# Dự báo sản lượng khai thác hải sản sử dụng các kỹ thuật phân tích chuỗi thời gian


1st Thái Minh Triết
*IS403.N21*
*Trường đại học Công nghệ thông tin*
19522397@gm.uit.edu.vn

2nd Nguyễn Ngọc Hiền
*IS403.N21*
*Trường đại học Công nghệ thông tin*
20520496@gm.uit.edu.vn

3rd Nguyễn Tô Đức Tài
*IS403.N21*
*Trường đại học Công nghệ thông tin*
20520743@gm.uit.edu.vn

4th Nguyễn Thị Kim Liên
*IS403.N21*
*Trường đại học Công nghệ thông tin*
20520909@gm.uit.edu.vn

5th Trần Ngọc Linh
*IS403.N21*
*Trường đại học Công nghệ thông tin*
20521538@gm.uit.edu.vn



***Tóm tắt nội dung*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**



***Index Terms*—component, formatting, style, styling, insert**


## I. Giới thiệu chung

## II. Các công trình nghiên cứu liên quan

wikipedia[1]

## III. Materials

### A. Tổng quan dataset

*1) Thông tin dataset:* Nguồn gốc, thuộc tính

### B. Thống kê mô tả

## IV. Công cụ

## V. Chuẩn bị dữ liệu

### A. Tiền xử lý dữ liệu

### B. Phân chia tập dữ liệu

## VI. Độ đo đánh giá

### A. Mean Squared Error (MSE)

### B. Mean Absolute Error (MAE)

### C. Mean Absolute Percentage Error (MAPE)

### D. R-Squared (R2)

## VII. Phương pháp

VII

[1]wikipedia.com

### A. Linear Regression

Hồi quy tuyến tính (Linear regression) là một phương pháp phân tích thống kê dựa trên việc xác định mối quan hệ giữa hai loại biến bao gồm một biến phụ thuộc (kết quả) và các biến độc lập (dự đoán). Mục đích chính của hồi quy tuyến tính là giúp ta dự đoán được giá trị của biến phụ thuộc dựa trên giá trị của các biến độc lập, kiểm tra xem các biến độc lập có ảnh hưởng thế nào đến giá trị của biến phụ thuộc và các biến độc lập nào là yếu tố quan trọng trong việc dự đoán giá trị của biến phụ thuộc. Hồi quy tuyến tính được sử dụng phổ biến và ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, có thể kể đến như: Y tế, môi trường, giáo dục, kinh tế...

Phương trình của hồi quy tuyến tính đa biến có dạng như sau:

$$Y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \ldots + \beta_i \cdot x_i + \varepsilon$$

Trong đó:
Y: là gía trị đầu ra dự đoán của mô hình.
$x_1$, $x_2$,..., $x_i$: là glà các biến độc lập.
$\varepsilon$: là sai số của mô hình.

### B. Exponential Smoothing

Exponential Smoothing (ETS) là một phương pháp thống kê được sử dụng để mô hình và dự đoán dữ liệu chuỗi thời gian, được phát triển vào cuối những năm 1950 và được cải tiến bởi Holt (1957), Winters (1960). Nó dựa trên kỹ thuật smoothing mà giả định rằng giá trị hiện tại của chuỗi thời gian phụ thuộc vào các giá trị trước đó. ETS được áp dụng rộng rãi trong các lĩnh vực như tài chính, kinh tế học, và quản lý tồn kho.

Mô hình ETS bao gồm ba thành phần chính: trend (T), seasonal (S) và error (E). Trong đó trend (T) thể hiện sự thay đổi giá trị trong chuỗi, seasonal (S) thể hiện cho chu kỳ của dữ liệu, và error (E) là thành phần không thể dự đoán của chuỗi dữ liệu. Theo đó các giá trị T, S, E bao gồm các giá trị như sau:

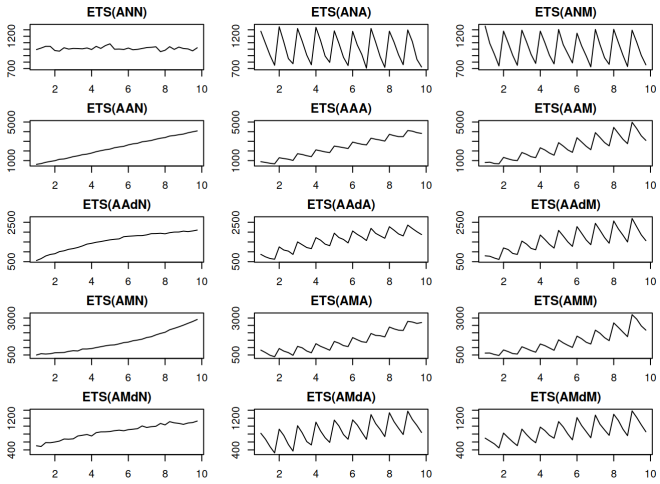
- Error: “Additive” (A), “Multiplicative” (M).

- Trend: “None” (N), “Additive” (A), “Additive damped” (Ad), “Multiplicative” (M), “Multiplicative damped” (Md).
- Seasonal: “None” (N), “Additive” (A), “Multiplicative” (M).

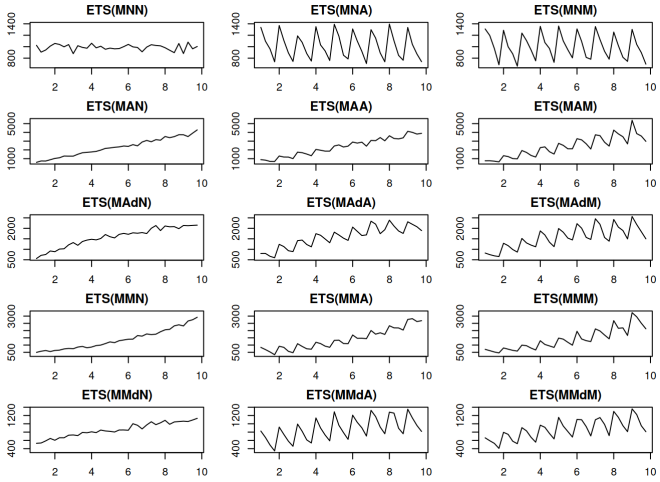
Từ những tham số bên trên, ta có các loại mô hình ETS khác nhau:

| Model | Model | Model |
|---|---|---|
| ETS (M, M, N) | ETS (A ,M, A) | ETS (M, N, M) |
| ETS (M, A, N) | ETS (A, Md, N) | ETS (M, N, A) |
| ETS (M, A, M) | ETS (A, Md, M) | ETS (M, N, N) |
| ETS (A, M, N) | ETS (A, N, A) | ETS (M, A, A) |
| ETS (A, N, N) | ETS (M, Ad, M) | ETS (A, Ad, M) |
| ETS (A, A, M) | ETS (M, Ad, N) | ETS (M, M, A) |
| ETS (M, M, M) | ETS (M, Md, M) | ETS (A, A, A) |
| ETS (A, N, M) | ETS (A, Ad, N) | ETS (A, Ad, A) |
| ETS (A, A, N) | ETS (M, Md, A) | ETS (M, Ad, A) |
| ETS (A, M, M) | ETS (M, Md, N) | ETS (A, Md, A) |

Trong tổng số 30 mô hình ETS bên trên, ta có 15 mô hình Additive error và 15 mô hình Multiplicative error.




Hình 1. Mô hình ETS với Additive error




Hình 2. Mô hình ETS với Multiplicative error

Để xác định mô hình phù hợp nhất trong số 30 mô hình ETS có thể sử dụng một số tiêu chí như Akaike (AIC). Công thức tính AIC như sau:

$$\text{AIC} = -2\left(\frac{LL}{T}\right) + \frac{2t_p}{T}$$

Trong đó:

LL = log likelihood.

$t_p$ = Tổng số tham số.

T = Số lượng quan sát.

### C. K-Nearest Neighbors

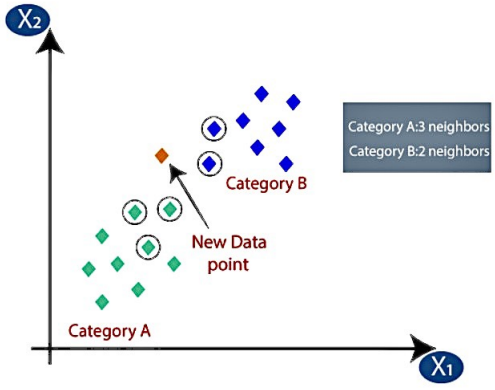
K-Nearest Neighbors (KNN) là một thuật toán học có giám sát, phi tham số. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. KNN có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression.

Ý tưởng của thuật toán là các điểm gần nhau trong không gian đặc trưng có xu hướng thuộc cùng một lớp hoặc có tính chất tương tự. Tham số k trong KNN đề cập đến số điểm được dán nhãn (neighbors) để phân loại.

Cách hoạt động của KNN như sau:

- Bước 1: Chọn tham số K.
- Bước 2: Tính khoảng cách từ điểm dữ liệu đến K số neighbors.
- Bước 3: Lấy K số neighbors gần nhất theo khoảng cách đã tính.
- Bước 4: Trong số K neighbors này, hãy đếm số điểm dữ liệu trong mỗi nhóm.
- Bước 5: Điểm dữ liệu mới sẽ thuộc nhóm có số lượng neighbors nhiều hơn.

Để tính khoảng cách k (k-distance), có thể sử dụng đo khoảng cách Euclidean, Manhattan, Minkowski,...



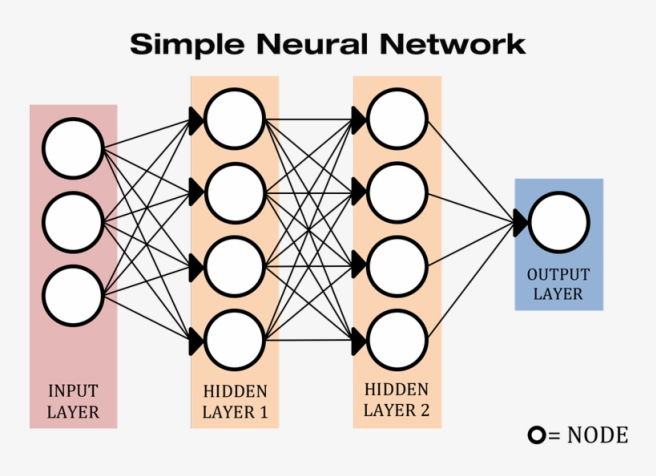

Hình 3. Minh họa cách hoạt động của K-Nearest Neighbors

### D. Recurrent Neural Networks

*1) Khái niệm Neural Network:* Neural Network là một mô hình tính toán lấy cảm hứng từ cấu trúc và hoạt động của các hệ thống thần kinh sinh học. Nó được sử dụng để giải quyết

các vấn đề phân loại, dự đoán và nhận dạng trong lĩnh vực trí tuệ nhân tạo.

Một mạng nơ-ron bao gồm một số nơ-ron được kết nối với nhau. Mỗi nơ-ron nhận đầu vào từ các nơ-ron ở lớp trước, tính toán một hàm kích hoạt (activation function) và chuyển đầu ra cho các nơ-ron ở lớp tiếp theo. Các kết nối giữa các nơ-ron có trọng số và các trọng số này được điều chỉnh trong quá trình huấn luyện của mạng.




Hình 4. Kiến trúc của Neural Network

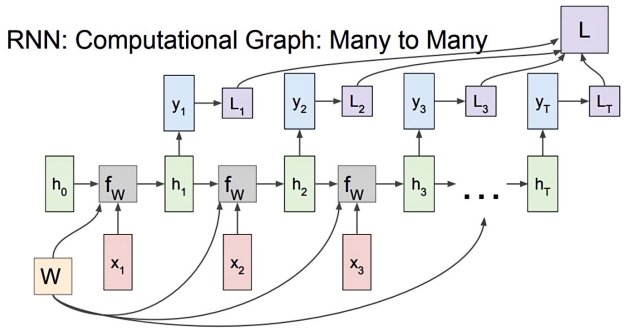
Một mạng nơ-ron có thể có nhiều lớp, bao gồm một lớp đầu vào (Input Layer), một hoặc nhiều lớp ẩn (Hidden Layer) và một lớp đầu ra (Output Layer).

- Lớp đầu vào: nhận dữ liệu đầu vào, chẳng hạn như hình ảnh, văn bản hoặc âm thanh.
- Lớp ẩn: giúp mạng tìm hiểu các tính năng phức tạp từ dữ liệu.
- Lớp đầu ra: tạo ra các kết quả mong muốn

*2) Recurrent Neural Network:* Ý tưởng chính của RNN (Recurrent Neural Networks) là sử dụng chuỗi các thông tin. Trong các mạng nơ-ron truyền thống, tất cả các đầu vào và cả đầu ra là độc lập với nhau. Tức là chúng không liên kết thành chuỗi với nhau. Nhưng các mô hình này không phù hợp trong rất nhiều bài toán.

RNN được gọi là hồi quy (Recurrent) bởi lẽ chúng thực hiện cùng một tác vụ cho tất cả các phần tử của một chuỗi với đầu ra phụ thuộc vào cả các phép tính trước đó. Nói cách khác, RNN có khả năng nhớ các thông tin được tính toán trước đó. Trên lý thuyết, RNN có thể sử dụng được thông tin của một văn bản rất dài, tuy nhiên thực tế thì nó chỉ có thể nhớ được một vài bước trước đó mà thôi.

Nếu như mạng Neural Network chỉ là input layer x đi qua hidden layer h và cho ra output layer y với full connected giữa các layer thì trong RNN, $x_t$ sẽ được kết hợp với hidden layer $h_{t-1}$ bằng hàm $f_W$ để tính toán ra hidden layer $h_t$ hiện tại và output $y_t$ sẽ được tính ra từ $h_t$ , W là tập các trọng số và nó được xuất hiện ở tất cả các cụm, các $L_1$, $L_2$, ..., $L_t$ là các hàm mất mát. Như vậy kết quả từ các quá trình tính toán trước đã được "nhớ" bằng cách kết hợp thêm $h_{t-1}$ tính ra $h_t$ để tăng




Hình 5. Computational Graph Many to Many

độ chính xác cho những dự đoán ở hiện tại. Cụ thể quá trình tính toán được viết dưới dạng toán như sau:

$$h_t = f_W(h_{t-1}, x_t)$$

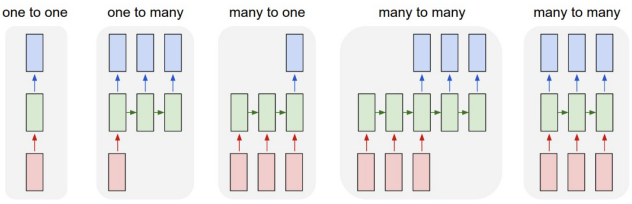
Hàm $f_W$ chúng ta sẽ sử dụng hàm **tanh**, công thức trên sẽ trở thành:

$$h_t = \tanh(\mathbf{W_{hh}h_{t-1}} - \mathbf{W_{xh}x_t})$$
$$\mathrm{y}_t = W_{hy}h_t$$

Đối với mạng Neural Network chỉ sử dụng một ma trận trọng số W duy nhất thì với RNN nó sử dụng 3 ma trận trọng số cho 2 quá trình tính toán: $W_{hh}$ kết hợp với "bộ nhớ trước" $h_{t-1}$ và $W_{xh}$ kết hợp với $x_t$ để tính ra "bộ nhớ của bước hiện tại" $h_t$ từ đó kết hợp với $W_{hy}$ để tính ra $y_t$.

Ngoài mô hình Many to Many như ta thấy ở trên thì RNN còn rất nhiều dạng khác như sau:

- One to One
- One to Many
- Many to One
- Many to Many

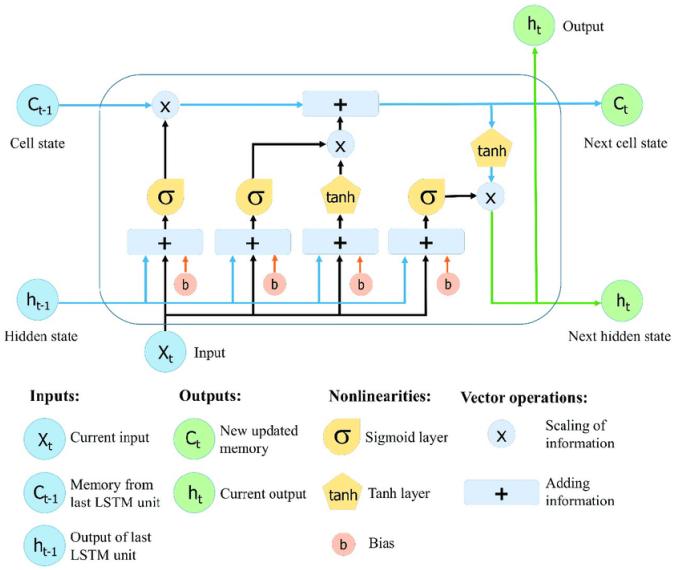



Hình 6. Các dạng khác của mô hình

### *E. Long Short Term Memory Network*

Long Short Term Memory (LSTM) là một loại mạng nơ-ron hồi quy (Recurrent Neural Network) được thiết kế đặc biệt nhằm giải quyết các bài toán về phụ thuộc xa (long-term dependency).

Thành phần quan trọng của LSTM là ô nhớ và các cổng (bao gồm cổng quên, cổng đầu vào và cổng đầu ra). Những cổng này quyết định những thông tin nào cần thêm, xóa và xuất ra khỏi ô nhớ.




Hình 7. Kiến trúc của Long Short Term Memory Network

Cổng quên sẽ xóa những thông tin không còn hữu ích khỏi ô nhớ. Hai đầu vào là $x_t$ (dữ liệu đầu vào tại thời điểm cụ thể) và $h_{t-1}$ ( giá trị đầu ra tại thời điểm trước đó) được đưa đến cổng và nhân với trọng số, sau đó cộng thêm độ lệch, kết quả đầu ra $f_t$ là một số trong khoảng từ 0 đến 1 cho mỗi số trong ô nhớ $C_{t-1}$. Phương trình của cổng quên là:

$$f_t = \sigma(\mathbf{W_f}.[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_f})$$

Trong đó:

$\sigma$ là hàm kích hoạt sigmoid

$W_f$ và $b_t$ lần lượt là trọng số và độ lệch của cổng quên.

Cổng vào sẽ bổ sung những thông tin hữu ích vào ô nhớ. Đầu tiên, thông tin được điều chỉnh bằng hàm sigmoid và lọc các giá trị cần nhớ tương tự như cổng quên sử dụng đầu vào $h_{t-1}$ và $x_t$. Sau đó, một vectơ được tạo bằng cách sử dụng hàm $tanh$ cho kết quả đầu ra từ -1 đến 1, chứa tất cả các giá trị có thể có từ $h_{t-1}$ và $x_t$. Cuối cùng, các giá trị của vectơ và các giá trị quy định được nhân lên để thu được thông tin hữu ích. Phương trình của cổng đầu vào là:

$$i_t = \sigma(\mathbf{W_i}.[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_i})$$
$$C_t = tanh((\mathbf{W_c}.[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_c})$$
$$C_t = \mathbf{f_t} * \mathbf{C_{t-1}} + \mathbf{i_t} * \mathbf{C_t}$$

Trong đó:

$C_{t-1}$ và $C_t$ là ô nhớ lần lượt ở thời điểm t-1 và t $W_c$ và $b_c$ lần lượt là trọng số và tham số của ô nhớ.

tanh là hàm kích hoạt tanh.

Cổng đầu ra: sẽ trích xuất thông tin hữu ích từ ô nhớ hiện tại để trình bày dưới dạng đầu ra. Đầu tiên, một vectơ được tạo bằng cách áp dụng hàm tanh trên ô. Sau đó, thông tin được điều chỉnh bằng cách sử dụng hàm sigmoid và lọc theo các giá trị cần nhớ bằng đầu vào $h_{t-1}$ và $x_t$. Cuối cùng, các giá trị của vectơ và các giá trị quy định được nhân lên để gửi dưới dạng đầu ra và đầu vào cho ô tiếp theo. Phương trình của cổng đầu ra là:

$$o_t = \sigma(\mathbf{W_o}.[\mathbf{h_{t-1}}, \mathbf{x_t}] + \mathbf{b_o})$$

### *F. Autoregressive Integrated Moving Average*

Autoregressive Integrated Moving Average (ARIMA) là một mô hình được sử dụng trong dự báo chuỗi thời gian và chỉ hoạt động tốt nhất nếu dữ liệu phụ thuộc nhiều vào thời gian.

Mô hình ARIMA được chia thành ba phần: quá trình tự hồi quy (AutoRegressive - AR), tích hợp sai phân (Integrated - I) và quá trình trung bình trượt (Moving Average - MA), tương ứng với ba tham số p, d và q là các số không âm.

AR(q) - AutoRegressive là quá trình tìm mối quan hệ giữa dữ liệu hiện tại và dữ liệu quá khứ.

$$y_t = \alpha_0 + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + ... + \alpha_p y_{t-p} + \varepsilon_t$$

Trong đó:

$y_t$ là giá trị chuỗi thời gian tại thời điểm $t$.

$\alpha 0, \alpha 1, ..., \alpha p$ là các tham số tự hồi quy.

$\varepsilon_t$ là sai số ngẫu nhiên tại thời điểm t.

Điều kiện dừng của việc chọn p là:

$$\sum_{i=0}^{p} \alpha_i < 1$$

MA(q) - Moving Average là quá trình tìm mối quan hệ giữa dữ liệu hiện tại và phần lỗi của quá khứ.

$$y_t = \beta_0 + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + ... + \beta_q \varepsilon_{t-q} + \mu_t$$

Trong đó:

$y_t$ là giá trị chuỗi thời gian tại thời điểm $t$.

$\beta 0, \beta 1, ..., \beta p$ là các hệ số trung bình di động.

$\mu_t$ là phần sai số tại thời điểm t.

Điều kiện dừng của việc chọn q là:

$$\sum_{i=0}^{q} \beta_i < 1$$

I(d) - Integrated là hiệu giữa giá trị hiện tại và giá trị trước đó. Quá trình sai phân bậc d của chuỗi được thực hiện như sau:

- Sai phân lần 1:

$$I(1) = \Delta y_t = y_t - y_{t-1}$$

- Sai phân lần 2:

$$I(2) = \Delta(\Delta y_t) = (y_t - y_{t-1}) - (y_{t-1} - y_{t-2})$$

- Sai phân lần d:

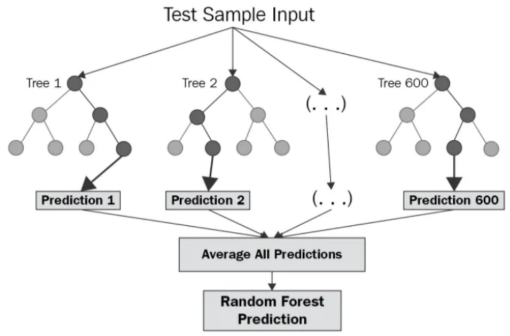
$$I(d) = \Delta(\Delta(...\Delta(y_t)))$$

Phương trình hồi quy ARIMA(q,d,q) được biểu diễn dưới dạng:

$\Delta y_t = \alpha_1 \Delta y_{t-1} + \alpha_2 \Delta y_{t-2} + ... + \alpha_p \Delta y_{t-p} + \beta_1 \varepsilon_{t-1} + \beta_2 \varepsilon_{t-2} + ... + \beta_q \varepsilon_{t-q}$

### G. Random Forest Regression

Random Forest Regression là một thuật toán học có giám sát sử dụng phương pháp ensemble learning trong bài toán hồi quy. Mô hình hoạt động bằng cách kết hợp nhiều cây quyết định độc lập và kết hợp kết quả của chúng để đưa ra giá trị dự đoán cuối cùng.




Hình 8. Kiến trúc của Random Forest

Mô hình xây dựng các cây quyết định bằng cách chia dữ liệu huấn luyện theo phương pháp Bootstrap Sampling. Một phần dữ liệu sẽ được lấy ngẫu nhiên từ tập dữ liệu để làm tập huấn luyện.và phần còn lại được sử dụng làm tập xác thực của mỗi cây quyết định. Một cách tổng quát, mô hình được xây dựng theo từng bước như sau:

Cho tập dữ liệu $D = (x_1, y_1), ..., (x_N, y_N)$, với $x_i = x_{i,1}, ...x_{i,p}$. Lặp $j = 1$ đến $J$:

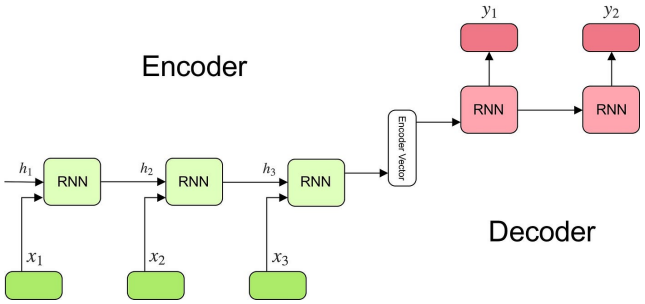
- Bước 1: Lấy mẫu dữ liệu bootstrap $D_j$ từ $D$.
- Bước 2: Sử dụng $D_j$ như dữ liệu huấn luyện. Tiến hành xây dựng cây quyết định sử dụng phương pháp phân chia đệ quy nhị phân (Binary Recursive Partitioning):
  a. Đặt tất cả các mẫu dữ liệu huấn liệu vào một node duy nhất.
  b.Thực hiện đệ quy các bước bên dưới cho các node chưa được phân chia:
  i. Chọn ngẫu nhiên $m$ predictors từ $p$ predictors có sẵn.
  ii. Tìm phân chia nhị phân tốt nhất trên $m$ predictors đã chọn từ bước i.
  iii. Chia node thì hai node con theo phân chia tốt nhất tìm được ở bước ii.
- Bước 3: Giá trị dự báo được tìm bằng cách tính trung bình dự đoán trên tất cả $J$ như sau:

$$\widehat{f}(x) = \frac{1}{J}\sum_{j=1}^{J}\widehat{h}_j(x)$$

Trong đó:
$\widehat{f}(x)$ là hàm dự báo của mô hình tại thời điểm x.
$J$ là số lượng cây
$\widehat{h}_j(x)$ là giá trị dự báo của cây j tại thời điểm x.

### H. Sequence To Sequence

Sequence To Sequence (Seq2Seq) là một loại mô hình trong học máy sử dụng kiến trúc nhiều-nhiều của RNN (Recurrent Neural Network). Được sử dụng rộng rãi trong các tác vụ như dịch ngôn ngữ, tóm tắt văn bản và chú thích hình ảnh, do khả năng xử lý các chuỗi đầu vào và đầu ra có độ dài thay đổi.Mô hình gồm 3 phần: bộ mã hóa (encoder), véc tơ trung gian và bộ giải mã (decoder).




Hình 9. Bộ mã hóa-bộ giải mã trong mô hình trình Squence To Sequence

Encoder:

- Nhiều ô RNN có thể được xếp chồng lên nhau để tạo thành bộ mã hóa. RNN đọc tuần tự từng đầu vào
- Đối với mỗi bước thời gian (mỗi đầu vào) t, trạng thái ẩn (vectơ ẩn) h được cập nhật theo đầu vào tại bước thời gian đó X[i].
- Sau khi tất cả các đầu vào được đọc bởi mô hình bộ mã hóa, trạng thái ẩn cuối cùng của mô hình biểu thị ngữ cảnh/tóm tắt của toàn bộ chuỗi đầu vào.
- Các trạng thái ẩn $h_i$ được tính theo công thức:

$$h_t = \mathbf{f}(\mathbf{W^{(hh)}h_{t-1}} + (\mathbf{W^{(hx)}x_t})$$

Encoder Vector:

- Đây là trạng thái ẩn cuối cùng được tạo ra từ phần mã hóa của mô hình. Được tính bằng công thức trên.
- Vectơ này nhằm đóng gói thông tin cho tất cả các yếu tố đầu vào để giúp bộ giải mã đưa ra dự đoán chính xác.
- Sau khi tất cả các đầu vào được đọc bởi mô hình bộ mã hóa, trạng thái ẩn cuối cùng của mô hình biểu thị ngữ cảnh/tóm tắt của toàn bộ chuỗi đầu vào.
- Vectơ hoạt động như trạng thái ẩn ban đầu cho phần giải mã của mô hình.

Decoder:

- Bộ giải mã tạo chuỗi đầu ra bằng cách dự đoán đầu ra tiếp theo $y_t$ với trạng thái ẩn $h_t$.
- Đầu vào cho bộ giải mã là vectơ ẩn cuối cùng thu được ở cuối mô hình bộ mã hóa.
- Mỗi lớp sẽ có ba đầu vào, vectơ ẩn từ lớp trước $h_{t-1}$ và đầu ra của lớp trước đó $y_{t-1}$, vectơ ẩn ban đầu h.
- Mọi trạng thái ẩn $h_t$ được tính theo công thức:

$$h_t = \mathbf{f}(\mathbf{W^{(hh)}h_{t-1}})$$

- Đầu ra $y_t$ tại thời điểm bước t được tính theo công thức:

$$y_t = \mathbf{softmax}(\mathbf{W^S h_t})$$

Kết quả đầu ra được tính bằng cách sử dụng trạng thái ẩn ở bước thời gian hiện tại $h_t$ cùng với trọng số W(S) tương ứng. Softmax được sử dụng để tạo một vectơ xác suất giúp xác định đầu ra cuối cùng.