ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

MeLIME: Meaningful Local Interpretable Model-agnostic Explanations

[GIÁO VIÊN]

Nguyễn Ngọc Đức

20127038-NGUYỄN PHƯỚC GIA HUY

20127088-NGUYỄN THIỆN HOÀNG TRÍ

MÔN HỌC: NHẬP MÔN HỌC MÁY – 20KHDL2

Thành phố Hồ Chí Minh – 2023

MỤC LỤC

I. TỔNG QUAN

1. Thông tin nhóm

No.	MSSV	Tên	Email
1	20127038	Nguyễn Phước Gia Huy	20127038@student.hcmus.edu.vn
2	20127088	Nguyễn Thiện Hoàng Trí	20127088@student.hcmus.edu.vn

2. Thông tin đồ án

Yêu cầu: Nghiên cứu và trình bày về kỹ thuật hiện đại của mô hình học máy.
Bài nghiên cứu: **MeLIME** - **Me**aningful Local Explanation for Machine Learning Models
Lý do: Nhóm thực hiện đồ án này để nghiên cứu và tìm hiểu thêm về một khía cạnh của học máy, mà ở đây cụ thể là XAI. Đây là một chủ đề nóng và khá mới, đồng thời đây cũng là mục tiêu nhóm hướng tới trong tương lai khi thực hiện khóa luận tốt nghiệp.
Thư mục:

- Report.pdf: file báo cáo và tổng hợp thông tin về kết quả đồ án seminar.
- Source.ipynb: file code (demo) cài đặt ứng dụng LIME và MeLIME.
- Slide.ppt: file để trình bày senimar.

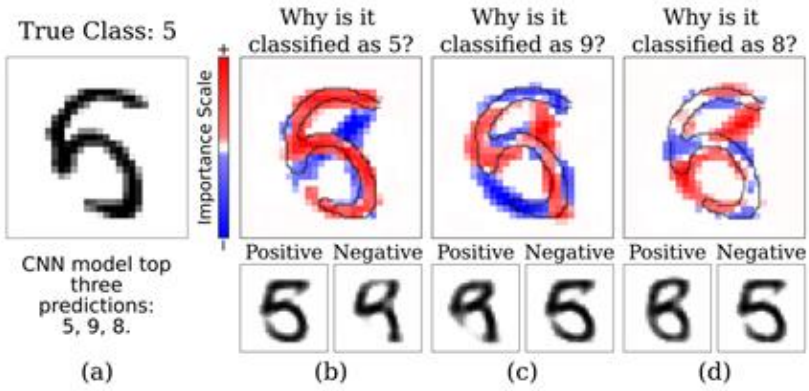
3. Môi trường cài đặt

Ngôn ngữ làm việc: Python.
Môi trường làm việc: Visual studio code, Jupyter Notebook.
Thư viện cài đặt:
+ **Numpy:** Tính toán thông thường.
+ **Matplotlib:** Dùng để trực quan dữ liệu thành biểu đồ.
+ **Scikit-learn:** Để chạy thuật toán Random Forest và thông số đánh giá.
+ **LIME:** Để áp dụng thuật toán LIME giải thích mô hình.
+ **MeLIME:** Để triển khai thuật toán MeLIME giải thích mô hình.

II. GIỚI THIỆU

Có nhiều thuật toán học máy hiện đại tạo ra các mô hình black-box nhằm ngăn cản việc áp dụng chúng trong nhiều lĩnh vực nhạy cảm. Do đó, nhiều phương pháp để giải thích các mô hình học máy đã được đề xuất để giải quyết vấn đề này. Trong bài báo cáo này, nhóm sẽ giới thiệu chiến lược để cải thiện các giải thích cục bộ - phương pháp MeLIME, tạo ra nhiều giải thích có ý nghĩa hơn so với các kỹ thuật khác qua các mô hình học máy khác nhau, hoạt động trên các loại dữ liệu khác nhau. MeLIME tổng quát hóa phương pháp LIME, cho phép lấy mẫu bị nhiễu linh hoạt hơn và sử dụng các mô hình có thể giải thích cục bộ khác nhau.

Ngoài ra, tác giả của phương pháp cũng giới thiệu các sửa đổi cho các thuật toán huấn luyện tiêu chuẩn của các mô hình có thể giải thích cục bộ (local interpretable models) khuyến khích các mô hình giải thích mạnh mẽ (robust explanations) hơn, ngoài ra cho phép tạo ra các ví dụ phản chứng. Để chứng minh sức mạnh của phương pháp được đề xuất, chúng tôi tiến hành các thí nghiệm trên dữ liệu bảng, hình ảnh và văn bản; tất cả đều cho thấy sự cải thiện về giải thích. Cụ thể, MeLIME tạo ra các giải thích có ý nghĩa hơn trên tập dữ liệu MNIST so với các phương pháp như GuidedBackprop, SmoothGrad và Layer-wise Relevance Propagation.



Mô tả các giải thích cục bộ MeLIME được tạo ra cho một mô hình CNN được huấn luyện trên tập dữ liệu MNIST. Hình ảnh hiển thị một hình ảnh số năm và các giải thích được tạo ra với các ví dụ phản chứng cho ba dự đoán hàng đầu: (a) số năm; (b) số sáu; và (c) số tám.

Do MeLIME được phát triển và được thực hiện dựa trên famework LIME, nên trước tiên chúng ta sẽ tìm hiểu thêm về sơ lược về LIME

III. LIME

1. Khái niệm

LIME - Local Interpretable Model-agnostic Explanations là một phương pháp model-agnostic (không phụ thuộc mô hình) để tạo ra các giải thích cục bộ cho các mô hình học máy. Nó giới thiệu một framework chung để tạo ra giải thích cục bộ cho bất kỳ mô hình máy học nào. LIME hoạt động như sau. Cho một mô hình học máy f , thì một giải thích cục bộ sẽ được xây dựng cho mẫu x^* được sử dụng một mô hình có khả năng diễn giải g, với $g \in G$ mà G là một tập các mô hình có khả năng diễn giải. Mô hình cục bộ g được tìm ra bằng cách tối tiểu hóa

$$\xi(x^*) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_{x^*}) + \Omega(g) \quad (1)$$

Mà $L(f, g, \pi_{x^*})$ là một độ đo lường cho việc sai lệch g trong việc sấp sỉ hàm f trên π_{x^*} (một độ đo tính cục bộ xung quanh x^*) và $\Omega(g)$ là một độ đo tính phức tap của mô hình cục bộ g .

Trong thực tế, cài đặt nguyên bản của LIME là sử dụng làm một hàm mất mát bình phương được trọng số hóa bởi π_{x^*}, mà π_{x^*} là một Gaussian kernel mà tổng được thực hiện trên các mẫu được vẽ đều gôm đều chọn một cách thống nhất trong không gian đặc trưng. Hơn nữa, nó sử dụng G như một lớp các mô hình tuyến tính và Ω được chọn sao cho tối đa tham số K được chọn trong mô hình tuyến tính. Sự diễn giải bởi LIME không cần phải được thực hiện trên cùng không gian được sử dụng để xây dựng g ; một không gian đặc trưng có khả năng diễn giải hơn sẽ được sử dụng.

Các biến thể của LIME đã được đề xuất trong tài liệu, trong đó số đó thì K-LIME, LIME-SUP và NormLIME. K-LIME sử dụng các kỹ thuật phân cụm (K-means) để phân chia một tập dữ liệu thành K cụm; mỗi phân vùng dữ liệu được sử dụng để huấn luyện một mô hình tuyến tính cục bộ. Giá trị K được điều chỉnh để tối đa hóa R^2 cho tất cả các mô hình cục bộ. LIME-SUP sử dụng một chiến lược tương tự; tuy nhiên, nó cài đặt một cây phân vùng giám sát để có thể cải thiện các giải thích được tạo ra. NormLIME tổng hợp và chuẩn hóa nhiều giải thích cục bộ để tạo ra một giải thích toàn cục cho lớp cụ thể.

2. Quy trình

1) <u>Xác định điểm cần giải thích</u>
   Chuyển x → x’

$$x \in \mathbb{R}^d \underset{x = h_x(x')}{\overset{x' = h_x^{-1}(x)}{\rightleftarrows}} x' \in \{0,1\}^{d'}$$

Original representation the instance being explained

Human interpretable representation of this instance

Với x là mẫu dữ liệu ban đầu.
Với x' là vector - đại diện có thể giải thích (có thể hiểu bởi con người) của x.
(Chú thích với dữ liệu dạng bảng thì x = x')

2) Tạo các biến lân cận của mẫu đang xét
Với x^*là mẫu đang xét – cần tạo LIME sử dụng các hàm phân phối $\mathbb{G}(x^*, r)$ để tạo ra các biến lân cận xung quanh x^*, sử dụng mô hình black box $f(x)$, để dựa đoán (phân lớp) cho các điểm dữ liệu đó. Ví dụ sử dụng phân phối đều $x^{\sim} \sim$ **Uniform(*I*),** với $I = \{\{x \in x_1, \dots, x_n\}: d(x^*, x) \leq r\}$

Dựa trên khoảng cách của từng điểm dữ liệu trên đến với x^*, tính trọng số cho mỗi điểm lân cận, π_{x^*} như là độ đo proximity. Ví dụ:

$$\pi_x(z) = exp(-D(x,z)^2/\sigma^2)$$

$\pi_x(z)$: Trọng số của z
z: Điểm mô phỏng lân cận đang xét
$D(x,z)$:Khoảng cách giữa x và z
σ^2: Phương sai

3) Khớp dữ liệu

$$\xi(x^*) = \arg\min_{g \in G} \mathcal{L}(f, g, \pi_{x^*}) + \Omega(g)$$

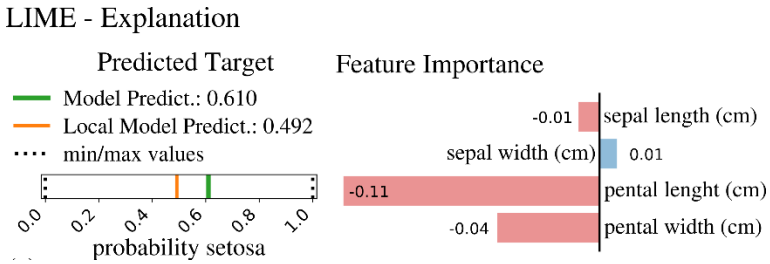
Sử dụng mô hình g để khớp dữ liệu dựa trên các điểm lân cận và trọng số.
Với $g \in G$, với Glà tập hợp (class) các mô hình khả năng diễn giải, ví dụ như tập các mô hình hồi quy tuyến tính, tập hợp các mô hình decision tree,...
Công thức hàm mất mát (locally weighted square loss) :

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in \mathcal{Z}} \pi_x(z) \left(f(z) - g(z')\right)^2$$

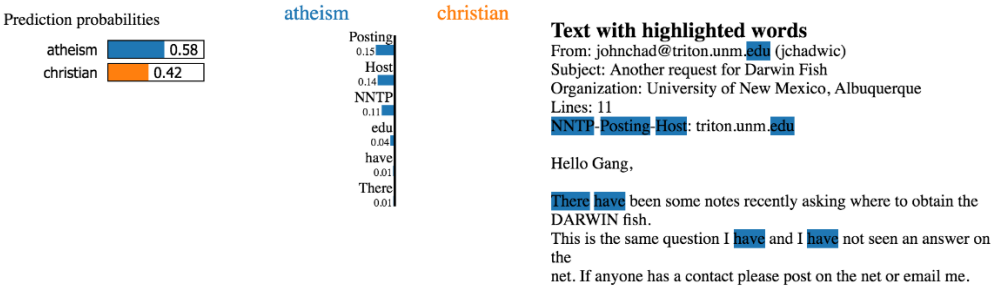
Và độ phức tạp $\Omega(g)$ có thể tính bằng số lượng thuộc tính khác 0 đối với mô hình tuyến tính và độ sâu đối với mô hình decision tree.
Tính tổng độ mất mát của từng mô hình $L(f, g, \pi_{x^*})$ với độ phức tạp $\Omega(g)$, chọn mô hình với tổng trên là nhỏ nhất.

4) Tạo ra các giải thích.
Sử dụng trọng số (linear) của x^* để tạo ra các lời giải thích cho chính nó, ví dụ:

- Đối với dữ liệu dạng bảng: sử dụng trọng số để thể hiện mức độ quan trọng của mỗi thuộc tính (thuộc tính với trọng số quá nhỏ có thể bỏ qua) đối với các loại phân lớp.
  Ví dụ:



- Đối với dữ liệu dạng ngôn ngữ: sử dụng mức độ quan trọng của các từ (trọng số) đối với các loại phân lớp.
  Ví dụ:



- Đối với dữ liệu dạng ảnh: hiển thị hoặc tô màu pixel quan trọng (có trọng số cao) đối với kết quả phân lớp .
  Ví dụ:

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

IV. MeLIME

1. Khái niệm

MeLIME – Meaningful Local Interpretable Model-agnostic Explanations là phương pháp được phát triển dựa trên dựa trên phương pháp LIME (bản mở rộng của LIME).

Tiền đề phát triển dựa trên khuyết điểm của LIME:

- Làm thế nào để tạo ra chính xác các điểm mẫu trên khu vực lân cận của một trường hợp?
- Làm thế nào để kiểm soát sự cân bằng giữa độ chính xác của mô hình diễn giải và khả năng diễn giải của nó?
- Làm thế nào để chọn số lượng điểm mẫu được tạo ra xung quanh trường hợp cần được giải thích?

Hướng giải quyết:

- Một cơ chế để tạo dữ liệu xung quanh khu vực lân cận có ý nghĩa của x^*, ngược lại với LIME, nó sẽ xem xét cân nhắc phân bố của dữ liệu (Mục Interpretable Local.
- Models).
- Một mô hình có tính diễn giải được sử dụng để tạo ra giải thích cục bộ (Mục Interpretable Local Models).
- một chiến lược để xác định số lượng mẫu cục bộ cần thiết để có được một giải thích cục bộ mạnh mẽ (Mục Robustness of Explanations).

Thuật toán: Hướng tiếp cận để giải thích dự đoán cho $x^* \in X$ tạo ra từ mô hình học máy hộp đen f đã được khớp với tập dữ liệu $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ như sau. Đầu tiên, chúng ta sẽ tạo ra các mẫu dự trên vùng lân cận của x^*với hàm tạo mẫu $\mathbb{G}(x^*, r)$ nó mã hóa những thứ là khu vực có nghĩa trong X . Hàm tạo mẫu này có thể được chọn dựa vào trường hợp; trong mục Robustness of Explanations, chúng tôi đưa ra vài ví dụ phù hợp dữ liệu dạng bảng, hình ảnh và dạng văn bản. Để $(x_1, f(x_1)), \ldots, (x_b, f(x_b))$ là các dữ liệu được tạo ra cùng với các dự đoán bởi mô hình học máy hộp đen. Sau đó chúng ta sẽ chuyển đổi các đặc trưng $x \in X$ thành không gian mới $X', T : X \rightarrow X'$, mà trong đó con người sẽ dễ hiểu các giải thích hơn. Người dùng nên xác định phép biến đổi T , nó sẽ phụ thuộc vào tính chất của giải thích mong muốn và miền kiến thức cụ thể của nhiệm vụ. Ví dụ, trong khi classifier (hàm phân lớp) dữ liệu văn bản có thể được huấn luyện bằng cách sử dụng word embeddings (nhúng từ), thường thì sẽ dễ dàng hơn để giải thích nó bằng không gian đặc trưng bag-of-words (túi từ). Khi người dùng muốn sử dụng các đặc trưng ban đầu, T sẽ được thiết lập như một phép biến đổi identity. Hãy để $(x'_1, f(x_1)), \ldots, (x'_b, f(x_b))$ là tập dữ liệu kết quả. Cuối cùng, chúng ta khớp một mô hình dự đoán có thể giải thích, g , cho tập này. Trong quá trình huấn luyện, vì $\mathbb{G}(x^*, r)$ mã hóa một khu vực ý nghĩa, các mẫu được tạo ra một cách cục bộ cũng có thể được sử dụng như các ví dụ nghịch đảo để bổ sung cho giải thích được tạo ra. MeLIME thu thập năm mẫu thuận lợi và không thuận lợi nhất theo dự đoán của hộp đen. Thủ tục này được lặp lại cho

đến khi các tiêu chí hội tụ cho giải thích thu được đạt thõa mãn (Mục Robustness of Explanations). Thuật toán 1 tóm tắt MeLIME. Bản triển khai đầy đủ của MeLIME có sẵn trên https://github.com/tiagobotari/melime.

Để rõ hơn ta sẽ xem xét 3 thành phần chính của MeLIME.

2. Bộ tạo mẫu lân cận (Generating samples on a meaningful neighborhood).

Để tạo ra một giải thích cục bộ có ý nghĩa, việc lấy mẫu dữ liệu xung quanh x^* là rất quan trọng. Cụ thể hơn, dữ liệu cục bộ (được sử dụng để khớp với mô hình có thể giải thích) cần được lấy mẫu từ một hàm ước lượng giống với phân phối tạo ra dữ liệu huấn luyện ban đầu. Nếu điều này không đúng, mô hình có thể được huấn luyện trên các khu vực của không gian đặc trưng không được sử dụng để huấn luyện mô hình hộp đen, có thể dẫn đến giải thích không chính xác.

Algorithm 1: Strategy to Generate Meaningful Explanation - MeLIME

Input : Black-box ML Model, f; instance x^* to be explained; transformation $T : \mathcal{X} \longrightarrow \mathcal{X}'$ that maps the original feature space to the space that will be used to produce the local model g; generator of samples on a neighborhood of size r around x^*, $\mathbb{G}(x^*; r)$; batch size b; convergence parameters ϵ_c and σ

Output: Explanation about why $f(x^*)$ is the prediction for x^*

$\mathcal{D} \leftarrow \emptyset$;

repeat

  for $i = 1, \ldots, b$ **do**

    $x_i \leftarrow \mathbb{G}(x^*; r)$;

    $x'_i \leftarrow T(x_i)$;

    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(x'_i, f(x_i))\}$;

  train g using $\mathcal{D}$;

  compute ϵ and δ, the g training error, and the converge criteria is defined in section .

until $\epsilon > \epsilon_c$ *and* $\delta > \sigma$;

Get explanations α from g (Section);

return α ;

Hãy cho r là tham số điều khiển kích thước vùng lân cận. Chúng tôi ký hiệu $\mathbb{G}(x^*; r)$ là hàm sinh mẫu ở vùng lân cận kích thước r xung quanh x^*. Chúng tôi nghiên cứu bốn hàm tạo mẫu khác nhau, $\mathbb{G}$:

1) **KDEGen**: Bộ ước lượng mật độ nhân (KDE – kernel density estimator) với kernel Gaussian được khớp bằng cách chọn một độ rộng hợp lý h của kernel. Sau đó, tập con I của các điểm trong tập huấn luyện gốc trong bán kính r được xác định từ x^*, tức là $I = \{\{x \in x_1, \ldots, x_n\}: d(x^*, x) \leq r\}$ trong đó d là hàm khoảng cách. Các mẫu cục bộ được tạo ra bằng cách lặp đi lặp lại việc vẽ $x^{\sim} \sim$ **Uniform(**I **)** để liên tục lấy mẫu một điểm mới $x_{new} \sim K(\cdot, ^{\sim} x)$. Với kernel Gaussian, điều này giống như việc lấy mẫu tiêu chuẩn từ KDE Gaussian, ngoại trừ việc chúng ta chỉ lấy mẫu từ các mẫu huấn luyện trong bán kính r . Xem thuật toán 2 để biết chi tiết.

Algorithm 2: KDEGen

Input : Instance x^*; neighborhood size r around x^*, smoothing kernel $K(x, x^*)$, samples $D_x = \{x_1, \ldots, x_n\}$
Output: new sample point x_{new}
Let $\mathcal{I} = \{x \in D : d(x^*, x) \leq r\}$
Sample $\tilde{x} \sim \text{Uniform}(\mathcal{I})$;
Sample $x_{\texttt{new}} \sim K(\cdot, \tilde{x})$;
return x_{new} ;

2) **KDEPCAGen**: KDEGen có thể không hiệu quả nếu số đặc trưng là lớn. Do đó, KDEPCAGen trước tiên thực hiện Phân tích thành phần chính (PCA - Principal Component Analysis) để ánh xạ các mẫu vào không gian chiều thấp hơn bằng cách áp dụng một phép biến đổi $W : X \to \mathbb{R}^m$, trong đó m là một số nhỏ được chọn bởi người dùng hoặc bằng cách đặt một giá trị tối thiểu cho tỷ lệ phương sai giải thích tích lũy. Sau khi dữ liệu được chiếu sang $\mathbb{R}^m$, KDEGen được áp dụng trên không gian này. Sau khi tạo ra một mẫu mới $z \in \mathbb{R}^m$, nó được ánh xạ trở lại X bằng cách sử dụng phép biến đổi nghịch đảo xấp xỉ W^{-1} (trong đó các thành phần có giá trị riêng nhỏ được đặt bằng không).

3) **VAEGen**: Đầu tiên, một *Variational Auto Encoder* (VAE - Mã hóa tự động biến phân) được huấn luyện trên tập dữ liệu D . Đặt $q_\phi(z|x)$ là bộ mã hóa (encoder) và $q_\theta(x|z)$ là bộ giải mã (decoder) của VAE, trong đó z là các biến ẩn. Cho một mẫu x^*, và khu vực lân cận với kích thước r , quá trình tạo ra mẫu mới được thực hiện bằng cách: (i) mã hóa x^* thông qua $z^* \sim q_\phi(z|x^*)$, (ii) rút $\epsilon \sim Uniform(-r, r)^m$ với m là kích thước của z , (iii) giải mã $x_{new} \sim q_\theta(x|z^* + \epsilon)$. Xem thuật toán 3 để biết chi tiết quá trình này.

Algorithm 3: VAEGen

Input : Instance x^*; Encoder: $q_\phi(z|x)$; Decoder: $p_\theta(x|z)$; neighborhood size r around the representation of x^* in the latent variable, z^*.
Output: new sample point x_{new}
$z^* \sim q_\phi(z|x^*)$;
$\epsilon \sim \texttt{Uniform}([-r, r])^m$;
$z \leftarrow z^* + \epsilon$;
$x_{\text{new}} \sim p_\theta(x|z)$;
return x_{new} ;

4) **Word2VecGen**: Xem dữ liệu văn bản như một tập các token $\mathbb{T}$. Đầu tiên, chúng ta huấn luyện nhúng word2vec cho $\mathbb{T}$ bằng cách sử dụng tập văn bản huấn luyện. Cho $\psi(t) \in \mathbb{R}^m$ là biểu diễn word2vec của token $t \in \mathbb{T}$. Để lấy một mẫu mới xung quanh x^*, đầu tiên chúng ta trích xuất các token của nó. Cho $t(x^*)$ là vectơ chứa các token đó. Sau đó, chúng ta chọn một phần tử trong $t(x^*)$ ngẫu nhiên, gọi là t_i . Cuối cùng, chúng ta thay thế t_i trong $t(x^*)$ bằng một token láng giềng được chọn ngẫu nhiên từ tập $\{t \in \mathbb{T}: d(\psi(t), \psi(t_i)) \leq r\}$. Xem thuật toán 4 để biết chi tiết.

Algorithm 4: Word2VecGen

Input : Set of tokens $\mathcal{S}^*$ of sentence x^*; a token corpus $s \in \mathcal{S}$; neighborhood size r;
$Encoder : \mathcal{S} \rightarrow Z$

Output: new sample point x_{new}

$s_j \sim Uniform(\mathcal{S}^*)$;

$z_j^* \leftarrow Encoder(s_j^*)$;

$z_i \leftarrow Encoder(s_i), \quad \forall s_i \in \mathcal{S}$;

$\mathcal{I} \leftarrow \{s_i : d(z_i, z_j^*) \leq r\}$;

$x_k \sim Uniform(\mathcal{I})$;

Generate x_{new} replacing the token s_j by s_k in S^*;

return x_{new} ;

3. Mô hình cục bộ (Interpretable Local Models).

1. **Local Linear Model**: Sử dụng một mô hình tuyến tính, chúng ta có thể thu được giải thích bằng cách phân tích các hệ số góc của mô hình.
2. **Local Regression Tree Model:** Các quy tắc quyết định của cây, cùng với tính quan trọng của tính năng, được sử dụng để tạo ra giải thích.
3. **Local Statistical Measures**: Chúng ta có thể tạo ra giải thích bằng cách trích xuất các đo lường thống kê đơn giản từ dự đoán của mô hình ML trên các nhiễu cục bộ. Cụ thể hơn, hãy để $x^* = (x_1, \dots, x_d)$ là trường hợp cần được giải thích. Sau đó, chúng ta tính toán các thống kê tóm tắt về các dự đoán thu được khi tạo ra các biến động trên mỗi chiều x_i. Trong thử nghiệm của chúng tôi, chúng tôi đã chọn sử dụng các đo lường trung bình, trung vị và độ lệch chuẩn.

Các mô hình khác được biết là dễ hiểu cũng có thể được sử dụng. MeLIME được thiết kế để cho phép dễ dàng bao gồm và cấu hình *các mô hình cục bộ có thể giải thích(interpretable local models).*

4. Khả năng giải thích tin cậy (Robustness of Explanations)

Phù hợp mô hình cục bộ phụ thuộc vào số lượng trường hợp được tạo ra xung quanh x^*. Để đảm bảo sự hội tụ và ổn định của mô hình cục bộ g, chúng ta thực hiện chiến lược local-mini-batch. Đó là việc chúng ta tạo ra các trường hợp cục bộ trong khi huấn luyện mô hình cục bộ g cho đến khi nó hội tụ theo các tiêu chí cụ thể được trình bày dưới đây. Tiêu chí khuyến khích các giải thích có *tính mạnh mẽ (robusst)*. Như thống kê ngắn gọn cho các giải thích, chúng ta sử dụng các hệ số của mô hình, là tính quan trọng của tính năng, và các giá trị trung bình của tính năng tương ứng cho Mô hình tuyến tính cục bộ (Local Linear Model), Mô hình Cây Hồi quy cục bộ (Local Regression Tree model) và Các chỉ số Thống kê cục bộ (Local Statistical Measures).

Hãy để $\alpha(g) \in A$ là tập hợp các thống kê ngắn gọn (explanations) được cung cấp bởi mô hình cục bộ $g \in G$. Thủ tục mini-batch hoạt động như sau. Đầu tiên, chúng ta tạo ra một lô các trường hợp có kích thước b xung quanh x^* và fit với mô hình cục bộ của chúng tôi với nó. Hãy để $g_1$là mô hình đã được fit. Sau đó, chúng tôi tính toán $\alpha_1 := \alpha(g_1)$. Tiếp theo, chúng tôi tạo ra b trường hợp bổ sung và phù hợp lại mô hình cục bộ của chúng tôi bằng cách sử dụng toàn bộ các trường hợp. Hãy để g_2 là mô hình đã phù hợp, $\alpha_2 := \alpha(g_2)$, và

$$\delta = \frac{1}{\dim(A)} \left|\left|a_2 - a_1\right|\right|_1.$$

Nếu $\delta < \sigma$ và $\epsilon < \epsilon_c$, trong đó σ và ϵ_c là một giá trị được chỉ định trước, thì tính quan trọng của đặc trưng và độ lỗi của mô hình đã được hội tụ. Nếu không, chúng ta sẽ lặp lại thủ tục này cho đến khi $\delta \leq \sigma$ và $\epsilon < \epsilon_c$.

Mô hình cục bộ sẽ được huấn luyện bằng cách sử dụng local-mini-batch mới được tạo ra cho đến khi đạt được tiêu chí hội tụ cho ϵ và δ.

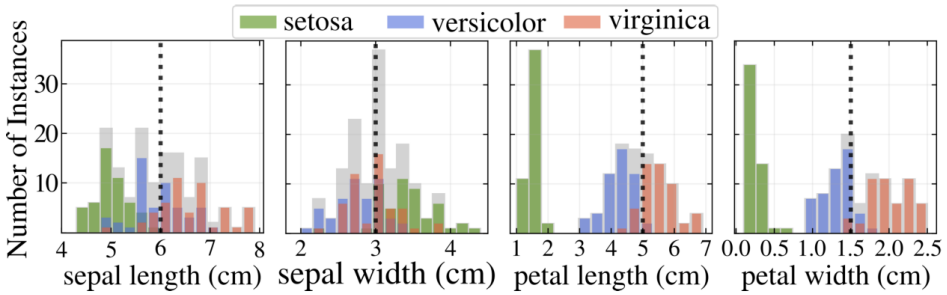
V. CÀI ĐẶT

Bộ dữ liệu: Iris dataset; nhiệm vụ là phân lớp loại hoa diên vĩ.

Black box: mô hình học máy Random Forest.

Mô hình cục bộ: Linear Sparse Model.

Trong phần này, chúng ta phân tích các giải thích được tạo ra cho một mô hình học máy được huấn luyện trên Bộ dữ liệu Iris. Nhiệm vụ là phân loại các mẫu hoa Iris thành ba loài: setosa, versicolor và virginica. Bộ dữ liệu có 150 mẫu với bốn đặc trưng đại diện cho chiều rộng và chiều dài của đài hoa và cánh hoa tính bằng centimet. Chúng ta đại diện một mẫu là x = (độ dài đài hoa; độ rộng đài hoa; độ dài cánh hoa; độ rộng cánh hoa). Trực quan của bộ dữ liệu được hiển thị trong sau.

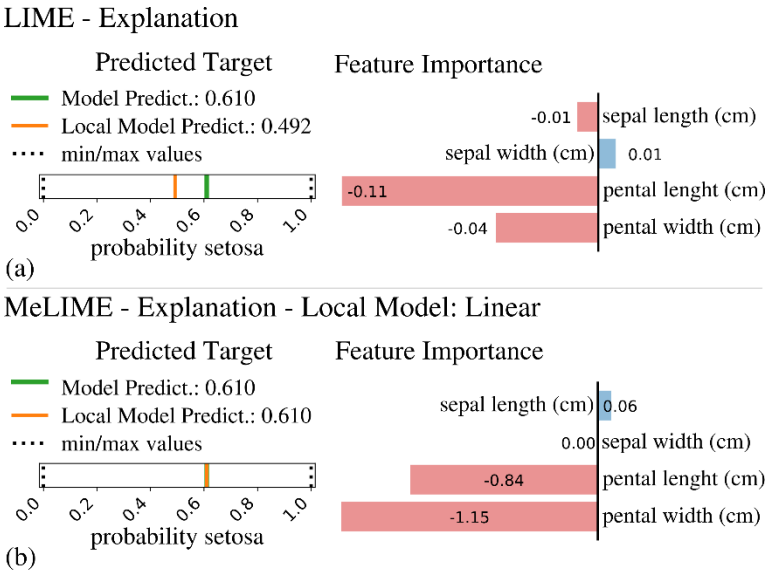


Phân bố của bộ dữ liệu Iris trên bốn đặc trưng và ba lớp. Đường màu đỏ đại diện cho mẫu $x^* = (6.0; 3.0; 5.0; 1.5)$. Các cặp mối quan hệ cho bộ dữ liệu Iris được đưa trong tài liệu bổ sung.

Để thực hiện thí nghiệm của mình, chúng ta chia bộ dữ liệu thành hai tập, tập huấn luyện và tập kiểm tra. Chúng ta ngẫu nhiên chọn 80% và 20% các mẫu lần lượt cho các tập huấn luyện và kiểm tra. Sử dụng tập huấn luyện, chúng ta đã khớp một bộ phân loại Random Forest (RF) được thực hiện trong gói scikit-learn sử dụng các tham số điều chỉnh mặc định. Mô hình đã được khớp có độ chính xác là 0,97 trên tập kiểm tra.

Bây giờ chúng ta phân tích các giải thích được cung cấp bởi cả LIME và MeLIME cho một trường hợp cụ thể $x^* = (6.0; 3.0; 5.0; 1.5)$, được đại diện bằng đường chấm dọc trong hình trên. Với trường hợp x^* này, các giá trị chiều dài và chiều rộng của cánh hoa có thể phân chia hầu hết tất cả các trường hợp thuộc các lớp versicolor và virginica (hình trên), vì vậy chúng ta

mong đợi một giải thích cục bộ sẽ chỉ ra những đại lượng này như là những đặc trưng quan trọng nhất. Hơn nữa, một sự tăng giá trị về chiều dài và chiều rộng của cánh hoa sẽ giảm khả năng của mô hình RF trong việc phân loại một trường hợp nào là iris versicolor.



Giải thích cục bộ cho trường hợp $x^* = (6.0; 3.0; 5.0; 1.5)$ được tạo ra bởi một mô hình RF huấn luyện trên tập dữ liệu Iris. (a) Giải thích LIME; (b) MeLIME sử dụng KDEPCAGen và một mô hình tuyến tính làm mô hình cục bộ.

Dựa vào hình trên cho thấy các giải thích được thu được. Chúng tôi sử dụng chiến lược KDEPCAGen trong MeLIME và hai mô hình cục bộ - mô hình tuyến tính để tạo ra độ quan trọng của đặc trưng. Kết quả thu được bởi MeLIME cung cấp các giải thích có ý nghĩa: giải thích cho độ quan trọng cao hơn cho chiều dài và chiều rộng cánh hoa, điều này phù hợp với phân tích trước đó của chúng ta. Sự khác biệt giữa mô hình black-box và mô hình cục bộ là rất cao (khoảng 0.12), điều này có thể giảm sự tin tưởng của người dùng vào các giải thích được cung cấp bởi phương pháp này. Điều này không đúng với MeLIME, với sự khác biệt tối đa chỉ là 0,04.

VI. THAM KHẢO

[1]. https://arxiv.org/abs/2009.05818

[2]. https://github.com/tiagobotari/melime\

[3]. https://arxiv.org/pdf/1602.04938.pdf

[4]. https://github.com/marcotcr/lime

[5]. https://youtu.be/v6VJ2RO66Ag

[6]. https://www.youtube.com/watch?v=CYl172IwqKs

[7]. https://codelearn.io/sharing/giai-ma-thuat-toan-lime-explainable-ai