Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.00049

MÔ HÌNH DỰ ĐOÁN GEN TƯƠNG QUAN VỚI HỆ THỐNG VẬT CHỦ DÙNG TRONG TÁI TỔ HỢP

Dương Thị Kim Chi[1,2], Trần Văn Lăng[3,2]

[1] Khoa Công nghệ Thông tin, Trường Đại học Thủ Dầu Một
[2] Trường Đại học Lạc Hồng
[3]Viện Cơ học và Tin học ứng dụng, VAST

chidtk@tdmu.edu.vn, langtv@vast.vn

***TÓM TẮT**: Gen mục tiêu là đoạn gen quý được dùng để nhân dòng hay tổng hợp protein trong y học, dược học bằng công nghệ tái tổ hợp. Nhu cầu sử dụng codon của gen mục tiêu trong quá trình dịch mã rất khác biệt giữa các loài nên dự đoán tốt sự tương quan về xu hướng sử dụng codon cho quá trình này sẽ nâng cao hiệu quả biểu hiện cho gen mục tiêu. Với số lượng gen lớn, phân bố codon trong hệ thống biểu hiện không đều (thưa) nên khó xác định mức độ đáp ứng của các hệ thống biểu hiện. Trong phạm vi bài viết chúng tôi đề xuất giải pháp xây dựng mô hình dự đoán tương quan của gen mục tiêu với hệ thống vật chủ biểu hiện dựa trên phương pháp học máy và thuật toán phân lớp kết hợp Extreme Gradient Boosting Mô hình đề xuất cho kết quả tốt hơn về thời gian thực thi và độ dự đoán cũng chính xác hơn so với mô hình được xây dựng bằng Random Forest trên cùng bộ dữ liệu huấn luyện và bộ dữ liệu kiểm thử.*

***Từ khóa**: DNA tái tổ hợp, Relative Synonymous Codon Usage, Gradient Boosting Tree, Extreme Gradient Boosting, Regression Trees, Random Forest.*

I. GIỚI THIỆU

Việc sản xuất các sản phẩm tái tổ hợp thường được bắt đầu bằng việc lựa chọn một gen mong muốn (gen mục tiêu - objecttive gene) lắp vào một *vector tạo dòng*; đồng thời chọn một vật chủ để có thể nhân dòng hoặc được dịch mã thành chuỗi polypeptide; từ đó tạo protein mong muốn [2] còn được gọi là protein tái tổ hợp. Việc chọn lọc được những đoạn gen quý - gen mục tiêu và lựa chọn hệ thống nhân dòng hay hệ thống biểu hiện gen là công việc quan trọng quyết định kết quả cho tái tổ hợp gen mục tiêu này. Môi trường nội bào của vật chủ biểu hiện là nơi cung cấp các nucleotide cần thiết cho quá trình dịch mã và đây cũng là công đoạn quan trọng quyết định sản lượng cũng như chất lượng sản phẩm tái tổ hợp từ gen mục tiêu. Giai đoạn này phụ thuộc rất nhiều vào cấu trúc gen mục tiêu và khả năng đáp ứng nhu cầu codon của tế bào vật chủ - đây là loại tế bào dùng để nhân các phân tử DNA lên nhiều lần.

Các phương pháp thường được dùng để xác định khả năng đáp ứng cho việc biểu hiện gen mục tiêu thường dùng là tìm tập gen biểu hiện cao (High Expression Gene - HEG) [3] của tế bào vật chủ. Dựa vào khả năng biểu hiện của HEG để dự đoán khả năng đáp ứng codon của vật chủ cho gen mục tiêu. Phương pháp được đề xuất để tính HEG cho vật chủ như dùng thống kê được đề xuất bởi Pere Puigbò và cộng sự năm 2007 [6]. Với phương pháp này có thể tóm lượt như sau: (1) Tính giá trị CAI (Codon Adaptation Index) của các gen trong nhóm gen biểu hiện cao thu nhận được từ cơ sở dữ liệu HEG-DB. (2) thống kê khoảng tập trung giá trị CAI nhằm loại bỏ các giá trị cá biệt. (3) dự đoán HEG với lần lượt các giá trị ngưỡng CAI trong khoảng tập trung giá trị CAI từ bước (2). (4) dựa vào độ nhạy (sensitive): Tỉ lệ giữa số gen mã hóa cho Protein Ribosome trong tập gen biểu hiện cao dự đoán được và tổng số gen mã hóa cho Protein Ribosome. Tập gen dự đoán là HEG của vật chủ dựa vào một chỉ số CAI thì cơ hội tìm gen tương quan có ngưỡng thấp hơn mong muốn sẽ bị bỏ qua và làm giảm cơ hội dự đoán mức độ đáp ứng Host với gen mục tiêu. Công thức để tính CAI là:

$$CAI = exp\left(\frac{1}{O_{tot}}\sum_{c\in C}\log w_c\right) \quad (1)$$

Trong đó O_{tot} : tổng số codon trong gen; wc: độ thích nghi tương đối của codon loại c.

Một cách khác dùng dự đoán HEG là dựa vào các thuật toán phân cụm dữ liệu cho hệ gen tế bào Host: (1) Tính chỉ số sử dụng codon đồng nghĩa RSCU (Relative Synonymous Codon Usage) [7] của từng gen. (2) Áp dụng phân cụm các gen dựa trên tiêu chí đặc trưng tương tự nhau, chủ yếu là đặc trưng về xu hướng sử dụng codon dựa trên các gen vốn đã được biết là HEG, được đặt tên là "kernel". (3) Đánh giá một nhóm được phân cụm càng có nhiều kernel càng chứng tỏ nhóm đó càng gần với kernel và kết luận nhóm này có khả năng cao là HEG. Về phương pháp này giúp xác định gen tương quan tốt hơn vì có thể khảo sát đến 61 loại codon biểu diễn cho một gen. Áp dụng việc học không giám sát bằng các phương pháp phân cụm [8] như: Pam, Clara sẽ cho kết quả HEG tốt hơn nhưng chỉ có thể áp dụng riêng cho từng Host.

Khi cần biết gen mục tiêu sẽ phù hợp với loại Host nào thì luôn phải tính lại mức độ tương quan của gen với từng vật chủ. Mô hình xác định gen tương quan hay HEG của vật chủ với gen mục tiêu là cần thiết giúp các nhà sinh học quyết định lựa chọn hệ thống vật chủ phù hợp cho việc triển khai tái tổ hợp cho gen mục tiêu cũng như đánh giá xu hướng sử dụng codon của hệ thống biểu hiện. Với mong muốn dự đoán sự thích nghi của gen mục tiêu với Host nói cách khác là cần *phân lớp dữ liệu để xây dựng mô hình phân lớp từ tập dữ liệu có nhãn (lớp)* đã được định nghĩa trước,

cũng như dự đoán nhãn tự động cho gen mục tiêu. Có nhiều phương pháp được các nhà nghiên cứu về máy học - Machine Learning [14] đề xuất: Classifiation Using Naive Bayes, Classifiation Using Decision Trees and Rules cho việc giải quyết vấn đề phân lớp gán nhãn tự động. Tuy nhiên khi áp dụng cho các bài toán về tin sinh học với số chiều cao, không được bỏ dữ liệu trống thì đây cũng là một thách thức. Các giải pháp thường được áp dụng cho bài toán dạng này là áp dụng thuật toán phân lớp kết hợp - Ensemble algorithms [11] sử dụng cho bộ dữ liệu chiều cao và áp dụng cho dữ liệu thưa. Kỹ thuật thường được dùng Boosting được Jerome H. Friedman [12] và các đồng nghiệp của ông đề xuất là xây dựng tập hợp các mô hình phân lớp yếu để nâng cao hiệu quả của các bộ phân lớp này. Ý tưởng chính của giải thuật này là lặp lại quá trình học của một bộ phân lớp yếu nhiều lần. Sau mỗi bước lặp, bộ phân lớp yếu sẽ tập trung học trên các phần tử bị phân lớp sai trong các lần lặp trước. Các thuật toán áp dụng cho Boosting tiêu biểu như: AdaBoost (Adaptive Boosting), Gradient Boosting. Gần đây Tianqi Chen [12] đề xuất ý tưởng XGBoost (Extreme Gradient Boosting) cho việc nâng cao hiệu quả phân lớp dữ liệu thưa chiều cao dựa trên ý tưởng Gradient Boosting.

Bài viết này đề xuất giải pháp xây dựng mô hình dự đoán tương quan của gen mục tiêu với hệ thống vật chủ biểu hiện dựa trên áp dụng phương pháp học máy và thuật toán phân lớp kết hợp XGBoost. Kết quả từ giải pháp này là mô hình có khả năng dự đoán mối tương quan gen mục tiêu với hệ thống gen vật chủ bằng kỹ thuật gán nhãn tự động cho gen mục tiêu này. Khái niệm liên quan đến bài toán dự đoán gen tương quan và cách thức tổ chức dữ liệu đầu vào cho bài toán cũng như các khái niệm về cây hồi quy và dự đoán được trình bày ở phần II của bài báo. Phần III giới thiệu thuật toán Gradient Boosting các kỹ thuật liên quan và vận dụng các giải thuật này cho việc xây dựng mô hình. Phần IV được dùng để phân tích các kết quả thực nghiệm cũng như so sánh kết quả của mô hình đề xuất với mô hình được xây dựng bằng giải thuật Random Forest trên tập gen của ba loại tế bào vật chủ.

II. CÁC KHÁI NIỆM LIÊN QUAN

2.1. DNA tái tổ hợp

DNA tái tổ hợp là phân tử DNA được tạo thành từ hai hay nhiều trình tự DNA của các loài sinh vật khác nhau. Trong kỹ thuật di truyền, DNA tái tổ hợp thường là được tạo thành từ việc gắn những đoạn DNA có nguồn gốc khác nhau vào trong véctơ tách dòng. Những véctơ tách dòng mang DNA tái tổ hợp này có thể biểu hiện thành các protein tái tổ hợp trong các sinh vật [9].

2.2. Codon đồng nghĩa (Synonymous Condon)

Cơ sở khoa học của việc chọn lựa gen biểu hiện cao dựa trên hiện tượng codon đồng nghĩa. Một codon gồm ba nucleotide sẽ có $4^3 = 64$ codon. Tuy nhiên hiện nay mới chỉ có 20 loại amino acid để mã hoá cho 64 codon này. Như vậy một amino acid có thể được mã hóa bởi ít nhất hai loại codon khác nhau, các codon như vậy được gọi là các codon đồng nghĩa [11].

Chỉ số sử dụng codon đồng nghĩa RSCU:

Đặc trưng cho các gen được sử dụng là chỉ số RSCU của các codon trong gen [6]:

$$r_{ac}(g) = \frac{O_{ac}}{\frac{1}{k_a}\sum_{c \in C_a} O_{ac}} = \frac{O_{ac}}{\overline{O_{ac}}} \quad (2)$$

Trong đó:
r_{ac}: RSCU của codon c mã hoá cho amino acid *a;*
O_{ac}: tần suất xuất hiện của codon c trong trình tự gen;
C_a: tập hợp các codon cùng mã hoá cho amino acid *a;*
K_a: số lượng các loại codon c mã hóa cho amino acid *a.*

Việc tính toán RSCU cho các codon kết thúc (UAA,UGA,UGA) và các codon không có codon đồng nghĩa (AUG - mã hóa cho Methionine, UGG - mã hóa cho Trytophan) là không cần thiết. Do đó, gen được biểu diễn lại bởi một vector RSCU 59 chiều có dạng như sau:

$$r(g)=\{ r_1, r_2, r_3 \ldots r_{59}\}^T \quad (3)$$

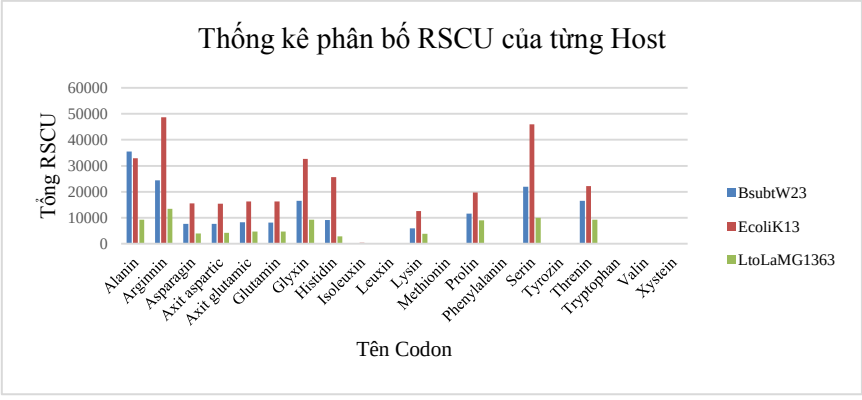
Bài viết sử dụng chỉ số RSCU biểu diễn tập gen của các hệ thống Host thường được dùng trong tái tổ hợp như: E.coli, B.subtilus, Latococcus Lactis và dùng làm tập dữ liệu đầu vào (*tập dữ liệu Training)* để xây dựng mô hình cây kê sơ bộ việc phân bố loại amino acid và số lượng tổng hợp codon của các Host như (hình 1). Trong đó Host Ecoli hồi quy dự đoán gen tương quan tương quan của vật chủ với *gen mục tiêu (tập dữ liệu Testing)*. Việc xây dựng mô hình dự đoán này có thể xem là bài toán phân lớp, gán nhãn cho *gen mục tiêu* dựa vào sự tương tự của xu hướng sử dụng codon của gen mục tiêu và xu hướng đáp ứng của các gen trong các hệ thống Host.

Yêu cầu đặt trưng đối với các bài toán sinh học là tôn trọng dữ liệu tự nhiên không can thiệp bằng các biện pháp hiệu chỉnh cho các giá trị trống, hay điền giá trị trung bình cho những vị trí dữ liệu thiếu. Nên khi tính toán biểu diễn tập dữ liệu Training cho hệ thống bằng công thức (2), chúng tôi giữ nguyên những đặc trưng dữ liệu của từng Host.

Thống có 12/20 loại amino acid và khuyết hai loại codon Lysin_AAG và Threnin_ACG. Host Subtilus có khoảng 13/20 amino acid khuyết Isoleuxin_AUU. Host Latococcus Lactis có 13/20 amino acid cũng bị khuyết

Isoleuxin_AUU. Nhưng đối với Isoleuxin_AUU của Host Ecoli lại có nguồn cung ứng tốt. Với mỗi Host chỉ có khoảng 12 đến 13 loại amino acid được dùng và mật độ phân bố các loại codon trên từng gen cũng khác nhau dao động từ 26-28 loại cho mỗi Host. Như vậy từ ta có thể biểu diễn lại công thức (2) cho bộ dữ liệu đầu vào:

$$r(g)=\{ r_1, r_2, r_3 \ldots r_{28}\}^T \quad (4)$$



Hình 1. Khảo sát tổng quan phân bố các loại Acid amin của từng Host

Do cần giữ nguyên giá trị đặt trưng RSCU cho từng codon (sau này sẽ gọi là đặc trưng hay biến phụ thuộc hay chiều của dữ liệu) nên bài viết áp dụng ma trận thưa (Sparse Matrix) để lưu trữ và dữ liệu cho tập dữ liệu huấn luyện với 1:28 biến phân loại và biến TypeH được dùng làm biến phân lớp, được minh họa (hình 2) với 6 dòng đầu của tập dữ liệu huấn luyện.

```
6 x 29 sparse Matrix of class "dgCMatrix"
  [[ suppressing 29 column names '(Intercept)', 'A4_Alanin_GCA', 'A4_Alani

1 1 3.0000000 1.000000 .         2 1.500000 0.5000000 0.5714286 0.5714286
2 1 0.7741935 1.290323 1.935484 2 1.058824 0.9411765 1.2307692 1.2307692
3 1 0.8135593 1.152542 2.033898 2 1.625000 0.3750000 0.5714286 2.2857143
4 1 0.3809524 2.666667 0.952381 2 1.200000 0.8000000 0.5000000 1.7500000
5 1 .         .        4.000000 2 2.000000 .         .         .
6 1 8.0000000 3.500000 5.000000 2 1.777778 0.2222222 1.6666667 2.0000000
```

Hình 2. Minh họa một phần bộ dữ liệu Host dưới dạng ma trận thưa

2.3. Cây hồi quy-Regression Trees, Tree Ensemble

Với tập dữ liệu đầu vào với n mẫu gen, và m thuộc tính thì $D = \{(x_i, y_i)\}$ (|D| =n, xi ∈R, yi ∈R) tập gen được phân lập từ các Host và biểu diễn thành tập dữ liệu đầu vào có dạng:

$$\mathcal{L}(X_i, Y_i)\ _{i=1}^{N} \quad (5)$$

Trong đó:

$X_i = \{x_1, x_2, \ldots x_p\}$ dùng mô tả các giá trị RSCU của từng loại codon dùng làm tiêu chí mô tả giá trị mức độ đáp ứng của tế bào vật chủ,

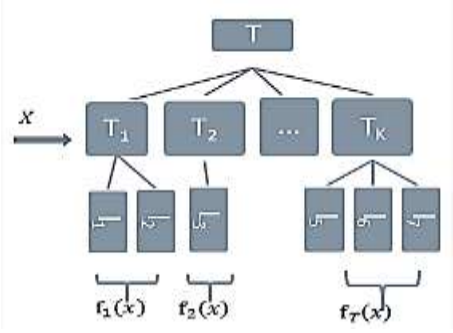
$Y_i = \{y_1, y_2, \ldots, y_k\}$ là tập biến đích gán nhãn cho bộ dữ liệu mới.

Cây hồi quy: là cây được xây dựng *mô hình cây hồi quy* thường đơn giản thông qua việc tách đệ quy theo hàng của tập dữ liệu đầu vào $\mathcal{L}$ thành các tập dữ liệu nhỏ hơn. Việc hình thành nút và lá của cây được tính dựa vào kỳ vọng tối thiểu hàm mất mát [10,13] thường là tổng *hàm lỗi bình phương sai số của nút đang xét.* Tại mỗi lần tách nút, một thuộc tính và giá trị tách của thuộc tính này được chọn để chia nút thành 2 nút con, nút con trái và nút con phải.

Xây dựng mô hình dự đoán dùng cây hồi quy đã xây dựng, cần phải tính toán giá trị wi(xi,θ) cho mỗi mẫu $x_i \in \mathcal{L}$ cho nút tập nút lá của cây. Với dữ liệu X và tập dự đoán Y ta có hàm dự đoán và hồi quy từ cây như sau:

$$\hat{y}_i = \sum_{i=1}^{N} w_i\,(x, \theta)\ Y_i \quad (6)$$

Tree Ensemble: lấy ngẫu nhiên các tập con từ tập huấn luyện T (hình 3), sau đó xây dựng cây quyết định với mỗi tập con đó (T_1, T_2,… T_K). Cây đặc điểm các cây hoàn toàn độc lập với nhau, rồi kết hợp tất cả các cây lại để tạo ra một mô hình dự báo bằng cách lấy trung bình hoặc bình chọn theo số đông các kết quả dự báo từ các cây quyết định nói trên. Mỗi cây kế tiếp được xây dựng bằng cách sử dụng kết quả từ những cây trước đó.



Hình 3. Mô hình cây hồi quy [12]

Hàm hồi quy thu được từ cây hồi quy theo Boosting:

$$\hat{f}(x) = b_1 f_1(x) + b_2 f_2(x) + \ldots + b_T f_T(x) \quad (7)$$

Độ đo mức độ hiệu quả của mô hình dự đoán là hàm hồi quy tổng quát:

$$Y = \beta_1 + \beta_2 X_{2i} + \beta_3 X_{3i} \ldots \beta_k X_{ki} + \varepsilon_i \quad (8)$$

Trong đó:

$X_1, X_2, ..X_k$: các biến phụ thuộc,

$\beta_1, \beta_2, .. \beta_k$: các hệ số hồi quy.

Giá trị sai số dự đoán (Residual) có dạng:

$$\varepsilon = Y - \hat{f}(x) \quad (9)$$

Các giải thuật thường tập trung vào việc nâng cao hiệu của mô hình dự đoán cũng có nghĩa là làm làm giảm quan sát bị dự báo sai hay cụ thể là giảm ε. Phương pháp thường được dùng phương pháp bình phương tối thiểu bài toán hồi quy đa biến còn được gọi là Residual of Squares. Theo đề xuất của Chen and Guestrin [12] mỗi một cây mới sẽ được xây dựng với mục tiêu tối thiểu dần tổng mất mát (Loss) của cây trước đó bằng việc sử dụng phương pháp Gradient descent để hàm số (7) đạt giá trị nhỏ nhất. Với việc kết hợp của K lớp và cây hồi quy (K Classification and RegressionTrees - CART) có dạng $\{T_1(x_i, y_i) .... T_k(x_i, y_i)\}$ cho ra kết quả dự đoán $\hat{y}_i$ bởi việc tổng kết và đánh giá của các hàm từ cây hồi quy khác theo công thức:

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i) \qquad f_k \in F, \quad (10)$$

Trong đó f_k đại diện cho cấu trúc cây hồi quy độc lập với giá trị lá của các cây trong không gian F.

Giá trị của hàm dự đoán có chính xác và hiệu quả được xát định bằng việc thiết lập qua hàm mục tiêu sau:

$$Obj = \sum_i^n l(y_i, \hat{y}_i) + \sum_k^K \Omega(f_k) \quad (11)$$

Trong đó:

l: là độ đo sự khác biệt giữa hàm dự đoán ŷ và nhãn đích y_i của biến quan sát tại I,

Ω: là hàm phạt của mô hình để tránh overfitting, và công thức hàm phạt này có dạng:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j-1}^{T} w_j^2 \quad (12)$$

Với T là số nút của cây, w là tổng điểm số của các nút lá trên cây, hai giá trị λ γ là các hằng số điều chỉnh độ chính xác của mô hình.

2.4. Gradient Tree Boosting

Boosting [12] hoạt động theo cách lấy ngẫu nhiên các tập con từ tập T (hình 3) ban đầu sau đó xây dựng cây quyết định với mỗi tập con đó ($T_1, T_2, ..T_k$) Tại mỗi bước thêm một cây mới, đều kết hợp các "weak learner" để tạo thành một "strong learner" và đều tập trung vào những quan sát bị dự báo sai. Với Gradient Boosting, mỗi một cây mới sẽ được xây dựng với mục tiêu minimizes dần tổng mất mát của cây trước đó bằng việc sử dụng phương pháp Gradient descent. Trong Gradient Boosting[4], mỗi một cây mới được xây dựng với mục tiêu cực tiểu dần tổng mất mát của cây trước đó

bằng việc sử dụng phương pháp Gradient Descent. Việc tính tổng mất mát dựa vào việc lựa chọn loại "Loss function", chẳng hạn như: square loss. Hàm dự đoán tại bước đó sử dụng kết quả dự đoán từ những cây trước đó để quyết định xây dựng cây hiện tại, nên công thức (8) được viết lại [13]:

$$L^{(t)} = \text{Obj} = \sum_i^n l\left(y_i, \hat{y}_{(i)}{}^{(t-1)} + f_t(x_i)\right) \quad (13)$$

Hàm tối ưu của hàm dự đoán ở bước t:

$$\hat{L}^{(t)} = \sum_{i=1}^n [l(y_i, \hat{y}_{(i)}{}^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i)] + \Omega(f_t) \quad (14)$$

Với $g_i = \partial_{\hat{y}_{(i)}{}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ và $h_i = \partial^2_{\hat{y}_{(i)}{}^{(t-1)}} l\left(y_i, \hat{y}^{(t-1)}\right)$ (15)

Như vậy ta có thể viết (11):

$$\hat{L}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i)] + \Omega(f_t) \quad (16)$$

Đặt $I_j = \{i | q(x_i) = j\}$ là tập hợp nút lá của không gian q, kết hợp với công thức (13)**,** thì công thức (16) viết lại

$$\hat{L}^{(t)} = \sum_{i=1}^n [g_i f_t(x_i) + \frac{1}{2} h_i {f_t}^2(x_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j-1}^T w_j^2 \quad (17)$$

$$= \sum_{j=1}^T [(\sum_{i\in I_j} g_i) w_j + \frac{1}{2}\lambda(\sum_{j-1}^T h_i + \lambda) w_j^2] + \gamma T$$

Với w_j^* là trọng số tối ưu được tính cho tập I có dạng

$$w_j^* = \frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda} \quad (18)$$

Và giá trị tối ưu tương ứng tại bước t, của không gian q:

$$\hat{L}^{(t)}(q) = \frac{1}{2} \sum_{j=1}^T \frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T \quad (19)$$

Nếu xem không gian xây dựng cây q gồm tập các nút $I = I_L \cap I_R$, thì ta có cách tách tối ưu tại các nút như công thức (18), và đây cũng có thể cách tính Gain của phương pháp phân lớp bằng cây quyết định.

$$L_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right]\gamma \quad (20)$$

III. THUẬT TOÁN

Việc phân lớp và dự đoán tập gen tương quan với vật chủ Host như đã phân tích ở trên là bài toán phân lớp nhiều lớp với tập dữ liệu huấn luyện có đặc tính thưa và chiều cao. Áp dụng mô hình cây hồi quy [12] để xây dựng mô hình phân lớp và dự đoán thì X = [x_1 , x_2,... x_{28}] là một véctơ hàng chứa thông tin *input*, y là một số biểu diễn output. Công thức hồi quy (7) (8), thuật toán Boosting cho Regression Tree [12] được phát biểu như sau:

Thuật toán 1: Booting cho Regression Tree

Đầu vào: Tập hợp các chuỗi gen X=[x_1 , x_2, .. x_{28}]

Đầu ra: Mô hình dự đoán.

1. **Đặt***:* $\hat{f}(x) = 0$, $r_i = y_i$ cho tất cả các i trong tập training
2. **For** b = 1 **to** B **do**
3. **Fit** cây $\hat{f}^b$ với d splits (d+1 nút trong) cho tập dữ liệu (X, r)
4. **Cập nhật** giá trị $\hat{f}^b$: $\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$
5. **Cập nhật** lại residuals: $r_i \leftarrow r_i + \lambda \hat{f}^b(x_i)$
6. **EndFor**
7. **Output** mô hình Boosting: $\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x)$

Ý tưởng Boosting như đã phân tích ở trên mỗi cây kế tiếp được xây dựng bằng cách sử dụng kết quả từ những cây trước đó. Với Gradient Boosting, mỗi một cây mới sẽ được xây dựng dựa trên giảm thấp nhất tổng hàm Loss **(17)** của cây trước đó bằng việc sử dụng phương pháp Gradient descent [13]. Vận dụng ý tưởng này, Tianqi Chen [12] đề xuất ý tưởng XGBoost cho áp dụng Gradient Boosting cho ma trận thưa với thuật toán Split Finding như sau:

Thuật toán 2: Split Finding

Đầu vào: Tập hợp I, $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$, X

Đầu ra: Giá trị split .

1. **Đặt**: $G \leftarrow \sum_{i \in I} g_i \quad H \leftarrow \sum_{i \in I} h_i$
2. $\hat{f}(x) = 0$, $r_i = y_i$ cho tất cả các i trong tập training
3. **For** k = 1 **to** *m* **do**

//xử lý dữ liệu missing bên nhánh trái

4. $G_L \leftarrow 0, H_L \leftarrow 0$
5. **For** *j* **in** *sorted(Ik, ascent order* $\mathbf{x}_{jk}$*)* **do**
6. $G_L \leftarrow G_L + g_j, \; H_L \leftarrow H_L + h_j$
7. $G_R \leftarrow G - G_L, \; H_R \leftarrow H - H_L$
8. $score \leftarrow \max\left(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
9. **EndFor**

//xử lý dữ liệu missing bên nhánh phải

10. $G_R \leftarrow 0, H_R \leftarrow 0$
11. **For** *j* **in** *sorted(Ik, descent order* $\mathbf{x}_{jk}$*)* **do**
12. $G_R \leftarrow G_R + g_j, \; H_R \leftarrow H_R + h_j$
13. $G_L \leftarrow G + G_R, \; H_L \leftarrow H - H_R$
14. $score \leftarrow \max(\text{score}, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
15. **EndFor**

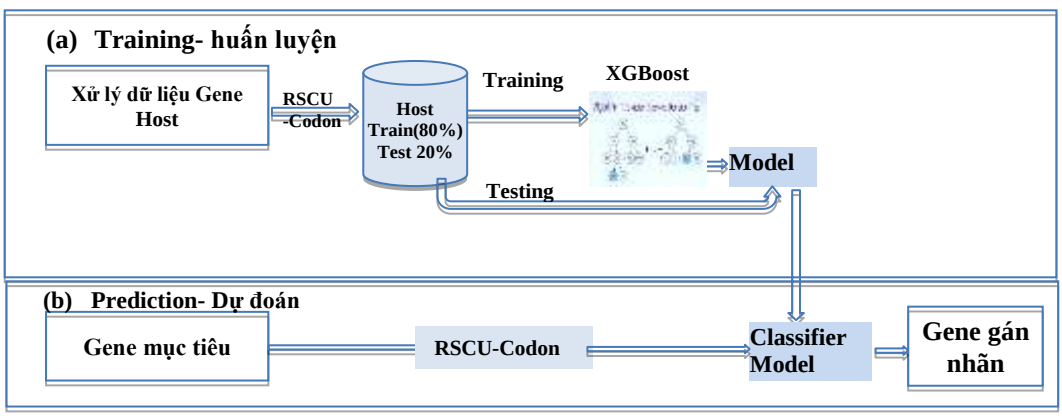
16. EndFor

17. Output hướng và giá trị Split

Với mục tiêu là xây dựng *mô hình học*, bước đầu sẽ khởi tạo với cây hồi quy và hàm lỗi cho trước công thức (14) giải thuật tìm mô hình cực tiểu hóa lỗi hồi quy. Sau đó áp dụng giải thuật Boosting cho Regression Tree: (1) Bước đầu dùng giải thuật dự đoán biến đầu ra ŷ. (2) Tiếp theo lặp lại K lần (số cây hồi quy K là tham số của mô hình) để thực hiện. (3) Tính toán phần dư ε ở công thức và xây dựng mô hình cây hồi quy dùng phần dư ε là biến đích với mục tiêu cực tiểu hóa lỗi. Dự đoán mẫu dùng mô hình cây hồi quy ở bước trước đó. Các ý tưởng xây cây của XGBoost cũng áp dụng tương tự và đối với bài toán dự đoán gen tương quan thì áp dụng Split Finding tại mỗi bước xây cây và tính toán cập nhật ŷ bằng cách thêm các giá trị dự đoán ở lần lặp trước vào các giá trị dự đoán được tạo ra trong bước trước đó.

IV. THỰC NGHIỆM

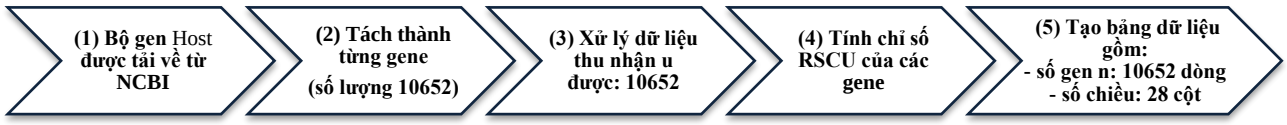
Trong phạm vi bài viết, chúng tôi sử dụng phần mềm Rstudio cùng gói xgboost có chứa các thuật toán về các mô hình hồi quy liệt kê ở mục III đều được tích hợp trong gói phần mềm này, được thử nghiệm trên máy tính với RAM 4 GB, Intel Core i3. Quy trình thực nghiệm như Hình 4, với hai nội dung chính là xây dựng mô hình huấn luyện từ các tập ba hệ gen E.coli, B.subtilus, Latococcus Lactis MG1363. Sau đó dùng mô hình này để dự đoán gen tương quan (gán nhãn dữ liệu)



Hình 4. Sơ đồ xây dựng mô hình dự đoán gen tương quan

4.1. Xử lý dữ liệu thực nghiệm

Bên cạnh *E. coli*, vi khuẩn chủng Bacillus subtilis và Latococcus Lactis cũng là một trong số các hệ thống biểu hiện đang được quan tâm nghiên cứu và sử dụng hiện nay trong lĩnh vực sản xuất protein tái tổ hợp [1]. Quá trình chuẩn bị dữ liệu được mô tả như sau:



Hình 5. Quy trình xử lý dữ liệu tổng quát

Dữ liệu sau khi tải về từ ngân hàng gen quốc tế NCBI, tách thành từng gen. Số lượng mẫu thu được là 10652 gen. Từ trình tự thu được của các gen tính RSCU cho từng gen, tạo bảng dữ liệu cho mẫu gồm 10652 gen và 59 codon như đã mô tả ở trên. Khảo sát sơ bộ dữ liệu phân bố tổng các loại codon của từng vật chủ theo Hình 5, nhận thấy chỉ có hơn phân nửa số trên tổng số 59 loại codon của từng vật chủ có hỗ trợ cho quá trình dịch mã, tổng số loại codon của các vật chủ cũng có sự khác nhau. Do đó tập dữ liệu huấn luyện để theo mô tả ở công thức (2) là bộ dữ liệu thưa có 29 chiều và thuộc ba loại vật chủ

4.2. Xây dựng mô hình dự đoán gen tương quan

Áp dụng gói phần mềm XGBoot [12] tiến hành thực nghiệm trên môi trường R Statictics với giải pháp XGBoot dùng cho dữ liệu thưa như đã mô tả ở phần II và gói phần mềm Random Forest dùng để kiểm chứng kết quả cho mô hình đề xuất. Bài viết đã thực nghiệm từ tập gen 10652 và 28 tiêu chí RSCU Codon với 80% tập gen này dùng làm dữ liệu huấn luyện và 20% dùng để làm dữ liệu kiểm thử mô hình. Khi xây dựng mô hình hồi quy, chúng tôi sử dụng kỹ thuật kiểm tra chéo 5-folds với quy tắc: (1) Cho ngẫu nhiên số lần lập n-round = 10. (2) Thực nghiệm mô hình với XGB.Cv trên tập huấn luyện liệt kê các giá trị hàm mất mát. (3) Chọn giá trị hàm loss thấp nhất. (4) Thực nghiệm điều chỉnh n_round về giá trị nhỏ nhất vừa tìm được, tìm được mô hình hoàn chỉnh. Áp dụng mô hình vừa tìm được trên tập dữ liệu kiểm tra. Xây dựng ma trận hỗn độn - *ConfusionMatrix* để đánh giá mô hình huấn luyện. Với mô hình này chúng tôi gán nhãn cho tập gen mục tiêu (khoảng 21 gen).

Thực nghiệm quy trình trên 9 lần với số n-round dao động từ 10 đến 400 để tìm tập các giá trị hàm mất mát nhỏ nhất và tìm mô hình tốt nhất cho mô hình dự đoán, Bảng 1 trình bày kết quả thống kê quá trình kiểm tra trên n-round như đã nêu.

Bảng 1. Khảo sát thông số cơ bản của các mô hình dự báo

Ten	Loai	N_round	Min.merro.id	Accuracy	95% CI _1	95% CI _2	Kappa
Model1	test	10	10	0,9376	0,9265	0,9475	0,9037
Model1	train	10	10	0,942	0,9369	0,9469	0,9107
Model2	test	20	20	0,9376	0,9265	0,9475	0,9037
Model2	train	20	20	0,942	0,9369	0,9469	0,9107
Model3	test	50	48	0,9376	0,9265	0,9475	0,9037
Model3	train	50	48	0,942	0,9369	0,9469	0,9107
Model4	test	100	89	**0,9639**	0,9265	0,9475	0,9037
Model4	train	100	89	**0.9876**	0,9369	0,9469	0,9107
Model5	test	150	146	**0,9376**	0,9265	0,9475	0,9037
Model5	train	150	146	**0,9920**	0,9899	0,9938	0,9877
Model6	test	200	163	0,9779	0,9708	0,9838	0,9659
Model6	train	200	163	0,9945	0,9927	0,9959	0,9915
Model7	test	250	201	0,9794	0,9724	0,9850	0,9681
Model7	train	250	201	0,9974	0,9961	0,9984	0,996
Model8	test	300	203	0,9789	0,9718	0,9846	0,9674
Model8	train	300	203	0,9974	0,9961	0,9980	0,996
Model9	test	350	232	0,9789	0,9718	0,9846	0,9674
Model9	train	350	232	0,9982	0,997	0,9990	0,9973

Nhận thấy sự khác biệt giữa mô hình huấn luyện và kiểm tra ở Model 4 là nhỏ nhất 0,0237 với n_round = 100 với dự đoán chính xác - khoảng 97%. Model 5 có sự sai khác lớn nhất 0,0544 giữa mô hình huấn luyện và kiểm tra với n_round = 150. Như vậy mô hình được chọn để làm mô hình Classifier Model cho dự đoán gen tương quan sẽ là Model4. Kết quả của mô hình Model4 này như sau:

Confusion Matrix:

```
          Reference
Prediction    1    2    3
         1 3281   12    9
         2    4 3282   23
         3   22   36 1852
```

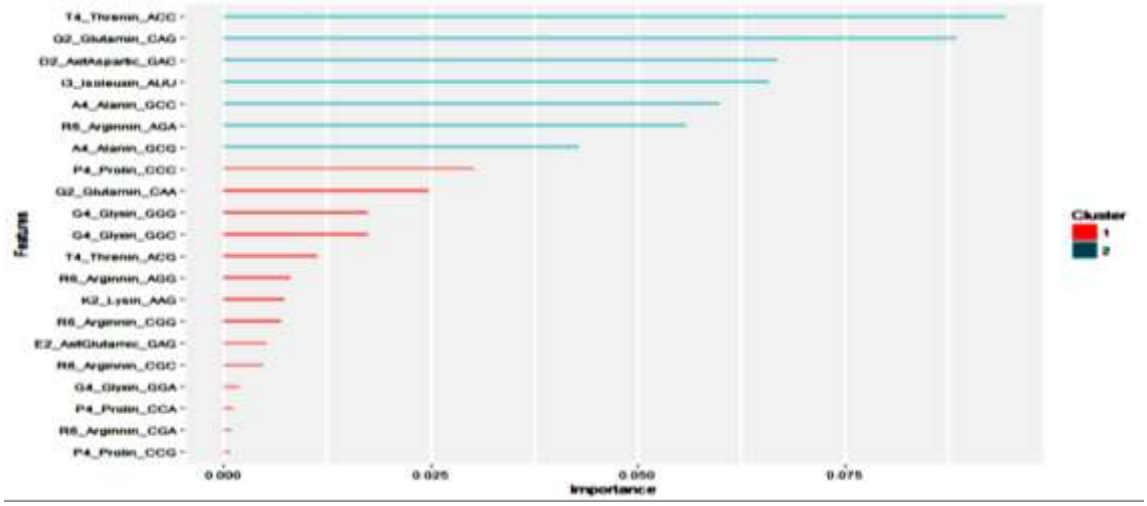
Hình 6a. Confusion Matrix của Model4

```
                Accuracy : 0.9876
                  95% CI : (0.985, 0.9898)
     No Information Rate : 0.3908
     P-Value [Acc > NIR] : < 2.2e-16

                   Kappa : 0.9808
  Mcnemar's Test P-Value : 0.006375
```

Hình 6b. Các thông tin thống kê của Model4

Bảng thống kê danh mục các biến (RSCU codon) của mô hình được trình bày ở Hình 7 với tập các biến được tính dùng để xây dựng mô hình cây hồi quy theo công thức tính đã trình bày ở mục II.



Hình 7. Thống kê các biến quan trọng dùng để xây dựng cây

Random Forest là một thuật toán phân lớp mạnh được đề xuất bởi Breiman [15] và cũng là một trong những phương pháp Ensemble điển hình. Kết quả của giải pháp này cũng tạo ra một tập hợp các cây quyết định, mỗi cây được xây dựng trên tập mẫu Bootstrap với hiệu quả phân lớp chính xác cao và có thể áp dụng phân lớp chiều cao cho dữ liệu thưa được sử dùng rất phổ biến hiện nay. Vận dụng phương pháp Random Forest xây dựng mô hình dự đoán trên cùng bộ dữ liệu đã thử cho mô hình đề xuất để so sánh tính hiệu quả của mô hình đề xuất bằng XGBoost.

Chúng tôi đã tiến hành thực thi Random Forest với phương pháp chia tập dữ liệu huấn luyện và kiểm tra như đã thực nghiệm với XGBoost, cũng áp dụng với cùng số lượng cây là 100. Mô hình dự đoán thu được là *rfhost* (hình 8a) kiểm chứng dự đoán nhãn cho tập 21 gen mục tiêu đã dùng với *Model4* (hình 8b), nhận thấy kết quả dự đoán của hai mô hình là giống nhau hoàn toàn

```
> dgentest<-read_xls("testGene.xls")
> result <- predict(rfhost, newdata=dgentest)
> result
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
 3  1  2  1  2  1  2  1  1  2  3  1  1  1  1  1  1  1  1  1  3
```

Hình 8a. Kết quả gán nhãn cho gen mục tiêu với Random Forest

```
> ##kiem tra tren bo du lieu 21 gen test
> answers<-predict(model,dtestgen)
> print(answers)
 [1] 3 1 2 1 2 1 2 1 1 2 3 1 1 1 1 1 1 1 1 1 3
```

Hình 8b. Kết quả gán nhãn cho gen mục tiêu với XGBoost

Từ kết quả Confusion Matrix của hai mô hình đã thực nhiệm, chúng tôi tính độ chính xác của việc phân lớp đúng cho từng loại Host như mô tả Bảng 2. Nhận thấy với mô hình đề xuất bằng XGBoost có độ chính xác cao hơn khi so sánh cho với các lớp khác nhau của các Host.

Bảng 2. So sánh độ Accuraccy của từng host của hai thuật toán tương ứng với các mô hình dự báo

Tên Host	Random Forest-rfhost	Xgboost-model
Host Ecoli (Class 1)	0,97	0,99
Host 2 B.Subtilus (Class 2)	0,94	0,99
Host 3 Latococcus (Class 3)	0,88	0,97

Tiêu chí thời gian cũng được xem xét cho quá trình thực thi các thuật toán, cụ thể thời gian cần cho mô hình huấn luyện với Random Forest là 30,22 giây, với XGBoost là 22,37 giây. Thời gian cần cho tập dữ liệu huấn luyện gồm 8520 gen cho cả hai thuật toán dao động 22 - 30 giây là có thể chấp nhận cho những ứng dụng thời gian thực.

Kết quả dự đoán nhãn cho tập gen mục tiêu với mô hình dự đoán Model4, với 21 gen mục tiêu được biểu diễn bằng RSCU đã được gắn nhãn phù hợp với thời gian thực thi hợp lý. Các yếu số quan trọng đã được kiểm chứng bằng phương pháp Random Forest cho kết quả tốt hơn về độ chính xác khi phân lớp cũng như về thời gian thực thi.

V. KẾT LUẬN

Bài báo trình bày các trình bày các khái niệm liên quan mô hình hồi quy, cũng như kỹ thuật Gradient Boosting được dùng cho bài toán phân lớp và dự đoán cho loại dữ liệu thưa chiều cao. Dự đoán gen tương quan cho các hệ thống vật chủ dùng trong công nghệ tái tổ hợp là một công đoạn quan trong để xác định các ngưỡng RSCU thích hợp cho từng gen mục tiêu. Các mô hình hồi quy tuyến tính thông dụng như k láng giềng, cây quyết định,... là các thuật toán hỗ trợ tốt cho dự đoán và phân lớp gán nhãn. Tuy nhiên các thuật toán này có khả năng chịu nhiễu thấp và thường phải loại bỏ nhiễu trước cho bộ dữ liệu huấn luyện. Loại bỏ giá trị trống thường làm giảm đi đặc tính sinh học cũng như khả năng dự báo của hệ thống. Gradient Boosting Tree và XGBoost đã được nghiên cứu, phân tích để áp dụng cho xây dựng mô hình dự đoán và gán nhãn. Kết quả thực nghiệm cho thấy mô hình đề xuất có khả năng dự đoán có độ chính xác cao khả năng gán nhãn tốt và hoàn toàn không dùng đến kỹ thuật lọc nhiễu khi tiến hành thực nghiệm. Ngoài ra, độ đo sự quan trọng của các tiêu chí cũng được tính toán từ mô hình và hiển thị trực quan giúp nhà sinh học có thông tin cần thiết để nâng cao khả năng dự đoán mức độ khả thi khi chọn lựa hệ thống vật chủ.

TÀI LIỆU THAM KHẢO

[1]. Alexey Natekin and Alois Knoll, "*Gradient boosting machines, a tutoria"*. Methods article, 2013.

[2]. Hoàng Trọng Phán, Trương Thị Bích Phượng, "*Giáo trình Di truyền học, vi sinh vật và ứng dụng"*. ĐH Huế, 2008.

[3]. Menzella, H.G., "*Comparison of two codon optimization strategies to enhance recombinant protein production in Escherichia coli"*, Microbial cell factories, 2011.

[4]. Tianqi Chen, Tong He Michael Benesty ,Vadim Khotilovich ,Yuan Tang, *Extreme Gradient Boosting*, CRAN, 2017.

[5]. Gupta, S., *"Project report Codon optimization"*, 2003. 2014.

[6]. Pere Puigbo, E. G., Antoni Romeu1 and Santiago Garcia-Vallve, "*A web server for optimizing the codon usage of DNA sequences"* . Nucleic Acids Research, p. W126-W131, 2007.

[7]. Sharp, P. M, Tuohy, T. M, Mosurski, K. R, "*Codon usage in yeast: cluster analysis clearly differentiates highly and lowly expressed gene"*, Nucleic Acids Res , 1987.

[8]. Dương Thị Kim Chi, Trần Văn Lăng, Huỳnh Xuân Hiệp, "*Dự đoán gen biểu hiện cao cho thiết kế gen dùng trong tái tổ hợp"*, Kỹ yếu Hội nghị quốc gia lần thứ IX Nghiên cứu cơ bản và ứng dụng CNTT p134-143, 4-5/8/2016,

[9]. N. A. CampBell, J. B. R. y., L. A Urry, M. L. C, Rain, S. A.Wasserman, P. V. Minorsky, R. B. Jackson, "*Sinh học",* GDVN, p. 10-15, 2014.

[10]. J. Friedman. Greedy, "*Function approximation: a gradientboosting machine"*. Annals of Statistics, 29(5): 1189-1232, 2001.

[11]. A. Carbone, A. Zinovyev,and F. Képès, "*Codon adaptation index as a measure of dominating codon bias"*. Oxford University Press, 2003.

[12]. Tianqi Chen, Carlos Guestrin, "*XGBoost: A Scalable Tree Boosting System"*. KDD '16, San Francisco, CA USA, 100-142, ACM, 2016.

[13] Jerome H. Friedman. "*Greedy function approximation: A gradient boosting machine',* The Annals of Statistics, Vol. 29, No. 5, 1189-1232, 2001.

[14]. Puigbò, P., Guzmán, E., Romeu, A. and Garcia-Vallvé, S. "*OPTIMIZER: a web server for optimizing the codon usage of DNA sequences*", Nucleic Acids Research, 35(suppl 2), W126-W131,(2007).

[15] Leo Breiman, "*Random Forests"*, Statistics Department University of California Berkeley, CA 94720, 2001.

PREDICTING THE CORRELATION OF OPJECTIVE GENE WITH THE HOST SYSTEM USING IN RECOMBINANT TECHNOLOGY

Dương Thi Kim Chi, Tran Van Lang

ABSTRACT: *Objective genes are precious gene that are used to multiply or synthesize proteins in medicine, medicine, and pharmacy by recombinant technology. The need to use the codon of the Objective gene during translation is very different among species, so a good prediction of the correlation of codon use trends for this process enhances the expression efficiency of Objective genes. With the large number of genes, codon distribution in the expression system is sparse, it is difficult to determine the level of response of the expression system. This paper proposes a solution for modeling predictive correlations of Target gene with Host cell systems based on machine learning methods and the Extreme Gradient Boosting; The results from the proposed model when compared to the Random Forest built model on the same training data set and the test data set provide better test results and accurate prediction.*