Chapter 6

Optimizing blood-brain barrier penetrating peptides using explainable AI

The design of drug delivery vehicles to carry medicinal drugs to the central nervous system has always been a challenging task, given the impenetrability of the blood-brain barrier (BBB) that protects the brain. Due to this, the handicap caused by neurological diseases like Alzheimer's disease, Parkinson's disease, etc., leads to a poor quality of life. This chapter proposes an artificial intelligence (AI)-based framework to design novel blood-brain barrier penetrating peptides (B3P2s) that can go through the BBB and deliver drugs to the brain. This framework consists of a metaheuristic-based technique that tries to design novel B3P2s by optimizing the predicted probability value given by a deep learning-based classifier and some properties of B3P2s that are considered desirable by an explainable machine learning-based classifier. To help the wet-lab researchers discover, optimize, and design novel B3P2s, an online tool has been deployed at `https://b3p2design.anvil.app/`.

6.1 Introduction

The growing use of artificial intelligence (AI) in computational biology and bioinformatics has sped up the development of new medicines. There are currently several *in silico* AI-based tools being developed for thoroughly searching the genomes and proteomes of living organisms to find proteins/ peptides (small chains of amino acids that form proteins) and prepare biopharmaceuticals that may be able to treat a wide range of diseases [44, 76, 77, 78, 120]. Experts have been successful in building machine/ deep learning models to find neurological medications to cure ailments that impact our central nervous system (CNS). However, due to the impermeability and strong selectivity of the blood-brain barrier (BBB), which prevents these therapeutic molecules from entering the brain, many diseases still remain incurable. In other words, the BBB's impermeability to all large molecules and about 98% of small molecules, including paclitaxel and adriamycin, presents a substantial challenge for treating life-threatening ailments like brain cancer. To address this issue, scientists have been concentrating on the identification of blood-brain barrier penetrating peptides (B3P2s), which can be used as shuttles to deliver these medications to the brain without compromising the BBB's structural integrity. These peptides are less toxic and more efficient than currently available drugs. There has been some research focused on building *in silico* online tools in the form of web apps for classifying B3P2s. For example, B3Ppredict [19] is a classifier built using the random forests (RF) algorithm, and BBPpred app [20] is a logistic regression (LR)-based classifier built for predicting B3P2s. The B3Pred model [21] was created using RF, LR, and extreme gradient boosting (XGBoost) for the same. Creating supervised machine learning or deep learning-based methods for classifying B3P2s is simple. However, given the dearth of knowledge about the nature of B3P2, it is difficult to identify B3P2s with desirable characteristics that boost their chances of entering the BBB. In other words, specific features or characteristics can be used to identify the best peptides (in terms of BBB penetration), allowing for acceler-

ated synthesis, experimental confirmation, and mass production. These characteristics can be discovered by consulting a subject-matter expert, which is time-consuming as he/ she would need to analyze each B3P2 separately and in relation to other B3P2s. Alternatively, AI can also be applied to achieve the same results.

The work presented in this chapter attempts to model the task of designing B3P2s with good BBB penetration as a multi-objective optimization (MOO) problem. A machine learning (ML)-based B3P2 classifier (ML-B3P2pred) has been built and interpreted using explainable AI (XAI) to identify the objectives for this MOO problem. XAI helps in decoding the models built using machine/ deep learning techniques. Various tools, such as SHAP [121], Captum [112], etc., are used for this purpose [122]. The features that have a reasonable contribution to the results given by ML-B3P2pred are used as objectives of the formulated MOO problem. Furthermore, the score or predicted probability value (PPV) given by a novel interpretable bidirectional encoder representation from transformers (BERT)-based B3P2 classifier called DL-B3P2pred has been used as the primary objective. After that, novel B3P2s with desirable features were designed by solving this MOO problem using a novel population-based search technique called the hybridized gravitational search algorithm (HyGSA). Also, the DL-B3P2pred has been used to highlight the crucial amino acids (AAs) of a B3P2 so that they are preserved when a particle or peptide is undergoing modifications.

To the best of our knowledge, this is the first work of its kind. The main contributions to this work are as follows.

- This work proposes hybridized GSA (HyGSA), a novel population-based search algorithm, to solve a novel MOO problem formulated to design optimal B3P2s.
- A fitness function has been formulated using four objectives determined using a novel explainable DL-based classifier (DL-B3P2pred) and an explainable ML-based classifier (ML-B3P2pred).
- The DL-B3P2pred is also used to determine the important AAs in a peptide so

that HyGSA does not alter them.

- A novel method has been developed to sample the population input to the HyGSA to save computational time and costs. It draws inspiration partially from a variant of the subset-sum problem [41].
- Non-dominated sorting is performed based on the fitness value to output a set of potent B3P2s.
- A freely accessible online tool has been deployed at `https://b3p2design.anvil.app` to help the researchers find optimal B3P2s with desirable traits in a protein sequence.

The rest of this chapter is organized as follows. Section 6.2 presents the techniques and the dataset used in this work. The proposed work is elucidated in section 6.3. The results and outcomes have been elaborated in section 6.4. The concluding remarks and future works are discussed in section 6.5.

6.2 Data and preliminaries

6.2.1 Dataset

The B3P2 sequences present in the dataset consist of the annotated B3P2s recorded in public repositories such as B3Pdb [123], Brainpeps [124], and BBPpred [20]. The negative dataset (non-B3P2s) has been built by querying the Swissprot database using the following keywords: "length: [5,50], NOT blood-brain barrier NOT brainpeps NOT b3pdb NOT permeability NOT transport NOT transfer NOT neuro AND reviewed: yes". Then, the peptides with non-standard AAs were removed (containing AAs represented by B, J, O, U, X, and Z). Following this, the CD-HIT [95] tool was used to remove the sequences with more than 90% similarity to avoid any performance bias. As a result, 425 B3P2s and 425 non-B3P2s comprise the dataset.

6.2.2 Gravitational search algorithm (GSA)

The gravitational search algorithm (GSA) is a nature-inspired MOO technique. It solves MOO problems by drawing an analogy to the laws of physics. A population of solutions (particles) interacts to output a near-optimal solution in the search space. The fitness value (determined by the fitness function) helps determine the desirability of each solution. GSA has performed well on various problems like feature selection, networking, pattern recognition, etc. It is remarkable in handling complex, high-dimensional search spaces and can easily escape getting stuck at a local optimum. Another advantage is its ability to find a good exploration-exploitation trade-off. GSA uses the gravitational constant to control the gravitational force between particles, which helps the algorithm converge to a good solution even with uncertain objectives. The working of GSA can be mathematically expressed using Eqs. 6.1-6.9.

$$Pop = \{X_1, X_2, X_3, \cdots, X_N\} \tag{6.1}$$

$$X_i = \{x_i^1, x_i^2, x_i^3, \cdots, x_i^d\} \text{ for } i = 1, \cdots, N \tag{6.2}$$

$$R_{X_i,X_j}(k) = ||X_i(k), X_j(k)||_2 \tag{6.3}$$

$$G(k) = G_0 \times (\frac{k_0}{k})^\gamma \tag{6.4}$$

$$F_{i,j}^d(k) = G(k) \times \frac{M_i(k) \times M_j(k)}{R_{i,j}(k) + \epsilon} \times (x_j^d(k) - x_i^d(k)) \tag{6.5}$$

$$F_i^d(k) = \sum_{j=1, j\neq i}^{N} r_j \times F_{i,j}^d(k) \tag{6.6}$$

$$a_i^d(k) = \frac{F_i^d(k)}{M_i(k)} \tag{6.7}$$

$$vel_i^d(k+1) = vel_i^d(k) + a_i^d(k) \tag{6.8}$$

$$x_i^d(k+1) = x_i^d(k) + vel_i^d(k+1) \tag{6.9}$$

Algorithm 3 Gravitational search algorithm

Input: Pop

Output: X_{Best}

procedure GSA(Pop)
  Randomly initialize a population Pop of N particles
  $N = |Pop|$
  for $k = 1$ to k_0 **do**
    Calculate $G(k)$ using Eq. 6.4
    for $i = 1$ to N **do**
      Calculate $Fit_i(k)$
      Calculate $M_i(k)$
      for $d = 1$ to D **do**
        Compute $F_i^d(k)$ using Eq. 6.6
        Compute $a_i^d(k)$ using Eq. 6.7
        Compute $vel_i^d(k+1)$ using Eq. 6.8
        Compute $x_i^d(k+1)$ using Eq. 6.9
      end for
    end for
  end for
  $Best = \{j | fit_j = min_{i \epsilon [1,\cdots,N]}(fit_i)\}$
return X_{Best}
end procedure

Here, the i^{th} particle in a population Pop of N particles is represented by X_i. The Euclidean distance between X_i and X_j is given by $R_{i,j}(k)$, and $G(k)$ is the gravitational constant for the i^{th} iteration (γ is its decay rate). The force applied by X_j on X_i in the d^{th} dimension is given by $F_{i,j}^d(k)$, and $F_i^d(k)$ and $a_i^d(k)$ are the net force and acceleration

of the i^{th} particle in the d^{th} dimension. The acceleration is calculated using the mass of X_i (given by $M_i(k)$). The $vel_i^d(k+1)$ and $x_i^d(k+1)$ represent the new velocity and position of X_i in dimension d. Also, a random number r_j (from the interval [0,1]) is chosen to decide the extent of the force exerted by X_j on X_i. The GSA begins with the population initialization, followed by the fitness, mass, force, acceleration, and velocity calculations of each particle under consideration. After this, the updated position of each particle is determined (as explained in Algorithm 3), and the entire procedure is repeated until convergence or for a predetermined number of iterations (k_0).

6.3 Proposed work

This work has been divided into two phases as illustrated in Figure 6.1. Phase-1 consists of building the explainable machine/deep learning-based B3P2 classifiers, which will be used in phase-2 to complete the HyGSA framework.

6.3.1 Phase-1: Construction of ML-B3P2pred and DL-B3P2pred

In this section, the ML-B3P2pred and DL-B3P2pred models have been described. These models are interpreted using XAI frameworks such as SHAP, ELI5 [125], Yellowbrick [126], and Captum. Both of these models have been explained using Figure 6.2.

6.3.1.1 The explainable ML-B3P2pred model

The compositional, physicochemical, and structural features of the peptides present in the dataset were computed prior to creating the ML-B3P2pred model. These characteristics included the isoelectric point (pI), molecular weight (mw), instability index, net charge, GRAVY index, aromaticity, alpha-helix propensity, beta-sheet propensity, beta-turn propensity, as well as the amino acid composition (AAC) for each standard amino acid.

The ML-B3P2pred model has been built using the XGBoost algorithm (a supervised

ML algorithm that uses boosting to combine several weak models to form a single strong model). To train and test the ML-B3P2pred model, the dataset has been divided into training (90%) and test sets (10%). Then, the GridSearchCV method has been used to find the best hyperparameters. After that, the model is explained using tools like ELI5, SHAP, and Yellowbrick. The explainability techniques are used to identify a collection of features that are essential for predicting a peptide as BBB penetrating. The fitness function of the multi-objective optimization problem solved in this study (as explained in the following sections) is formed using these features, which make a B3P2 desirable.

6.3.1.2 The explainable DL-B3P2pred model

Some of the most reliable pre-trained language models, such as BERT, are very efficient in sequence modeling. This study used a pre-trained BERT model with two dense layers and an output layer stacked after it. This model's predicted probability value (PPV) served as the main objective of the fitness function that HyGSA optimized. Also, the model is explained using Captum, which quantifies the contribution (positive or negative) of each AA residue in predicting a peptide as BBB penetrating. This helps determine the presence of which AAs are important in a B3P2, enabling the HyGSA to forego updating or replacing those AAs.

6.3.2 Phase-2: The Hybridised Gravitational Search Algorithm (HyGSA)

In this section, the proposed HyGSA framework has been elucidated. This includes the methodology used to encode the particles, the fitness function formulated in this study, the technique used for sampling the population, and the manner in which the explainable tools built in phase-1 are integrated into the proposed work.

6.3.2.1 Particle representation

The population consists of particles that represent the peptides. The d^{th} dimension of a particle X_i, is denoted by x_i^d, and it represents the d^{th} AA in the peptide pep_i

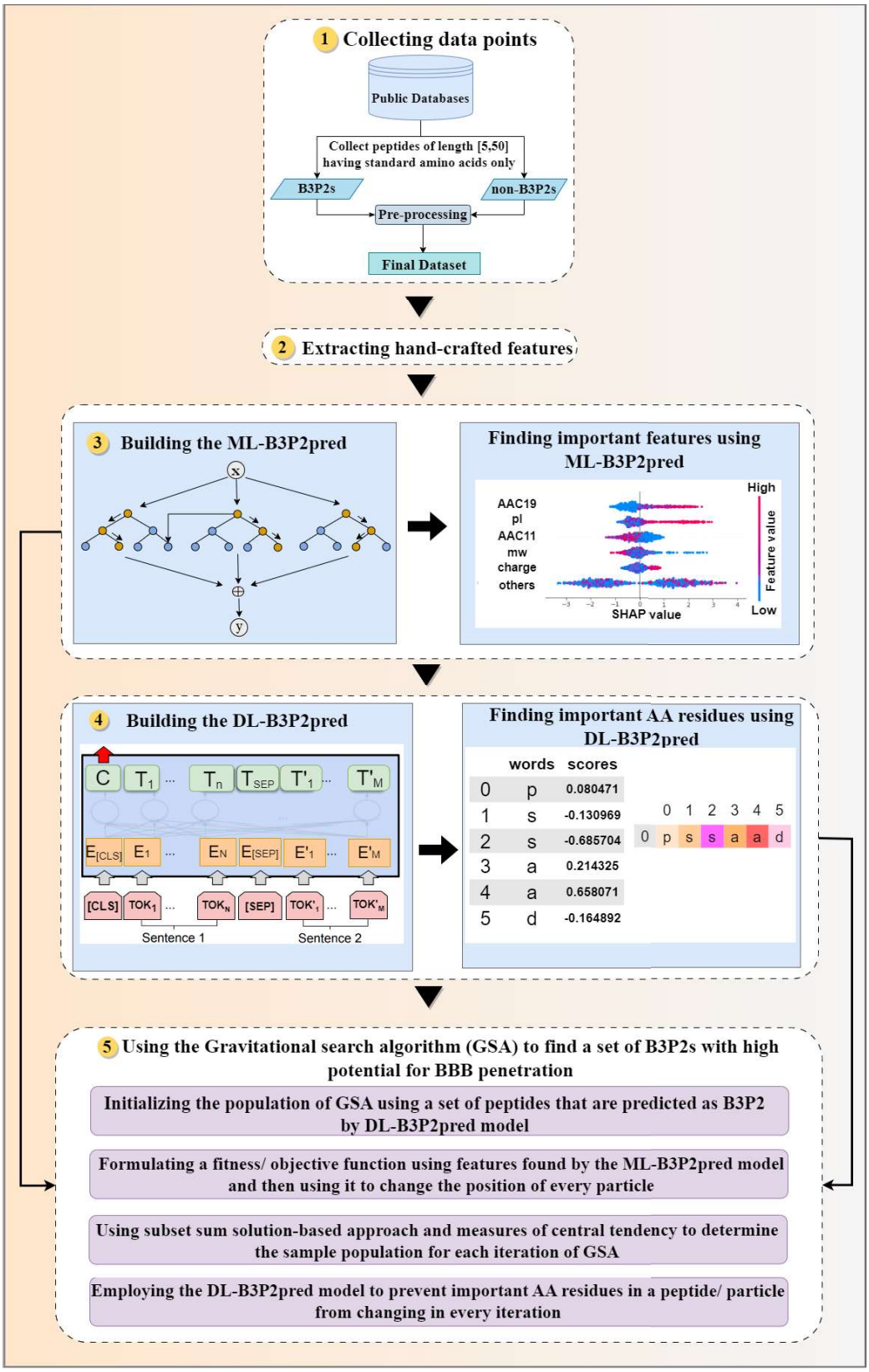


Figure 6.1: The overall process involved in building HyGSA

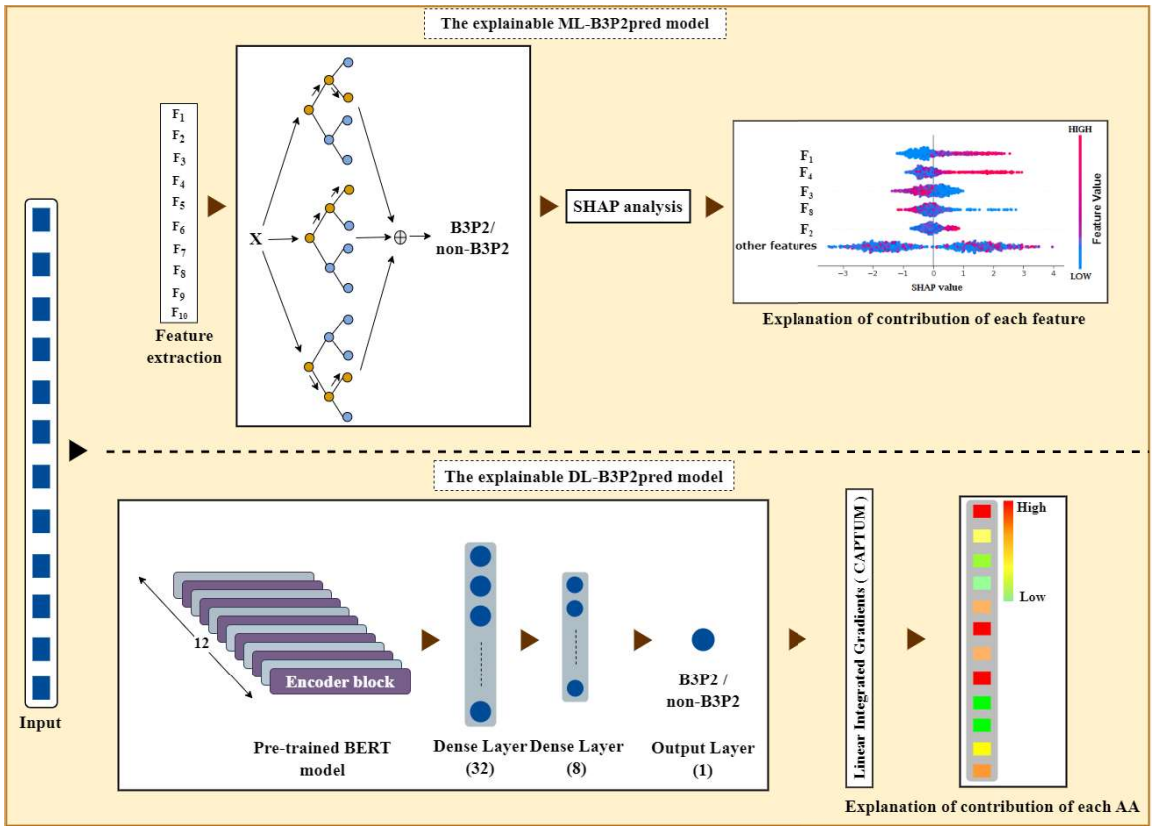


Figure 6.2: Architecture of ML-B3P2pred and DL-B3P2pred models

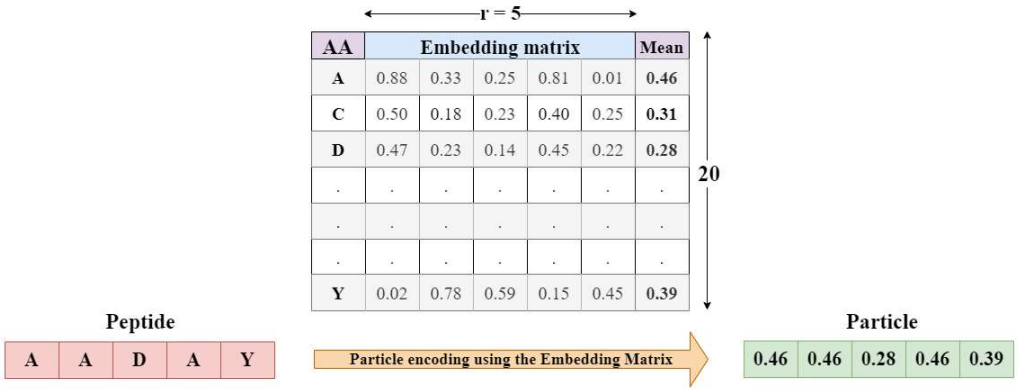


Figure 6.3: Particle encoding

(represented as pep_i^d). The skip-gram algorithm has been employed to find embedding vectors for each AA residue before encoding the particles. In order to calculate x_i^d, the embedding vector for pep_i^d is averaged, as given by Eq. 6.10. This procedure has been

illustrated using Figure 6.3.

$$x_i^d = \frac{\sum_{j=1}^{r} e_j(pep_i^d)}{r} \tag{6.10}$$

Here, $E_{r \text{ X } 20}$ denotes the matrix of 20 embedding vectors, each of length r (one for each standard AA). The term $e_j(pep_i^d)$ denotes the numerical value at the embedding vector's j^{th} position corresponding to the AA residue pep_i^d of a given peptide pep_i.

6.3.2.2 Problem formulation

The optimization of four objectives, which are enumerated and explained as follows, has been considered in this work.

1. **Predicted Probability Value (PPV):** The DL-B3P2pred model's predicted probability value (PPV) for each peptide present in the population is the first objective. It denotes the model's confidence in the BBB penetrability of a particular peptide. The higher the PPV, the more desirable the given peptide.
2. **Amino acid composition of tyrosine (AAC_Y):** The presence or absence of AAC_Y plays an important role in BBB penetration (as found in the experiments conducted in this study, explained in the upcoming sections). The AAC_Y if the i^{th} peptide can be calculated as per Eq. 6.11.

$$AAC\_Y_i = \frac{numY(pep_i)}{l_i} \tag{6.11}$$

   Here, pep_i is a peptide of length l_i, and $numY$ is a function that counts the total number of tyrosine (Y) in pep_i.
3. **Isoelectric point (pI):** At its isoelectric point (pI), a peptide is the least soluble, with a high tendency for aggregation. It can be calculated using the AA sequence of a peptide. To normalize the pI values, min-max normalization has been used as given by Eq. 6.12 (where npI_{min} and npI_{max} are the minimum and maximum

possible pI values for a given peptide length).

$$npI_i = \frac{(pI_i - pI_{min})}{(pI_{max} - pI_{min})} \tag{6.12}$$

4. **Molecular weight (mw):** As per [119], the peptides that have a low molecular weight (mw) are more stable than bulky ones. Also, the heavier peptides have less BBB penetrating potential (explained later in this chapter). The molecular weight of a peptide (mw_i) can be normalized using Eq. 6.13, where mw_{min} and mw_{max} are the minimum and maximum molecular weights for a given length.

$$nmw_i = \frac{(mw_i - mw_{min})}{(mw_{max} - mw_{min})} \tag{6.13}$$

As explained later in the upcoming sections, the peptides with high PPV, high npI, high AAC_Y, and low nmw have more BBB penetrability. So, the fitness function of the optimization problem solved by HyGSA in this work and the corresponding constraints are as given in Eqs. 6.14 and 6.15, respectively.

$$\textbf{Minimize: } fit(i) = \alpha*(1-PPV_i)+\beta*(1-AAC\_Y_i)+\gamma*(1-npI_i)+\delta*nmw_i \tag{6.14}$$

$$\text{Subject to: } \alpha+\beta+\gamma+\delta = 1 \textbf{ and } \alpha > \beta > \gamma > \delta > 0 \tag{6.15}$$

The fitness function has been reduced as given from Eq. 6.16 to 6.16.

$$\begin{aligned} & fit_i = \alpha * (1 - PPV_i) + \beta * (1 - AAC\_Y_i) + \gamma * (1 - npI_i) + \delta * nmw_i \\ \implies & fit_i = \alpha - \alpha * PPV_i + \beta - \beta * AAC\_Y_i + \gamma - \gamma * npI_i + \delta * nmw_i \\ \implies & fit_i = (\alpha + \beta + \gamma) - (\alpha * PPV_i + \beta * AAC\_Y_i + \gamma * npI_i) + \delta * nmw_i \end{aligned} \tag{6.16}$$

Using the values from Eq. 6.15.

$$\begin{aligned} & \alpha + \beta + \gamma + \delta = 1 \\ \implies & \alpha + \beta + \gamma = 1 - \delta \end{aligned} \tag{6.17}$$

Using the values from Eq. 6.17 in Eq. 6.16.

$$\begin{aligned} \implies & fit_i = (1 - \delta) - (\alpha * PPV_i + \beta * AAC\_Y_i + \gamma * npI_i) + \delta * nmw_i \\ \implies & fit_i = 1 - \delta(1 - nmw_i) - (\alpha * PPV_i + \beta * AAC\_Y_i + \gamma * npI_i) \end{aligned} \tag{6.18}$$

So, the final fitness function used in this work is as follows.

$$\textbf{Minimize } fit_i = 1 - \delta(1 - nmw_i) - (\alpha * PPV_i + \beta * AAC\_Y_i + \gamma * npI_i) \tag{6.19}$$

6.3.2.3 Working of the HyGSA

The proposed framework has been described using Algorithms 4-7. It begins with the $HYBRIDIZED - GSA$ function (Algorithm 4). The particles are formed from peptides using Eq. 6.10 and taken as the initial population. Then, some steps are repetitively performed for k_0 iterations (lines 4-26). The gravitational constant $G(k)$ is calculated using Eq. 6.5 for each iteration. In HyGSA, a representative sample of the population (*Sample_Pop*) consisting of particles with high, low, and average fitness is selected to save time and computational costs using the $SELECT$ function (given in Algorithm 5). The HyGSA uses a mechanism that finds the elements to solve the subset-sum problem [41] and a technique that considers the population as a normal distribution and uses its mean and standard deviation for selecting the particles of *Sample_Pop*. Note that the subset-sum problem consists of finding whether or not a subset of a set whose sum of elements is equal to a given value exists. It is a decision problem, but in this chapter, the Algorithm 5 tries to find such a subset. For this, the fitness of all particles is calculated and summed together

(cum_fit) (lines 7-14, Algorithm 5). After that, the function chooses a random number r_1 lying in [$\frac{cum\_fit}{2}$,cum_fit]. Then, one by one, the particles are selected by generating a random number r_2 lying in {1,N}. The r_2^{th} particle in the Pop is placed in the $Sample\_Pop$, and its fitness value is added to the sum_fit (until it becomes greater than r_1) (lines 15–19, Algorithm 5).

Algorithm 4 Using GSA to find B3P2s

Input: $Pop, Num, DL - B3P2pred$

Output: $B3P2s$

1: **procedure** HYBRIDIZED-GSA($Pop, Num, DL - B3P2pred$)
2: $temp \leftarrow Num$
3: $N = |Pop|$
4: **for** $k = 1$ to k_0 **do**
5: Compute $G(k)$ using Eq. 6.4
6: $Sample\_Pop = \text{SELECT}(Pop, N)$
7: $N_{SP} = |Sample\_Pop|$
8: **for** $i = 1$ to N_{SP} **do**
9: $pep_i = \text{CONVERTTOPEP}(X_i)$
10: $PPV_i =$ Compute PPV of pep_i using DL-B3P2Pred
11: $AAC\_Y_i =$ Compute AAC_Y of pep_i Eq.6.11
12: $npI_i =$ Compute npI of pep_i Eq.6.12
13: $nmw_i =$ Compute nmw of pep_i Eq.6.13
14: $fit_i = 0.5 \times (1 - PPV_i) + 0.25 \times AAC\_Y_i + 0.15 \times npI_i + 0.1 \times nmw_i$
15: $M_i = (\frac{fit_i}{\sum_{l=1}^{N_{SP}} fit_l})$
16: $LIG(pep_i) =$ Compute LIG using DL-B3P2pred
17: **for** $j = 1$ to N_{SP} **do**
18: **if** $j \neq i$ **then**
19: Compute $R_{i,j}$ using Eq. 6.3
20: **end if**

21: **for** $d = 1$ to D **do**
22: Compute $F_{i,j}^d$ using Eq. 6.5
23: **end for**
24: **end for**
25: Compute F_i^d using Eq. 6.6
26: Compute a_i^d using Eq. 6.7
27: **for** $d = 1$ to D **do**
28: **if** $LIG_d(pep_i) \leq 0$ **then**
29: Compute vel_i^d using Eq. 6.8
30: Compute x_i^d using Eq. 6.9
31: **end if**
32: **end for**
33: **end for**
34: **end for**
35: $fronts = \text{SORT}(Pop, N)$
36: **for** $i = 1$ to $|fronts|$ **do**
37: **if** $|front[i]| < temp$ **then**
38: $B3P2s = B3P2s \cup front[i]$
39: $temp = temp - |front[i]|$
40: **else**
41: **for** $j = 1$ to $temp$ **do**
42: $B3P2s = B3P2s \cup front[i][j]$
43: **end for**
44: **end if**
45: **end for**
46: **return** $B3P2s$
47: **end procedure**

Algorithm 5 Sample population selection

Input: Pop, N

Output: $Sample\_Pop$

procedure SELECT(Pop, N)
  $Sample\_Pop \leftarrow \phi$
  $sum\_fit \leftarrow 0$
  $cum\_fit \leftarrow 0$
  $\mu\_rem \leftarrow 0$
  $\sigma\_rem \leftarrow 0$
  for $i = 1$ to N **do**
    $pep_i =$ ConvertToPep(X_i)
    $PPV_i =$ Compute PPV of pep_i using DL-B3P2Pred
    $AAC\_Y_i =$ Compute AAC_Y of pep_i Eq.6.11
    $npI_i =$ Compute npI of pep_i Eq.6.12
    $nmw_i =$ Compute nmW of pep_i Eq.6.13
    $fit_i = 0.5 \times (1 - PPV_i) + 0.25 \times AAC\_Y_i + 0.15 \times npI_i + 0.1 \times nmw_i$
    $cum\_fit = cum\_fit + fit_i$
  end for
  $r_1 \leftarrow$ Random number from $\left[\frac{cum\_fit}{2}, cum\_fit\right]$
  while $(sum\_fit < r_1) \wedge (|Sample\_Pop| \leq \frac{2N}{3})$ **do**
    $r_2 \leftarrow$ Random particle picked from Pop without replacement
    $sum\_fit = sum\_fit + fit_{r_2}$
    $Sample\_Pop = Sample\_Pop \cup r_2$
  end while
  $l = \frac{2N}{3} - |Sample\_Pop|$
  if $l > 0$ **then**
    $Rem\_pop = Pop - Sample\_Pop$

$N_{rem} = |Rem\_pop|$

$\mu\_pop = \frac{cum\_fit}{N}$

for $i = 1$ to N **do**

  $\sigma\_pop = \sigma\_pop + (fit_i - \mu\_pop)^2$

end for

$\sigma\_pop = \sqrt{\frac{\sigma_{pop}}{N}}$

for $i = 1$ to N_{rem} **do**

  $fit_i = 0.5 \times (1 - PPV_i) + 0.25 \times AAC\_Y_i + 0.15 \times npI_i + 0.1 \times nmw_i$

  if $fit_i < (\mu\_pop - \sigma\_pop)$ **then**

    $pop1 = pop1 \cup Rem\_pop_i$

  else if $(\mu\_pop - \sigma\_pop) \leq fit_i \leq (\mu\_pop + \sigma\_pop)$ **then**

    $pop2 = pop2 \cup Rem\_pop_i$

  else

    $pop3 = pop3 \cup Rem\_pop_i$

  end if

end for

if $|Sample\_Pop| < \frac{N}{3}$ **then**

  if $l \geq |pop1|$ **then**

    $Sample\_Pop = Sample\_Pop \cup pop1$

    $l = l - |pop1|$

    if $l \geq |pop2|$ **then**

      $Sample\_Pop = Sample\_Pop \cup pop2$

      $l = l - |pop2|$

    else

      $Sample\_Pop = \textsc{Random\_Select}(Sample\_Pop, pop2)$

    end if

    while $(l > 0)$ **do**

52: $Sample\_Pop = \text{RANDOM\_SELECT}(Sample\_Pop, pop3)$
53: **end while**
54: **else**
55: $Sample\_Pop = \text{RANDOM\_SELECT}(Sample\_Pop, pop1))$
56: **end if**
57: **else**
58: **if** $l \geq |pop3|$ **then**
59: $Sample\_Pop = Sample\_Pop \cup pop3$
60: $l = l - |pop1|$
61: **if** $l \geq |pop2|$ **then**
62: $Sample\_Pop = Sample\_Pop \cup pop2$
63: $l = l - |pop2|$
64: **else**
65: $Sample\_Pop = \text{RANDOM\_SELECT}(Sample\_Pop, pop2)$
66: **end if**
67: **while** $(l > 0)$ **do**
68: $Sample\_Pop = \text{RANDOM\_SELECT}(Sample\_Pop, pop3)$
69: **end while**
70: **else**
71: $Sample\_Pop = \text{RANDOM\_SELECT}(Sample\_Pop, pop3)$
72: **end if**
73: **end if**
74: **end if**
75: **return** *Sample_Pop*
76: **end procedure**

Algorithm 6 Random selection

Input: $Sample\_Pop, popx, l$

Output: $Sample\_Pop$

procedure RANDOM_SELECT($Sample\_Pop, popx, l$)
  while $(l > 0)$ **do**
    $r_3 \leftarrow$ Random particle picked from $popx$ without replacement
    $Sample\_Pop = Sample\_Pop \cup r_3$
    $popx = popx - r_3$
    $l = l - 1$
  end while
return $Sample\_Pop$
end procedure

Algorithm 7 Single-objective based fronts

Input: Pop, N

Output: $fronts$

procedure SORT(Pop, N)
  $c \leftarrow 1$
  $Sorted\_Pop = \phi$
  $fronts = \phi$
  $front_1 = \phi$
  for $i = 2$ to N **do**
    $temp = X_i$
    $Sorted\_Pop_c = X_i$
    $c = c + 1$
    $j = i - 1$
    while $i > 0 \wedge X_j > temp$ **do**

12: $X_{j+1} = X_j$

13: $j = j - 1$

14: **end while**

15: $X_{j+1} = temp$

16: **end for**

17: $j = 1$

18: $front_j = front_j \cup Sorted\_Pop_1$

19: $front\_fit = round(fit_{Sorted\_Pop_1}, 2)$

20: **for** $i = 2$ to N **do**

21: **if** $round(fit_{Sorted\_Pop_i}, 2) = front\_fit$ **then**

22: $front_j = front_j \cup Sorted\_Pop_i$

23: **else**

24: $fronts = fronts \cup front_j$

25: $j = j + 1$

26: $front\_fit = fit_{Sorted\_Pop_i}$

27: **end if**

28: **end for**

29: **return** $fronts$

30: **end procedure**

$$\begin{aligned} \textbf{pop1} &= \{X_i : fit_i < \mu\_pop - \sigma\_pop\} \\ \textbf{pop2} &= \{X_i : \mu\_pop - \sigma\_pop \leq fit_i \leq \mu\_pop + \sigma\_pop\} \\ \textbf{pop3} &= \{X_i : fit_i > \mu\_pop + \sigma\_pop\} \end{aligned} \tag{6.20}$$

The Algorithm 5 ensures that the *Sample_Pop* contains two-thirds of the particles present in *Pop* by performing a few steps. Some more particles are added to the *Sample_Pop* from the remaining particles *Rem_pop* (particles in *Pop* minus the ones in *Sample_pop*). For this, the mean (μ_pop) and standard deviation (σ_pop) of *Pop* are calculated as per lines 21-27 (Algorithm 5). Then, based on the fitness value, the

Rem_pop is divided into three groups (as per Eq. 6.20, and lines 28-35 of Algorithm 5).

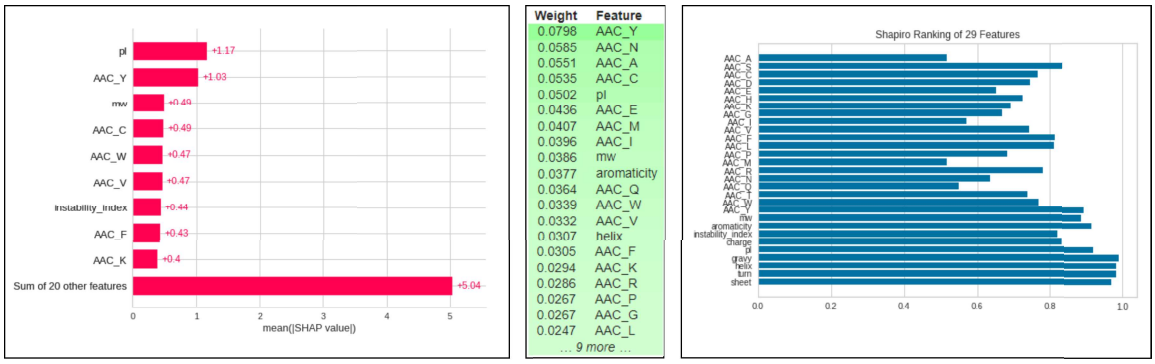
If the size of the *Sample_Pop* is less than one-third of *Pop*, it is inferred that mostly the particles with high fitness value are selected in the sample. This means there is a need to add some particles with low fitness values to facilitate convergence. In this case, there are two ways of selecting more particles, which are enumerated as follows (lines 36-48).

1. If the size of *pop*1 is less than the difference between $\frac{2N}{3}$ and $|Sample\_Pop|$, then all the particles from *pop*1 are added to *Sample_pop*. Then, the particles from *pop*2 are added randomly to the $|Sample\_Pop|$(lines 37-46).
2. If (1) is false, then particles are added randomly from *pop*1 to the *Sample_Pop* until the size of the latter becomes $\frac{2N}{3}$ (line 48).

Conversely, if the $|Sample\_Pop|$ is more than $\frac{N}{3}$, the sample might be devoid of particles with high fitness. Hence, the following two steps are executed (lines 50-61, Algorithm 5).

1. If the size of *pop*3 is less than the difference between $\frac{2N}{3}$ and $|Sample\_Pop|$, then all the particles from *pop*1 are added to *Sample_pop*. Then, the particles from *pop*2 are added randomly to the $|Sample\_Pop|$ (lines 50-59).
2. If (1) is false, then particles are added randomly from *pop*3 to the *Sample_Pop* until the size of the latter becomes $\frac{2N}{3}$ (line 61).

The algorithm becomes more stochastic due to the random selection of the particles. Hence, it facilitates the search for better solutions. Next, the PPV, nmw, npI, and AAC_Y values are calculated (lines 9–13, Algorithm 4) after converting the particle to its corresponding peptide (pep_i) using the utility function $ConvertToPep$. Then, the fitness and mass of X_i are determined in lines 14–15. The contribution of each AA is determined using the DL-B3P2pred model (line 16). The distance and force of each particle X_j on a specific particle X_i in every dimension, $d = \{1, \cdots, D\}$ is calculated in lines 17-21, which is then summed up to calculate the net force exerted on that particle



(a) (b) (c)

Figure 6.4: The important features and their weightage as per (a) SHAP, (b) ELI5, and (c) Yellowbrick (Shapiro-Wilk algorithm)

(line 22). The acceleration of X_i is calculated using its mass and force values (line 23), and the particle's velocity and position are updated using Eq. 6.8. Note that the velocity and position of a particle are updated only if the AA at that position is not positively influencing its potential to cross the BBB.

Once the algorithm has converged, the $SORT$ function (Algorithm 7) is called. The HyGSA algorithm sorts the final population using the fitness values (rounded to two decimal places), as described in lines 6–14 of Algorithm 7. Then, the potential solutions are categorized into distinct non-dominated fronts (lines 15-24) and returned to the function $HYBRIDIZED-GSA$. As indicated in lines 29-35 of Algorithm 4, the set $B3P2s$ is generated as output, containing the top Num peptides.

6.4 Experiments, results, and discussions

The HyGSA framework was coded in Python with the help of DL libraries like Keras, PyTorch, scikit-learn, and HuggingFace's transformers [127]. The DL-B3P2pred was trained on NVIDIA Tesla T4 GPU cores (12 GB RAM), and the ML-B3P2pred model was trained on a compute node having Intel(R) Xeon processors (51 GB RAM).

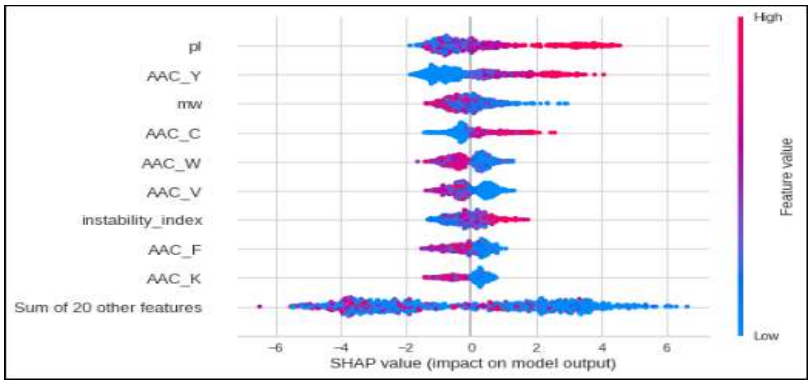


Figure 6.5: Relationship between feature and SHAP values. A single dot represents one data point.

6.4.1 Experiments conducted in phase-1

In phase 1, the ML-B3P2pred was used to find the critical features with respect to the classification of B3P2s. Following this, the DL-B3P2pred model was built to find the AAs that contribute positively to the task of classification. The details of the performed experiments are as follows.

6.4.1.1 Finding the features of importance using the ML-B3P2pred model

On the test set, the ML-B3P2pred showed an accuracy, f1-score, and area under the ROC curve (AUC) of 84%, 84%, and 91%, respectively. The explainability tools such as SHAP, Yellowbrick, and ELI5 aided in ranking the features, as illustrated in Figure 6.4. As per SHAP, the critical features are pI, AAC_Y, mw, AAC_C, AAC_W, AAC_V, instability index, AAC_F, and AAC_K. As per ELI5, AAC_Y, AAC_N, AAC_A, AAC_C, pI, AAC_E, AAC_M, AAC_I, and mw were important. The Yellowbrick tool considered pI, GRAVY index, alpha helix propensity, beta-turn propensity, beta-sheet propensity, AAC_Y, mw, aromaticity, and AAC_S important. Conclusively, mw, AAC_Y, and pI were selected as the objectives of the fitness function because all three explainability frameworks chose them as critical. The correlation of the critical features with the output can be illustrated using Figures 6.5 and 6.6. As shown, high values of AAC_Y

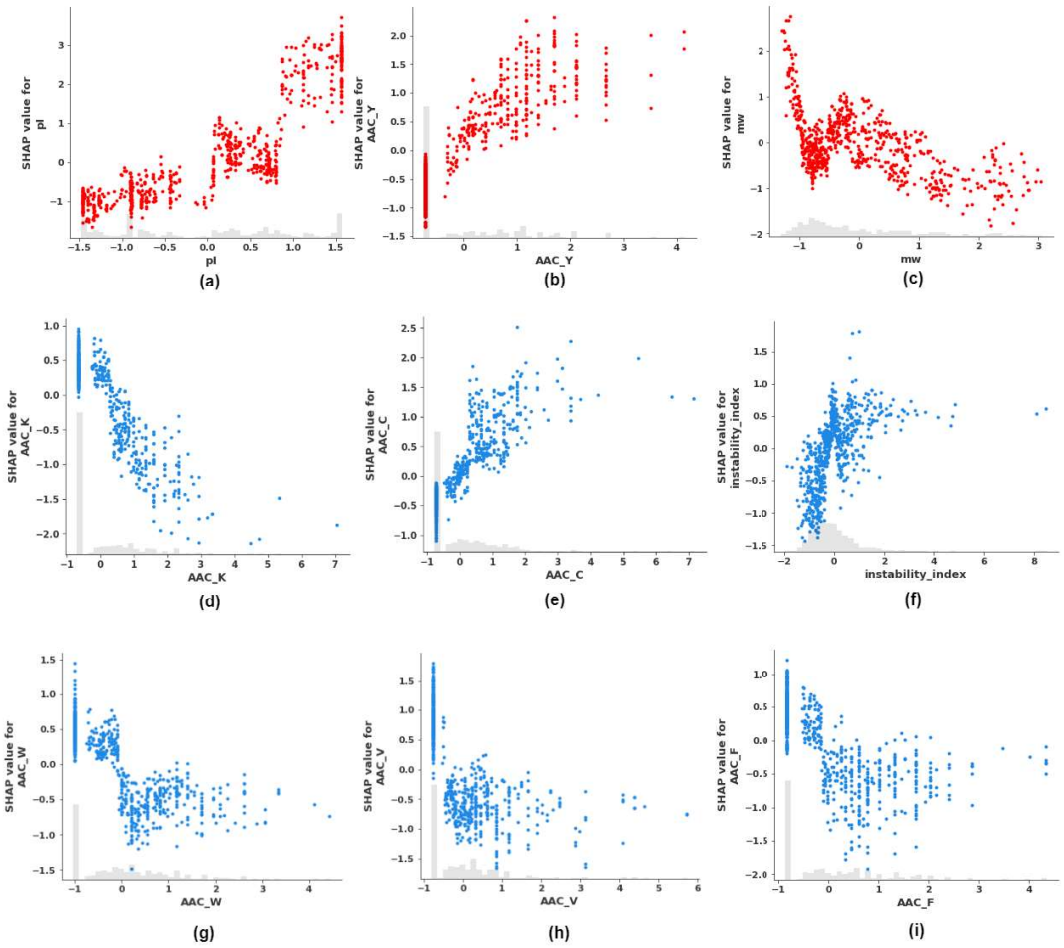


Figure 6.6: Relationship between the feature and the Shapley values (a) pI, (b) AAC_Y, (c) normalized mw, (d) AAC_K, (e) AAC_C, (f) instability index, (g)AAC_W, (h) AAC_V, (i) AAC_F

and pI and low values of mw resulted in high SHAP values.

6.4.1.2 Finding the important AA residues in the B3P2s using the DL-B3P2pred model

The DL-B3P2pred model was found to have an accuracy, f1-score, and AUC of 89%, 91%, and 95%, respectively. Then, the CAPTUM framework was employed to explain the model and determine the contribution of constituent AAs of a given peptide in its classification. Then, this model was integrated into the HyGSA framework to aid in updating the position and velocity of a particle.

Table 6.1: Comparison of DL-B3P2pred with existing state-of-the-art (SOTA) models on benchmark test sets.

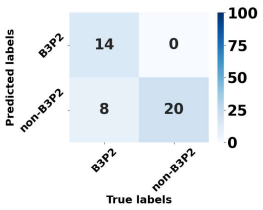
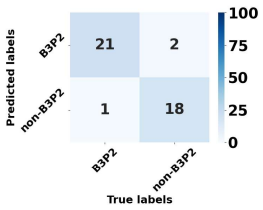
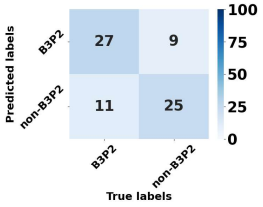
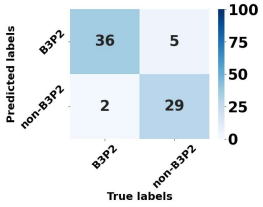
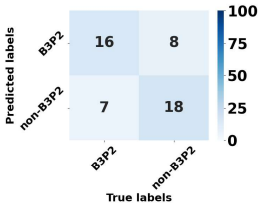
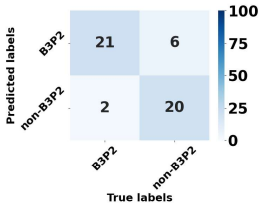
Model	Accuracy	F1-score	AUC	Confusion matrix (SOTA model)	Confusion matrix (DL-B3P2pred)
B3Pred	80.95	77.78	93.18		
DL-B3P2pred	92.86	93.33	97.50		
BBPpred	72.22	72.97	80.88		
DL-B3P2pred	90.28	91.14	95.82		
BBPpredict	69.39	68.09	78.18		
DL-B3P2pred	83.67	84	84.45		

Table 6.2: Parameters used in HyGSA

Parameter	Value
Initial gravitational constant (G_0)	100
No. of particles (N)	100
No. of iterations (k)	30
Rate of decay of G (γ)	20

6.4.1.3 Comparison of the DL-B3P2pred model with existing classifiers

There has been limited research in this area, and these studies focused on the binary classification of B3P2s/non-B3P2s only. The DL-B3P2pred model was compared with B3Pred, BBPpred, and BBPpredict as given in Table 6.1 (note that the peptides which were in the training sets of these models, and also present in the test set considered in this work, have been removed). The proposed DL-B3P2pred outperforms all the state-of-the-art classifiers in the case of all the performance metrics.

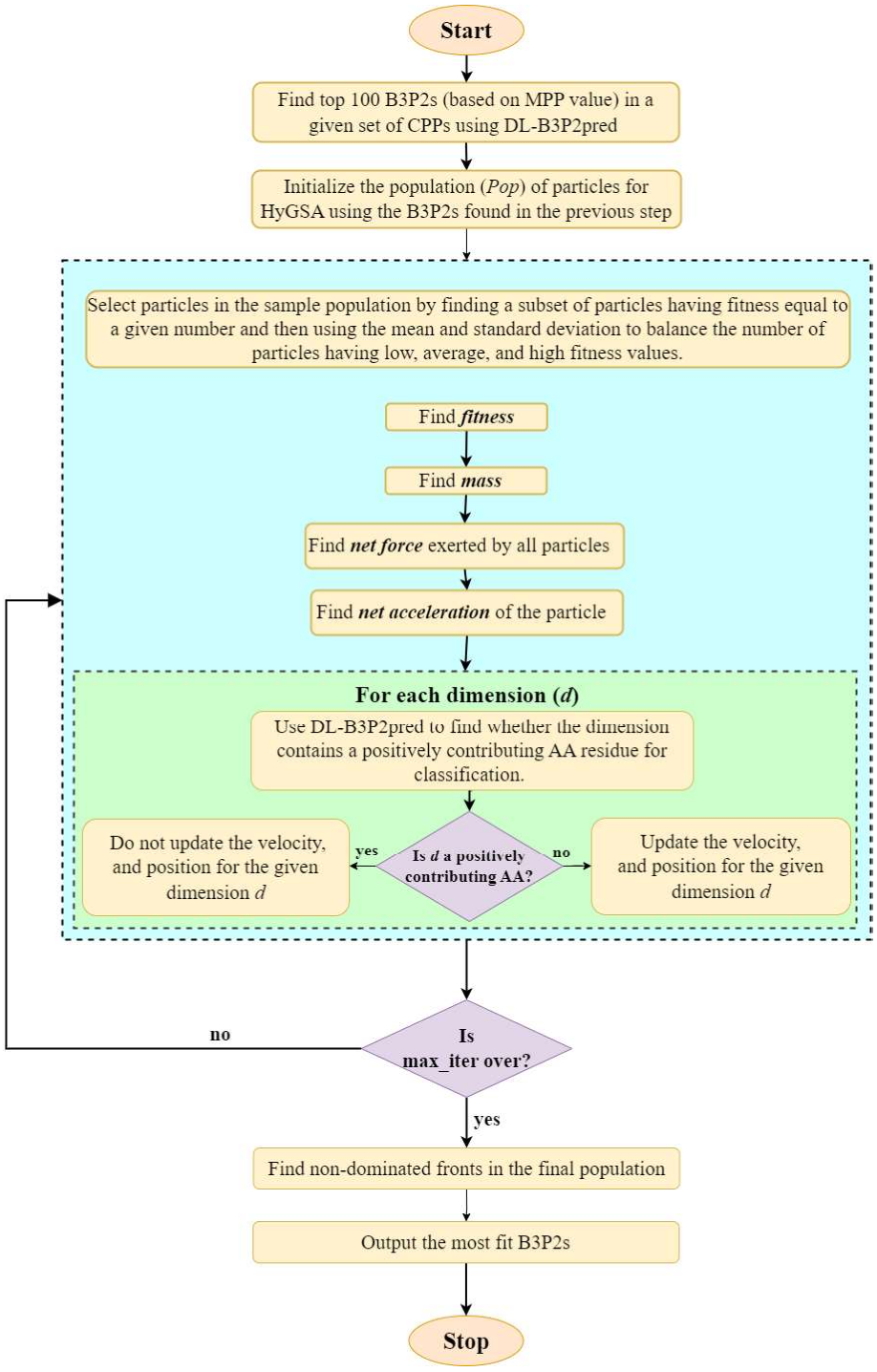


Figure 6.7: The flowchart depicting the working of HyGSA on a dataset comprising cell-penetrating peptides (CPPs)

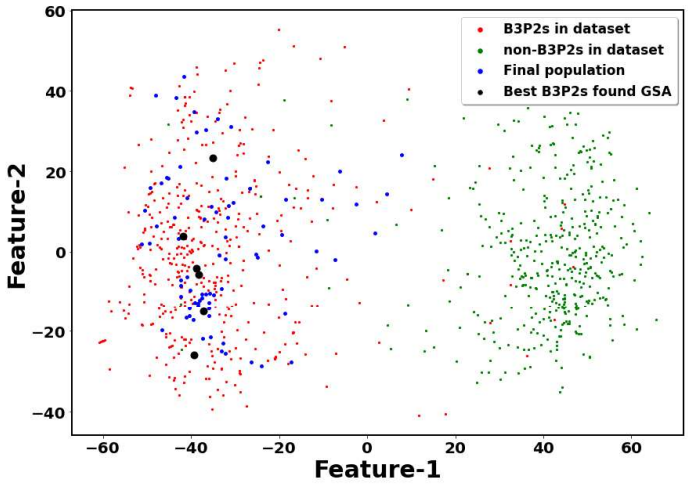


Figure 6.8: Isometric mapping of the B3P2s discovered by HyGSA, along with other CPPs and the peptides present in our dataset

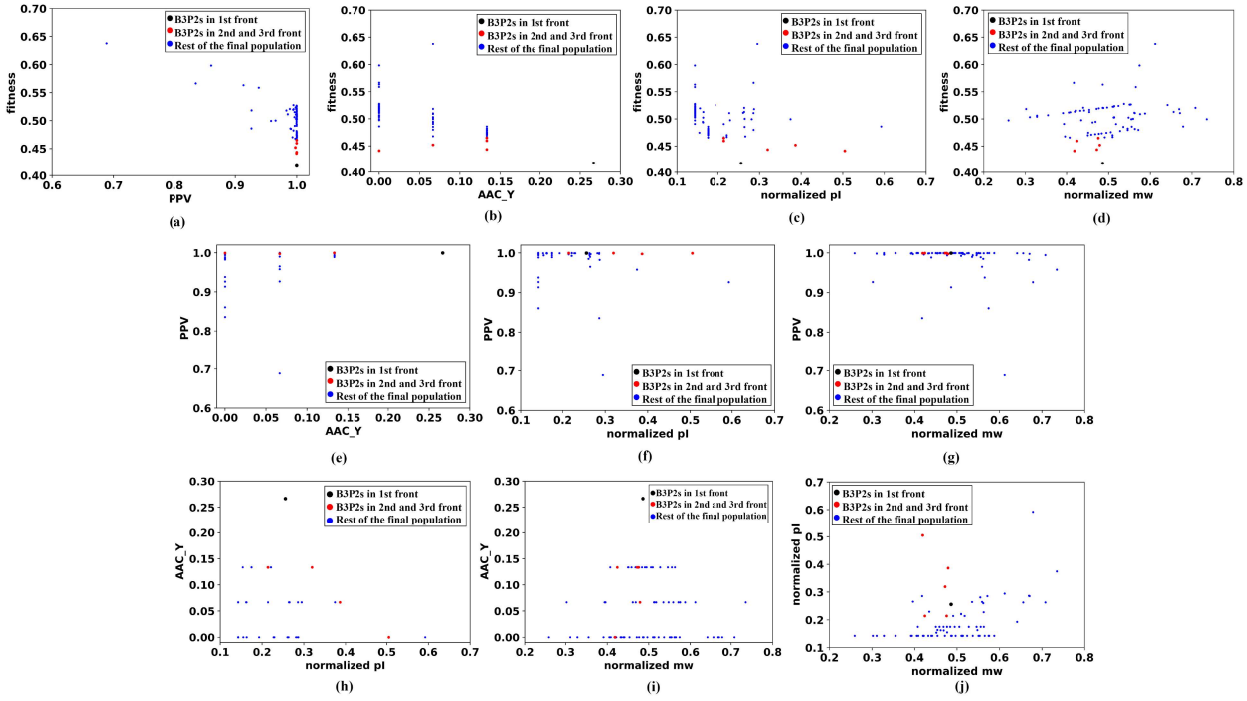


Figure 6.9: (a)-(d) show the scatter plot of objective values and the fitness values of all the particles. (e)-(j) show the pairwise relation between the values of objectives obtained by different particles

6.4.2 Using the HyGSA to find novel B3P2s

The experiments performed in phase-2 have been pictorially represented by Figure 6.7. A set of experimentally validated and annotated cell-penetrating peptides (CPPs) of length 15 were collected. After that, the DL-B3P2pred model was used to find the peptides with the best chance to penetrate the BBB (with $PPV > 0.9$), and they were used to initialize the population of the HyGSA. Then, HyGSA was executed, wherein each particle's position and velocity were updated iteratively. The parameters of HyGSA are given in Table 6.2. Note that, for every dimension d, the velocity and position for a given particle were updated only when the AA residue at position d in the corresponding peptide was found to have a negative contribution in the B3P2 classification, as per the explainability framework. After the algorithm iterated a maximum number of times (k_0), peptides from the first three fronts were given as output (considering $Num = 5$). Then, the isometric mapping (Isomap) technique was employed to map the distribution of the discovered B3P2s against the distribution of the peptides (B3P2/ non-B3P2) in the collected dataset. As shown in Figure 6.8, the discovered B3P2s were found well-dispersed within the core region of experimentally validated B3P2s. Figure 6.9 shows the scatter plot in which each point represents a population particle. The trade-off achieved between each objective and the fitness value has been illustrated using Figures 6.9 (a)-(d), and that between each pair of objectives is shown in Figures 6.9 (e)-(j). Based on this plot, certain inferences can be drawn, which are as follows.

1. **PPV**: The peptides in the first three fronts have the maximum PPV.
2. **AAC_Y**: The peptides in the first three fronts (except one) have higher AAC_Y than the mean.
3. **Normalized pI**: The peptides in the first three fronts have higher npI than the mean.
4. **Normalized mw**: The peptides in the first three fronts have lower nmw than the mean.

Thus, it can be conclusively said that the B3P2s in the first three fronts have better and much more desirable characteristics than the rest of the population.

6.5 Conclusion

This chapter proposes a novel population-based multi-objective optimization technique called hybridized gravitational search algorithm (HyGSA) for designing optimal blood-brain barrier-penetrating peptides with desirable characteristics such as high isoelectric point, low molecular weight, etc. This was achieved with the help of explainable AI-based tools and techniques such as SHAP, CAPTUM, etc., which were used to build explainable deep (DL-B3P2pred) and machine learning (ML-B3P2pred) models. The XAI-based ML-B3P2pred model found the critical features involved in B3P2 classification for formulating the fitness function of HyGSA. The explainable DL-B3P2pred model was used to figure out the positively contributing residues to help HyGSA while modifying the particles (peptides). Moreover, a novel method was proposed to sample the population of HyGSA to decrease the computational cost involved in searching for the optimal B3P2s. Lastly, several non-dominated fronts based on the fitness value were generated as the final output. In order to help the scientific community find and optimize the B3P2s in random proteins, a freely accessible web app has been deployed online.

In the future, reinforcement learning can help design novel B3P2s with desirable features [128] or for generating pseudo-random sequences to initialize the population of HyGSA [129]. Moreover, the current dataset of B3P2s is very scarce. So, generating synthetic data points could help build more robust, explainable models, which can add to HyGSA's performance. Some popular generative algorithms, such as generative adversarial networks, can be used for this purpose.