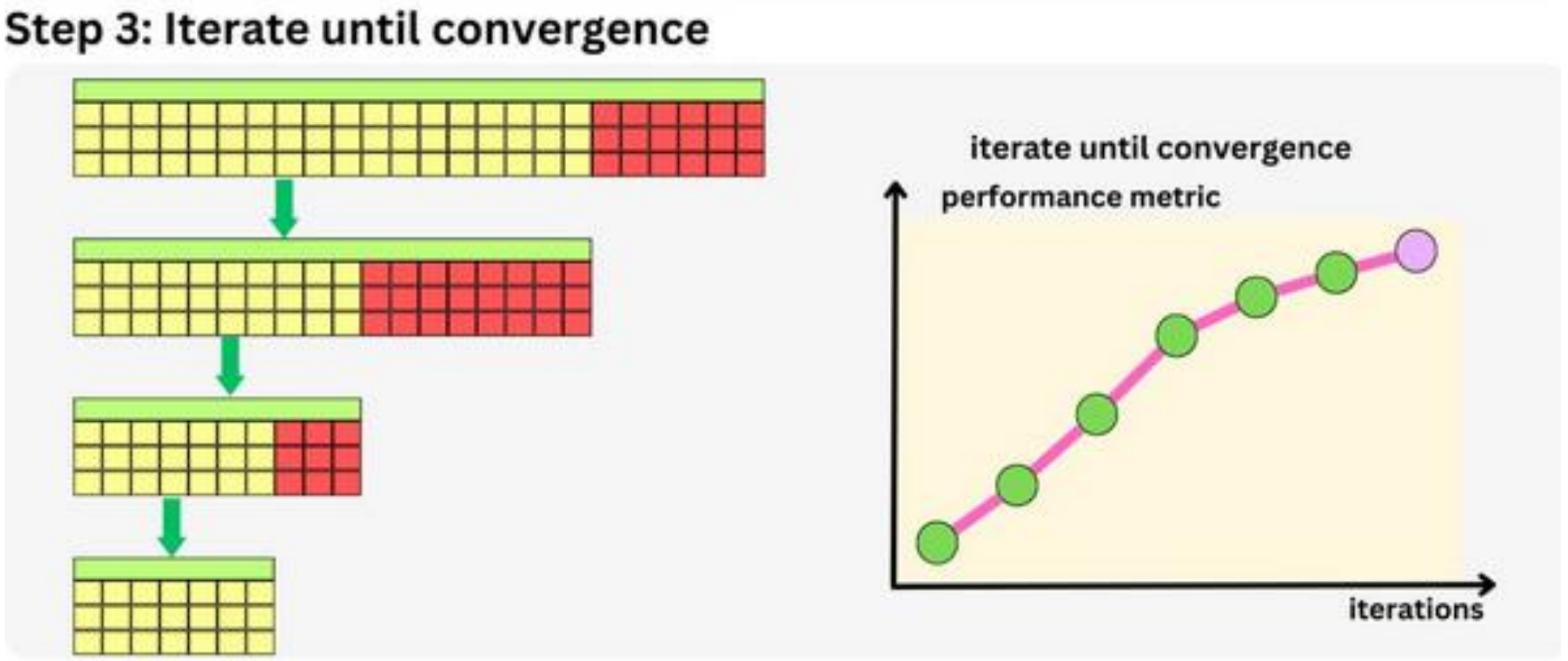


https://www.linkedin.com/feed/update/urn:li:activity:7059555375821307904?utm_source=share&utm_medium=member_desktop

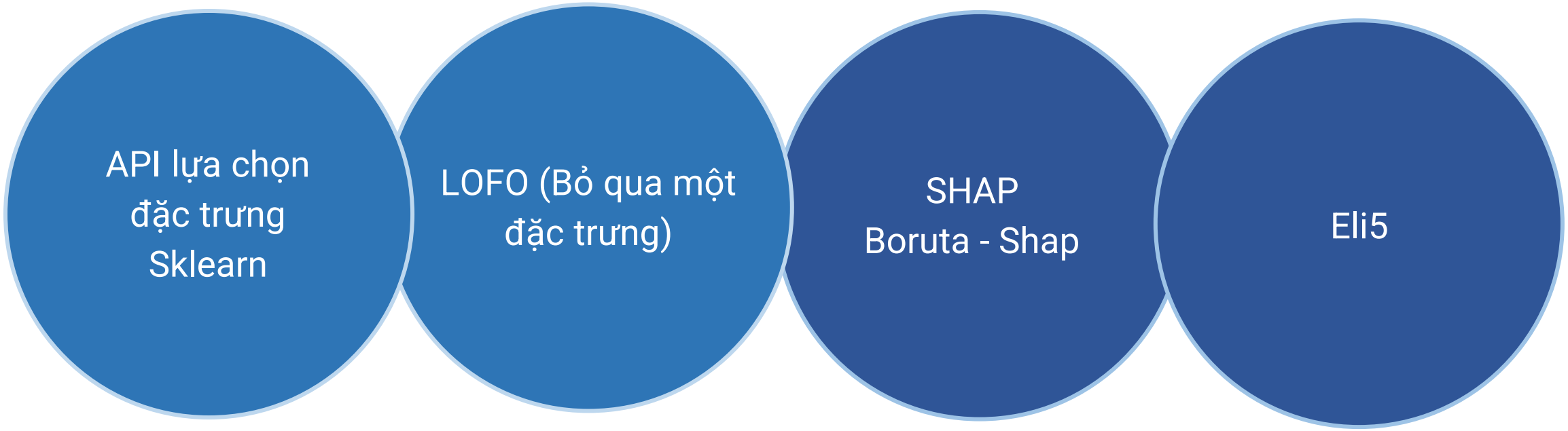
The Random Bar: Bước 2



The Random Bar: Bước 3



Công cụ lựa chọn đặc trưng



Sklearn: Lựa chọn đặc trưng

- API Reference
- Example RFE & RFECV

`sklearn.feature_selection`: Feature Selection

The `sklearn.feature_selection` module implements feature selection algorithms. It currently includes univariate filter selection methods and the recursive feature elimination algorithm.

User guide: See the Feature selection section for further details.

`feature_selection.GenericUnivariateSelect`([...])	Univariate feature selector with configurable strategy.
`feature_selection.SelectPercentile`([...])	Select features according to a percentile of the highest scores.
`feature_selection.SelectKBest`([score_func, k])	Select features according to the k highest scores.
`feature_selection.SelectFpr`([score_func, alpha])	Filter: Select the pvalues below alpha based on a FPR test.
`feature_selection.SelectFdr`([score_func, alpha])	Filter: Select the p-values for an estimated false discovery rate.
`feature_selection.SelectFromModel`(estimator, *)	Meta-transformer for selecting features based on importance weights.
`feature_selection.SelectFwe`([score_func, alpha])	Filter: Select the p-values corresponding to Family-wise error rate.
`feature_selection.SequentialFeatureSelector`(...)	Transformer that performs Sequential Feature Selection.
`feature_selection.RFE`(estimator, *[, ...])	Feature ranking with recursive feature elimination.
`feature_selection.RFECV`(estimator, *[, ...])	Recursive feature elimination with cross-validation to select features.
`feature_selection.VarianceThreshold`([threshold])	Feature selector that removes all low-variance features.

LOFO

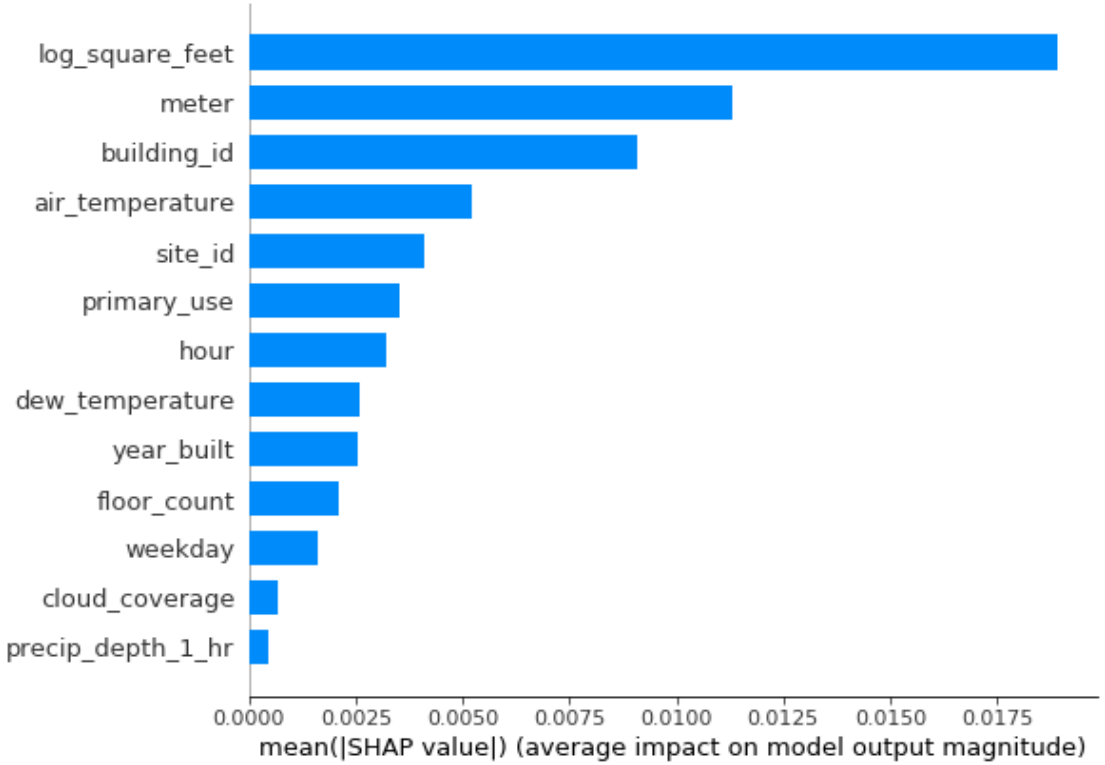
- LOFO trước tiên đánh giá hiệu suất của mô hình với tất cả các đặc trưng đầu vào được bao gồm, sau đó lặp đi lặp lại loại bỏ một đặc trưng tại một thời điểm, đào tạo lại mô hình và đánh giá hiệu suất của nó trên một tập hợp xác thực
- github
- Example notebook

```
 import pandas as pd
from sklearn.model_selection import KFold
from lofo import LOFOImportance, Dataset, plot_importance
%matplotlib inline

# import data
train_df = pd.read_csv("../input/train.csv", dtype=dtypes)

# extract a sample of the data
sample_df = train_df.sample(frac=0.01, random_state=0)
sample_df.sort_values("AvSigVersion", inplace=True) # Sort by time for time split validation

# define the validation scheme
cv = KFold(n_splits=4, shuffle=False, random_state=None) # Don't shuffle to keep the time split spli

# define the binary target and the features
dataset = Dataset(df=sample_df, target="HasDetections", features=[col for col in train_df.columns if

# define the validation scheme and scorer. The default model is LightGBM
lofo_imp = LOFOImportance(dataset, cv=cv, scoring="roc_auc")

# get the mean and standard deviation of the importances in pandas format
importance_df = lofo_imp.get_importance()

# plot the means and standard deviations of the importances
plot_importance(importance_df, figsize=(12, 20))
```

SHAP

- github
- Example notebook & other example

```
shap.summary_plot(shap_values, train[features], plot_type="bar")
```



Boruta - Shap

- Phương pháp lựa chọn đặc trưng bao bọc kết hợp cả thuật toán lựa chọn tính năng Boruta với các giá trị shapley
- github
- Notebook Example

eli5

- Một gói python giúp gỡ lỗi các trình phân loại học máy và giải thích dự đoán của chúng
- github
- docs
- Example Notebook

```
import eli5
from eli5.sklearn import PermutationImportance

perm = PermutationImportance(my_model, random_state=1).fit(val_X, val_y)
eli5.show_weights(perm, feature_names = val_X.columns.tolist())
```

Weight	Feature
0.1750 ± 0.0848	Goal Scored
0.0500 ± 0.0637	Distance Covered (Kms)
0.0437 ± 0.0637	Yellow Card
0.0187 ± 0.0500	Off-Target
0.0187 ± 0.0637	Free Kicks
0.0187 ± 0.0637	Fouls Committed
0.0125 ± 0.0637	Pass Accuracy %
0.0125 ± 0.0306	Blocked
0.0063 ± 0.0612	Saves
0.0063 ± 0.0250	Ball Possession %
0 ± 0.0000	Red
0 ± 0.0000	Yellow & Red
0.0000 ± 0.0559	On-Target
-0.0063 ± 0.0729	Offsides
-0.0063 ± 0.0919	Corners
-0.0063 ± 0.0250	Goals in PSO
-0.0187 ± 0.0306	Attempts
-0.0500 ± 0.0637	Passes

QUIZ & QUESTIONS