M.Sc. Engg. (CSE) Thesis

# REVISITING SUCCINYLATED LYSINE RESIDUE PREDICTION WITH CAREFULLY SELECTED PHYSICOCHEMICAL AND BIOCHEMICAL PROPERTIES OF AMINO ACIDS

Submitted by

Shehab Sarar Ahmed

0419052005

Supervised by

Dr. M. Sohel Rahman

Submitted to

**Department of Computer Science and Engineering**

**Bangladesh University of Engineering and Technology**

Dhaka, Bangladesh

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

May 2022

## Candidate's Declaration

I, do, hereby, certify that the work presented in this thesis, titled, "REVISITING SUCCINYLATED LYSINE RESIDUE PREDICTION WITH CAREFULLY SELECTED PHYSICOCHEMICAL AND BIOCHEMICAL PROPERTIES OF AMINO ACIDS", is the outcome of the investigation and research carried out by me under the supervision of Dr. M. Sohel Rahman, Professor, Department of CSE, BUET.

I also declare that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Shehab
22.05.22

Shehab Sarar Ahmed

0419052005

The thesis titled "**REVISITING SUCCINYLATED LYSINE RESIDUE PREDICTION WITH CAREFULLY SELECTED PHYSICOCHEMICAL AND BIOCHEMICAL PROPERTIES OF AMINO ACIDS**", submitted by Shehab Sarar Ahmed, Student ID 0419052005, Session April 2019, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents on May 16, 2022.

# Acknowledgement

Firstly, I want to convey my gratitude to the Almighty for keeping me healthy during the pandemic era and for enabling me to conduct this research and present it for the completion of my M.Sc. Engineering thesis.

I am extremely thankful to my supervisor Dr. M. Sohel Rahman for his relentless effort and constant guidance during this research. I had to conduct numerous computationally intensive experiments for a long period of time which would have been impossible without my supervisor providing me his high end PC. I am thankful to him for his boundless patience in times of my countless blunders, assistance in my work, outstanding advices and continuous supervision. Specifically, his motivation during my difficult times was irreplaceable.

I thank our honorable head of the department Dr. Mahmuda Naznin as well as the other members of my thesis committee Dr. Abu Sayed Md. Latiful Hoque, Dr. Md. Shamsuzzoha Bayzid, and especially the external member Dr. Ahsanur Rahman.

I also express my gratitude to Muhammad Ali Nayeem who has recently pursued his PhD degree at BUET, for his valuable advice and continuous mental support throughout my thesis work.

I would also like to express my utmost gratitude to the Department of Computer Science and Engineering (CSE) for providing me with the laboratory facilities during the thesis work. I used a PC with GPU of gaming lab for a long period solely for my thesis purpose. I sincerely thank those CSE Office staffs who have provided logistic support to me to successfully complete the thesis work.

In the end, I want to thank my family, friends, and all those who supported me with their advice, and suggestion during the course of my thesis.

Dhaka
May 16, 2022

Shehab Sarar Ahmed
0419052005

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# Abstract

Succinylation of lysine residue is a special type of post-translational modification (PTM). It has a crucial role in balancing the processes of cells. Abnormal succinylation can be the cause of cancers, metabolism diseases, inflammation and nervous system diseases. Detecting succinylation sites is of great importance to explore the function of proteins. However, the experimental methods to detect succinylation sites are costly, time and labor consuming. This thus calls for computational models with high efficacy and attention has been given in the literature for developing such models, albeit with only moderate success in the context of different evaluation metrics. In particular, the existing works failed to balance the two metrics, sensitivity and specificity, leaving a large room for improvements in this context. One important aspect in this context is the biochemical and physicochemical properties of amino acids, which appear to be useful as features for such computational predictors. However, some of the existing computational models did not use the biochemical and physicochemical properties of amino acids, while some others used them without considering the inter-dependency among the properties.

In this thesis, we revisit the computational prediction of succinylated lysine residue (SLR) and use a broad spectrum of weaponry to tackle this problem. We first focus on the biochemical and physicochemical properties of amino acids and formulate an optimization problem to find combination that is more suitable for the problem at hand considering their inter-dependencies and other factors. In particular, we propose a variant of genetic algorithm, called IBCGA, to search for suitable combinations thereof for efficient prediction of SLRs. In this context, we leverage the power of Random Forest (RF) and Balanced RF (a variant of RF to handle imbalanced data).

We then propose three deep learning architectures, CNN+Bi-LSTM (CBL), Bi-LSTM+CNN (BLC) and their combination (CBL_BLC) thereby leveraging the potential of deep neural network architectures for SLR prediction. We also employ different ensembling techniques to improve upon the performance of our models, which includes heterogeneous ensembling of traditional ML models with deep learning architectures as well. Finally, we apply differential evolution to tune the threshold of ensemble classifiers thereby providing the biologists and practitioners with a knob to balance the sensitivity and specificity.

The combinations of biochemical and physicochemical properties derived through our optimization process achieve better results than the results achieved by the combination of all the properties. In this context, one of the best performing combinations consists of only two properties. As for our deep learning architectures,

we find that CBL_BLC is outperforming the other two. Ensembling of different models successfully improves the results. For example, ensembling of multiple CBL_BLC models improves the sensitivity of CBL_BLC from 0.73 to 0.744, specificity from 0.684 to 0.691, accuracy from 0.707 to 0.718 and Matthews correlation coefficient from 0.415 to 0.436. Ensembling of the traditional ML models (BP-E) achieves a very competitive sensitivity score (even in comparison with the deep learning models). For instance, BP-E achieves a sensitivity of 0.745 on the first dataset while the best achieved sensitivity is 0.772. The sensitivity achieved by BP-E on the second dataset is 0.721 while the best achieved sensitivity is 0.724. Notably, tuning the threshold of the ensemble classifiers further improves the results. For example, tuning the threshold of the ensemble of all deep learning and traditional ML models improves the sensitivity from 0.772 to 0.795. Consequently, the specificity decreases from 0.681 to 0.666. Upon comparing our work with other existing works on two datasets, we find that our model is far better with respect to sensitivity. Our tuned models achieve the best sensitivity of 0.805 and 0.811 on the two datasets compared to the second best of 0.725 and 0.7874, respectively. However, the specificity is comparatively low compared to the existing works. We show that by altering the threshold of our proposed ensemble classifier, we are able to control the sensitivity and specificity thereby achieving better results from the state-of-the-arts with respect to both sensitivity and specificity.

For reproducibility and extensibility of our work, we make the source code publicly available at https://github.com/Dariwala/Succinylation-with-biophysico-and-deep-learning.

# Chapter 1

# Introduction

Proteins are an integral part of any organism. They conduct numerous complicated tasks of the body, for example, stimuli response, transportation of molecules, and forming the structure of cells. Proper function of the body's tissues can not be imagined without the existence of proteins.

Amino acids are the monomers that constitute proteins. Twenty types of amino acids are the most common in proteins. A polypeptide is a long chain of amino acids. A protein is made up of one or more polypeptides. Short polypeptides contain 20-30 amino acids. Amino acids are coupled together through peptide bond. Formation of a polypeptide through the long chain of amino acids is shown in Figure 1.1.

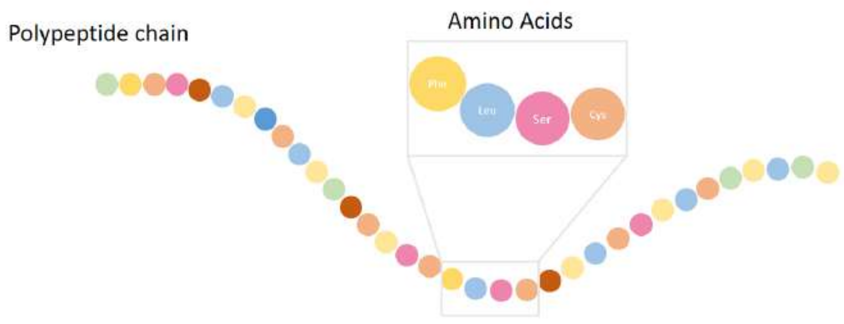


Figure 1.1: Chain of amino acids bonded together to form polypeptide.

A protein is formed from a gene in two steps. During this process, information flows from DNA $\rightarrow$ RNA $\rightarrow$ protein. This phenomenon is known as the *central dogma of molecular biology*.

The first step in this process is called **transcription**, during which the DNA is transcribed, into RNA. In eukaryotes, the RNA has to go through some processing steps to become a mature messenger RNA (mRNA).

The second step in the process is **translation**, during which the mRNA is decoded to form a sequence of amino acids (i.e., protein). The steps of the formation of protein from DNA are shown in Figure 1.2.

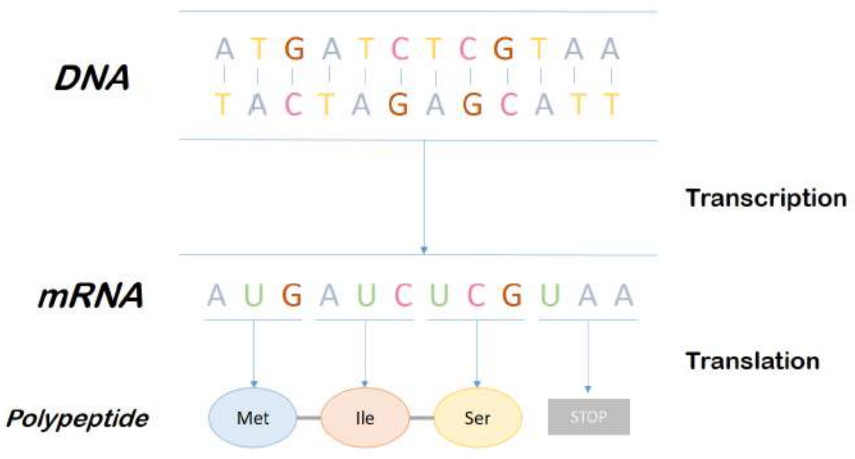


Figure 1.2: DNA is converted to RNA through transcription. RNA is converted to protein through translation.

An important process related to protein synthesis is the Post-translational modification (PTM). PTMs refer to the enzymatic alteration of proteins. They can expand the chemical range of the 20 standard amino acids by changing an already existing functional group or establishing a new one such as phosphate. PTMs play a crucial role in numerous biological processes through influencing the formation and dynamics of proteins [3, 4].

Phosphorylation is one of the most popular PTMs for synchronizing the activity of enzymes and is the most common post-translational modification [5]. It refers to the addition of a phosphoryl group ($PO_3^-$) to serine. Protein phosphorylation is specifically significant because of their roles; for example, this alteration excites (or depresses) almost 50% of the enzymes of Saccharomyces cerevisiae, hence balancing their function [6–8].

Notably, the aberrant phosphorylation of proteins has been found to result in major diseases like diabetes, cancer and rheumatoid arthritis [9]. Mutations in kinases and phosphatases result in disorders. Several naturally occurring toxins and pathogens employ their effects by changing the phosphorylation states of intracellular proteins [9].

Acetylation, another common PTM, introduces an acetyl ($-C(O)CH_3$) functional group into amino acid residue. Acetylation is crucial because of several of its important reactions in the body, including but not limited to Protein formation and Drug biotransformation. Dysregulation of acetylation or deacetylation leads to diverse disorders as epithelial cancers, leukemia, Rubinstein-Taybi syndrome and fragile X syndrome [10].

Lysine succinylation (addition of $OCCH_2CH_2CO$ or $HOOCCH_2CH_2CO$ to the lysine residue) is an important PTM in regulating the cellular processes [11]. It can cause alteration of charge in the surroundings and stimulate structural and functional adjustments to substrate proteins [12]. Succinylation alters charge of lysine residue from +1 to -1 (at physiological pH) and introduces a comparatively larger structural component (100 Da), bigger than acetylation (42 Da). Hence, Succinylation is expected to result in substantial changes in protein function and structure. Succinylation is involved in numerous key energy catabolism routes, including tricarboxylic acid cycle, amino acid decay and fatty acid metabolic process, pentose phosphate route, glycolysis, TCA cycle, and pyruvate metabolic process [13–17]. Hence, detecting and investigating the succinylated lysine residues is crucial to realize the function of proteins.

Moreover, succinylation may result in phthisis, inflammation [18]. The aberration of succinylations has also been found to cause diseases like metabolism diseases [14, 19], nervous system diseases [20], and cancers [21]. Therefore, it is crucial to identify succinylated sites in the field of physiology too.

Several experimental techniques exist for detecting PTM sites. Although these experimental techniques (e.g., mass spectrometry, liquid chromatography) can detect PTMs pretty accurately, they are usually costly, time and labor-consuming. This thus calls for in silico computational techniques to provide predictions of SLRs with high efficacy.

Several (traditional) machine learning based predictors (e.g., iSuc-PseAAC [22], iSuc-PseOpt [23], pSuc-Lys [24], SuccineSite [11], SuccineSite2.0 [2], GPSuc [25] and PSuccE [26]) have been proposed in the literature, albeit with unsatisfactory (and/or conflicting) results considering different metrics. An important research gap in this context relates to the biochemical and physicochemical (referred to as biophysico henceforth for brevity) properties of amino acids. Some common biophysico properties of amino acids are solubility, amphoteric property, isomerism etc. These properties can be used to numerically represent the amino acids, hence can be used as features in machine learning algorithms. The existing works did not take into consideration the relationship among different biophysico properties while choosing a subset of properties to be used for classification. More details about these properties and the associated research gap will be discussed in Section 4.1.

Very recently, the power of deep learning has been leveraged in DeepSucc [1], DeepSuccinylSite [27]. Although DeepSucc claims a very good specificity (percentage of negative samples correctly classified) and a reasonable sensitivity (percentage of positive samples correctly classified), the codebase shared by the authors contain errors and discrepancies. DeepSuccinylSite offers a reasonable sensitivity albeit at the cost of low specificity. Therefore, the quest for incorporating biophysico properties with a computational model to build a better predictor is still on.

## 1.1 Objective of this thesis

The overarching aim of the thesis is to develop a computational predictive model for the detection of succinylated lysine residue (SLR in short), i.e., to predict whether a lysine residue in a protein sequence represents a succinylation site or not.

The objectives of this thesis are as follows-

- To formulate an optimization problem of finding a suitable subset of biophysico properties from the AAIndex [28] for better prediction of the task at hand.
- To assess the statistical significance of each of the features to be used for the prediction of SLR.
- To leverage the power of deep learning to extract unknown features from the surrounding amino acids of the concerned amino acid.
- To build a robust predictor by incorporating the derived biophysico features as well as temporal relationships of the surrounding amino acids through applying both traditional and deep learning approaches.
- To make all the codes open-source for further research.

## 1.2 Our contributions

This thesis addresses some notable lackings in the field of SLR prediction and thus makes the following contributions.

**Suitable combinations of biophysico properties** We formulate an optimization algorithm to search for suitable combinations of biophysico properties for efficient prediction of SLR.

**Sensitivity-Specificity tradeoff** We provide the researchers with a knob to control the sensitivity and specificity of our proposed model thereby obtaining contextual results.

**Open Source flexible software framework** We developed an open source software framework for prediction of SLR using our proposed approaches. Its components can potentially be modified, replaced or further refined by bioinformatics researchers and practitioners.

## 1.3 Organization

The rest of the thesis is organized as follows. In Chapter 2, we discuss the basic terms related to our study. Chapter 3 discusses the state-of-the-works related to the detection of succinylated sites. In Chapter 4, we present the methodology for selection of suitable subset of biophysico properties. Application of deep-learning and incorporating biophysico properties therewith is discussed in Chapter 5. Finally, Chapter 6 concludes the thesis.

# Chapter 2

# Background

Previous chapter has illustrated the importance of the thesis by discussing the challenges of using experimental methods for detection of succinylation and mentioning briefly the limitations of the previous works. In this chapter, we describe in detail the concepts and leveraged techniques used in this thesis.

Since a large volume of background information hae been presented in this chapter, we present a roadmap of what is coming next. We start with a detailed discussion of the metaheuristic algorithms used in this thesis. We use genetic algorithm to search for suitable combinations of biophysico properties for efficient prediction of SLRs. We adopt differential evolution algorithm to find the optimal thresholds for improving performance of our proposed models. Subsequently, we discuss an experimental design technique named "Orthogonal Experiment Design". This technique uses the concept of *Orthogonal array* to create a representative set of experiments from the set of all possible experiments. We leverage the idea of *Orthogonal array* to perform *Orthogonal Array Crossover* as the crossover operation of the genetic algorithm. Next, we discuss the statistical measures taken to analyze the collected features. We discuss about the correlation coefficient in general and specifically the Spearman correlation coefficient, which we use to assess the statistical significance of the biophysico properties being used for succinylation site prediction. We also discuss a technique called "Permutation feature importance" to inspect the importance of each feature being selected for prediction. The next section elaborately describes the concepts related to machine learning used in this thesis. Finally, we mention the performance metrics that have been used to evaluate our models and perform comparative analysis with existing succinylation site predictors.

## 2.1 Meta-heuristics

Metaheuristics refers to context-independent and high-level algorithmic framework that provides a number of methods to construct heuristic algorithms [29]. Optimization methods formulated in accordance with the techniques provided in a metaheuristic framework, are generally heuristic. While the exact methods are guaranteed to find the optimal solution in a finite (although possibly very long) amount of time, metaheuristics can provide close to optimal solution within an affortable period of time. Hence, they won't break down in case of combinatorial explosion.

Specifically in case of complex or large problem instances, metaheuristics can balance between the goodness of the solution and cost of computation. In addition to that, metaheuristics offer more flexibility than exact methods do in two important ways. First, since these algorithms are designed as problem-independent, they can adapt to meet the requirement of many practical problems in order to obtain solutions of expected quality with an affordable computational cost. Second, different problems have different constraints and formulations. As metaheuristic algorithms are designed from a high level perspective, they can be applied on various optimization problems which vary in their formulation and constraints.

Key examples of metaheuristic algorithms include evolutionary algorithms, simulated annealing, tabu search, large neighborhood search and many more.

### 2.1.1 Genetic algorithm

Genetic algorithm is similar to the system of natural selection where the individuals that are strongest are selected for producing the population of the next generation.

There are five stages of the genetic algorithm.

**Initialization of population** A population refers to a set of individuals. In the first step, a random set of individuals is generated for the next steps.

**Evaluation of fitness** Each individual is assigned a fitness value by which it can be understood how good a solution that individual is in the context of the problem.

**Selection** In this phase, individuals are selected for reproduction based on the their fitness scores.

**Crossover** Crossover is an operation done on a pair of individuals (i.e., parents) to generate children of next generation. Through this operation, genes of the current generation are passed down to the next generation.

**Mutation** Mutation refers to slight change in some of the genes of some selected individuals of the new generation.

The genetic algorithm is described in Algorithm 1. The three most important steps in the genetic algorithm is the $SelectWithReplacement$, $Crossover\ and\ Mutate$ methods. $SeletWithReplacement$ selects an individual from the population based on some criteria. $Crossover$ does mating of of two parents to produce two children. $Mutate$ does some mutation on the children with a small probability. Different variants of genetic algorithm have emerged by adopting different techniques for these three methods. Below we discuss one of the most popular algorithms for selection operation.

**Algorithm 1** Genetic algorithm

```
1:  $popsize \leftarrow desired\ population\ size$
2:  $P \leftarrow \{\}$
3:  for $popsize$ times do
4:      $P \leftarrow P \cup \{new\ random\ individual\}$
5:  end for
6:  $Best \leftarrow null$
7:  while Best is the ideal solution or time is over do
8:      for each individual $P_i \epsilon P$ do
9:          CalculateFitness($P_i$)
10:         if $Best = null$ or $Fitness(P_i) > Fitness(Best)$ then
11:             $Best \leftarrow P_i$
12:         end if
13:     end for
14:     $Q \leftarrow \{\}$
15:     for $popsize/2\ times$ do
16:         Parent $P_a \leftarrow SelectWithReplacement(P)$
17:         Parent $P_b \leftarrow SelectWithReplacement(P)$
18:         Children $C_a, C_b \leftarrow Crossover(P_a, P_b)$
19:         $Q \leftarrow Q \cup \{Mutate(C_a), Mutate(C_b)\}$
20:     end for
21:     $P \leftarrow Q$
22: end while
23: return Best
```

### Tournament Selection Algorithm

Tournament Selection [30] is an algorithm for selecting individuals from the current population for reproduction (e.g., two individuals are needed to perform crossover operation). The K-way tournament selection algorithm will initially select k individuals and perform a tournament among them and the winner of the tournament is selected. This process is repeated until desired number of individuals are selected. The steps of the tournament selection algorithm are as follows:

1. Randomly choose k individuals from the population and run a tournament with them.

2. Choose the winner of this tournament.

3. Repeat steps 1 and 2 until the desired number of individuals are selected.

### 2.1.2 Differential evolution

Differential evolution (DE) [31,32] is an algorithm that tries to iteratively optimize the solutions with regard to the problem at hand.

DE does not utilize the gradient of the optimization problem. Hence, the problem does not need to be differentiable for applying DE. This is not the case with many other metaheuristic algorithms that use the information of gradients of the problem to optimize the solution.

DE searches for the solution of the optimization problem by keeping a set of individuals and generating new generation of children by combining current individuals according to its own techniques. Next, it keeps the candidate solutions that are the fittest with respect to the optimization problem at hand. In this way, the optimization problem is considered as a black box that simply provides a measure of fitness given an individual solution and the gradient is hence not required.

The steps of DE are described in Algorithm 2.

## 2.2 Orthogonal experiment design

Orthogonal experimental design (OED) with orthogonal array (OA) is an effective way of investigating the effect of a number of factors [33]. The factors work as parameters, which affect the response variables. The factors can take different values known as levels of the factors. A **complete factorial** experiment tries each combination of levels of each of the factors which is often infeasible [34], while a **fractional factorial** experiment tries to select a subgroup of combinations. Orthogonal experimental design leverages fractional factorial experiments to effectively search for the best combination of levels of factor for the purpose of optimization of the response variable. An illustrative example of OED using an objective function is as follows:

$$maximize\ y(x_1, x_2, x_3) = 5x_1 - 2x_2 + x_3$$

where $x_1 \in \{-1, 1\}$, $x_2 \in \{1, 2\}$ and $x_3 \in \{3, 4\}$.

This optimization problem can be thought as a problem of three factors, all of which have two levels. Let, $x_1$, $x_2$, $x_3$ be factors $F_1$, $F_2$, and $F_3$, respectively. Let, level 1 (2) denote the smaller (larger) value of each parameter. The response variable, $y$ is the objective function. The strategy of **complete factorial** experiment is to evaluate all of the $2^3 = 8$ combinations of levels. Hence, we would obtain the best combination $(x_1, x_2, x_3) = (1, 1, 4)$ with $y = 7$. The results of the discussed experiment are shown in Table 2.1.

While a complete factorial experiment inspects all possible combinations, a fractional factorial experiment intelligently chooses a subset of combinations of levels, such as the $2^{nd}$, $4^{th}$, $5^{th}$, and $7^{th}$ combinations. The best among the four combinations is $(x_1, x_2, x_3) = (1, 1, 3)$ with $y = 6$.

**Algorithm 2** Differential Evolution

```
1: $\alpha \leftarrow$ mutation rate
2: $popsize \leftarrow$ desired population size
3: $P \leftarrow \{\}$
4: $Q \leftarrow \{\}$
5: for $popsize$ times do
6:     $P \leftarrow P \cup \{new\ random\ individual\}$
7: end for
8: $Best \leftarrow null$
9: while Best is the ideal solution or time is over do
10:     for each individual $P_i \epsilon P$ do
11:         CalculateFitness($P_i$)
12:         if $Q \neq \{\}$ and $Fitness(Q_i) > Fitness(P_i)$ then
13:             $P_i \leftarrow Q_i$
14:         end if
15:         if $Best = null$ or $Fitness(P_i) > Fitness(Best)$ then
16:             $Best \leftarrow P_i$
17:         end if
18:     end for
19:     $Q \leftarrow P$
20:     for each individual $Q_i \epsilon Q$ do
21:         $\overrightarrow{a} \leftarrow$ a copy of individual other than $Q_i$, chosen at random with replacement from $Q$
22:         $\overrightarrow{b} \leftarrow$ a copy of individual other than $Q_i$ or $\overrightarrow{a}$, chosen at random with replacement from $Q$
23:         $\overrightarrow{c} \leftarrow$ a copy of individual other than $Q_i$ or $\overrightarrow{a}$ or $\overrightarrow{b}$, chosen at random with replacement from $Q$
24:         $\overrightarrow{d} \leftarrow \overrightarrow{a} + \alpha(\overrightarrow{b} - \overrightarrow{c})$
25:         $P_i \leftarrow$ one child from Crossover($d$,Copy($Q_i$))
26:     end for
27: end while
28: return Best
```

Table 2.1: An example of complete factorial experiment

| Combination No. | Factors | | | Parameters | | | Response variable |
|---|---|---|---|---|---|---|---|
| | $F_1$ | $F_2$ | $F_3$ | $x_1$ | $x_2$ | $x_3$ | |
| 1 | 1 | 1 | 1 | -1 | 1 | 3 | -4 |
| 2 | 1 | 1 | 2 | -1 | 1 | 4 | -3 |
| 3 | 1 | 2 | 1 | -1 | 2 | 3 | -6 |
| 4 | 1 | 2 | 2 | -1 | 2 | 4 | -5 |
| 5 | 2 | 1 | 1 | 1 | 1 | 3 | 6 |
| 6 | 2 | 1 | 2 | 1 | 1 | 4 | 7 |
| 7 | 2 | 2 | 1 | 1 | 2 | 3 | 4 |
| 8 | 2 | 2 | 2 | 1 | 2 | 4 | 5 |

Using OA, the best combination $(x_1, x_2, x_3) = (1, 1, 4)$ can be obtained by judiciously selecting a subset of the combinations. The procedure of constructing OA is described below.

Let, $N$ be the number of factors, each of which has two levels. Thus, a complete factorial experiment will include trying out $2^N$ combinations. To construct an OA with $N$ factors, let, $n = 2^{\lceil log_2(N+1) \rceil}$. Now, build a two level OA, $L_n(2^{n-1})$, with $n$ rows and $n-1$ columns. Only the first $N$ columns of this table will be used. The levels of the factors are denoted by 1 and 2 in each column. Each column has the same number of 1s and 2s. Each of the four pairs $(1, 1)$, $(1, 2)$, $(2, 1)$ and $(2, 2)$ will appear same number of times in each possible pair of the $n-1$ columns. For example, an $L_4(2^3)$ is as follows.

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 2 | 2 |
| 2 | 1 | 2 |
| 2 | 2 | 1 |

The algorithm used to generate OAs is described in Algorithm 3 (taken from [35]).

**Algorithm 3** Creating Orthogonal Array OA

```
Require: $N \geq 1$
1: $n \leftarrow 2^{\lceil \log_2 N+1 \rceil}$
2: for $i$ $from$ $1$ $to$ $N$ do
3:     for $j$ $from$ $1$ $to$ $N$ do
4:         $level \leftarrow 0$
5:         $k \leftarrow 0$
6:         $mask \leftarrow \frac{n}{2}$
7:         while $k \geq 0$ do
8:             if $k\%2 \neq 0$ & $bitwise\_AND(i-1, mask) \neq 0$ then
9:                 $level \leftarrow (level+1)\%2$
10:            end if
11:            $k \leftarrow \lfloor \frac{k}{2} \rfloor$
12:            $mask \leftarrow \frac{mask}{2}$
13:        end while
```

### 2.2.1 Factor analysis with orthogonal array

Factor analysis refers to the computation of the impact of each factor on the function to be optimized, sort the factors according to their effectiveness, and find the better of the two levels of each factor. An example of orthogonal experiment using an orthogonal array $L_K(2^{K-1})$ with $K$ rows and $K-1$ columns is shown in Table 2.2. In this example, $K = 4$ has been used, there are three factors, all of which have two levels (1 or 2). 1 corresponds to the exclusion and 2 corresponds to the incorporation of the factor in the proposed feature selection. Let $f_p$ denote the value of the objective function for combination $p$. Let, the contribution of factor $j$ with level $k$ be denoted as $S_{jk}$ where $j = 1, 2, 3, \dots, K-1$ and $k = 1, 2$ as:

$$S_{jk} = \sum f_p \cdot F_p \tag{2.1}$$

where $F_p = 1$ when factor $j$ of combination $p$ has level $k$; else, $F_p = 0$. If the optimization problem is a maximization problem, the level 1 of factor $j$ will be better than level 2 when $S_{j1} > S_{j2}$. After the best level for each factor is determined, we can create a combination by incorporating all the best levels.

Table 2.2: A sample orthogonal experiment

| $t$ | **Factors** | | | $f_t$ | **Rank** |
|---|---|---|---|---|---|
| | 1 | 2 | 3 | | |
| 1 | 1 | 1 | 1 | 35.3% | 6/8 |
| 2 | 1 | 2 | 2 | 32.1% | 8/8 |
| 3 | 2 | 1 | 2 | 60.5% | 2/8 |
| 4 | 2 | 2 | 1 | 57.8% | 3/8 |
| $S_{j1}$ | 67.4 | 95.8 | 93.1 | | |
| $S_{j2}$ | 118.3 | 89.9 | 92.6 | | |
| MED | 50.9 | 5.9 | 0.5 | | |
| Rank | 1 | 2 | 3 | | |
| Better level | 2 | 1 | 1 | 63.2% | 1/8 |

The Rank column in Table 2.2 shows the position of the combination $p$ among all ($2^3 = 8$) combinations. In this example, the created best combination gets the best value for the objective function which may not be the case in practical life.

## 2.3 Statistical measures

### 2.3.1 Correlation coefficient

Correlation coefficient is used to measure the relationship between two variables. The coefficient can be any value between $-1.0$ and $1.0$. A value of $-1.0$ means the two variables have completely opposite relationship (e.g., increase in one variable will result in decrease in another variable). On the other hand, a value of $1.0$ refers to a perfect relationship between the two variables. A value of $0.0$ means there is no relationship between the two variables.

#### Pearson Correlation Coefficient

Pearson correlation coefficient ($\rho$) is used to compute the linear relationship between two sets of data. It is defined as the ratio of the covariance of two variables and the product of their standard deviations; hence it can be regarded as a normalized measurement of the covariance, so that the result always ranges between $-1.0$ and $1.0$.

Let, two random variables $(X, Y)$, the equation for $\rho$ is:

$$\rho_{X,Y} = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{2.2}$$

where

- $cov$ is the covariance.
- $\sigma_X$ and $\sigma_Y$ are the standard deviations of $X$ and $Y$, respectively.

### Spearman Correlation Coefficient

The Spearman correlation coefficient is defined as the Pearson correlation coefficient between the rank of the two variables. For $n$ values of each of the two variables $X$ and $Y$, the $n$ scores $X_i$ and $Y_i$ are transformed into ranks R$(X_i)$, $R(Y_i)$. Next, $r_s$ is computed as

$$r_s = \rho_{R(X),R(Y)} = \frac{cov(R(X), R(Y))}{\sigma_{R(X)} \sigma_{R(Y)}} \tag{2.3}$$

where

- $\rho$ denotes the typical Pearson correlation coefficient, but applied to the ranks of the two variables.
- $cov(R(X), R(Y))$ is the covariance of the ranks of the two variables.
- $\sigma_{R(X)}$ and $\sigma_{R(Y)}$ are the standard deviations of the ranks of the two variables.

If all $n$ rank values are distinct integers, $r_s$ can be calculated using the following formula.

$$r_s = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{2.4}$$

where

- $d_i = R(X_i) - R(Y_i)$ is the difference between the two ranks of each observation.
- $n$ is the total number of observations.

### 2.3.2 Permutation feature importance

The *permutation feature importance* for each feature is calculated by randomly shuffling the values of that feature and measuring the decrease in accuracy due to the shuffling. The higher value of decrease in accuracy means that feature is more important for the prediction task at hand.

One thing to note that, if a feature has a low feature importance score with respect to a predictor does not necessarily mean that the feature is irrelevant. It may be possible that if the classifier is improved, the importance of that feature will be increased. Hence, it is important to first evaluate the model on a held-out test set. Only if the model's performance is satisfactory, we should run permutation feature importance algorithm to understand the contribution of each of the features on the model's performance being satisfactory.

It was introduced by [36] for random forests. Based on this idea, [37] came up with a model-independent version of the algorithm and named it model reliance.

The algorithm for permutation feature importance is stated below.

**Input**: Trained model $\hat{f}$, feature matrix $X$, target vector $y$, error measure $L(y, \hat{f})$.

1. Estimate the original model error $e_{orig} = L(y, \hat{f}(X))$
2. For each feature $j \in \{1, 2, \ldots, p\}$ do
   - Generate feature matrix $X_{perm}$ by permuting feature $j$ in the data $X$. This breaks the association between feature $j$ and true outcome $y$.
   - Estimate error $e_{perm} = L(y, \hat{f}(X_{perm}))$ based on the predictions of the permuted data.
   - Calculate permutation feature importance as quotient $FI_j = \frac{e_{perm}}{e_{orig}}$ or difference $FI_j = e_{perm} - e_{orig}$
3. Sort features by descending order of $FI$.

## 2.4 Machine learning

### 2.4.1 Random forest

Random forest is composed of a number of single decision trees and operates as an ensemble. Each decision tree in the random forest gives a prediction and the class with the most votes from the decision trees becomes the model's final prediction.

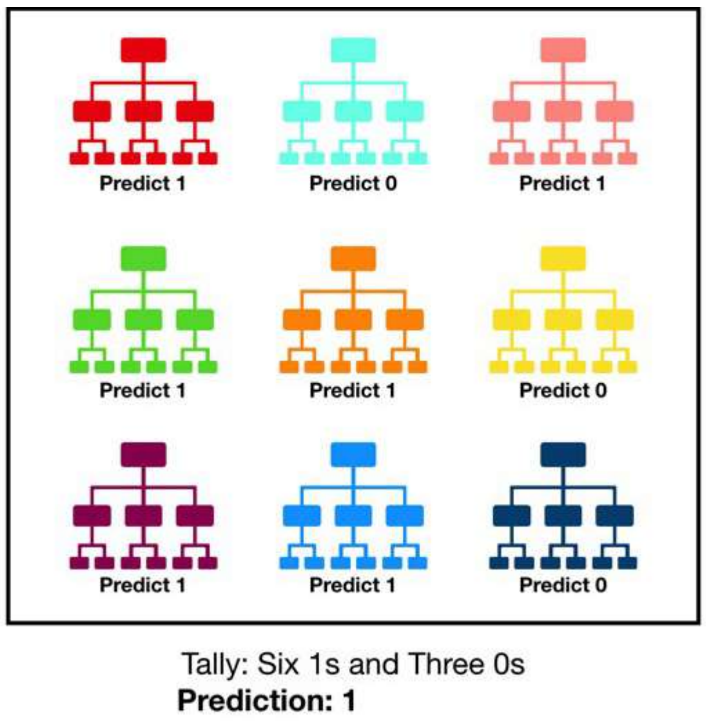


Figure 2.1: Visualization of a Random Forest Model Making a Prediction. Figure borrowed from medium.com.

### 2.4.2 Recurrent neural network (RNN)

Recurrent neural networks (RNNs) are a kind of neural networks that are capable of using previous outputs as inputs while having hidden states. Structure of an RNN is shown in Figure 2.2.

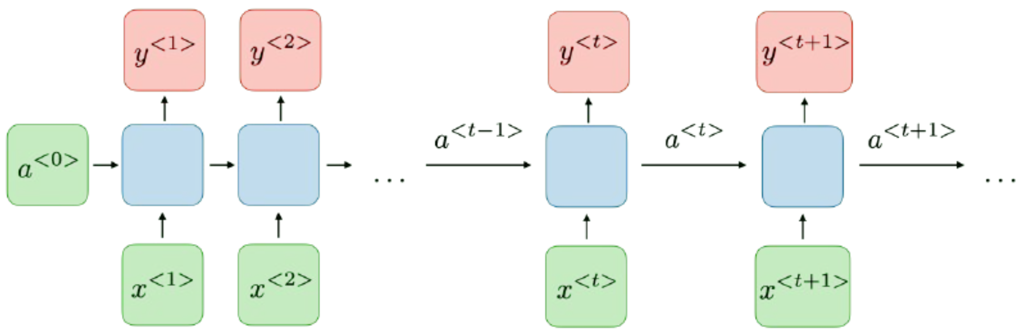


Figure 2.2: Structure of a typical RNN. Figure borrowed from stanford.edu.

#### Long-Short Term Memory (LSTM)

LSTMs are a special kind of RNN introduced by [38]. They are specially designed to remember important information which may be far away from the point where it is needed.

### 2.4.3 Convolutional neural network (CNN)

2 dimensional convolution neural network (2D CNN) is mostly popular because of its success in image classification. But there are two other classes of Convolution Neural Networks used in the real world, that are 1 dimensional (1D CNN) and 3-dimensional CNNs (3D CNN). We talk about 1D and 2D CNNs below.

#### 2D CNN

It is usually applied on image data. It is called 2D CNN because the kernel slides along 2 dimensions on the data as shown in Figure 2.3.

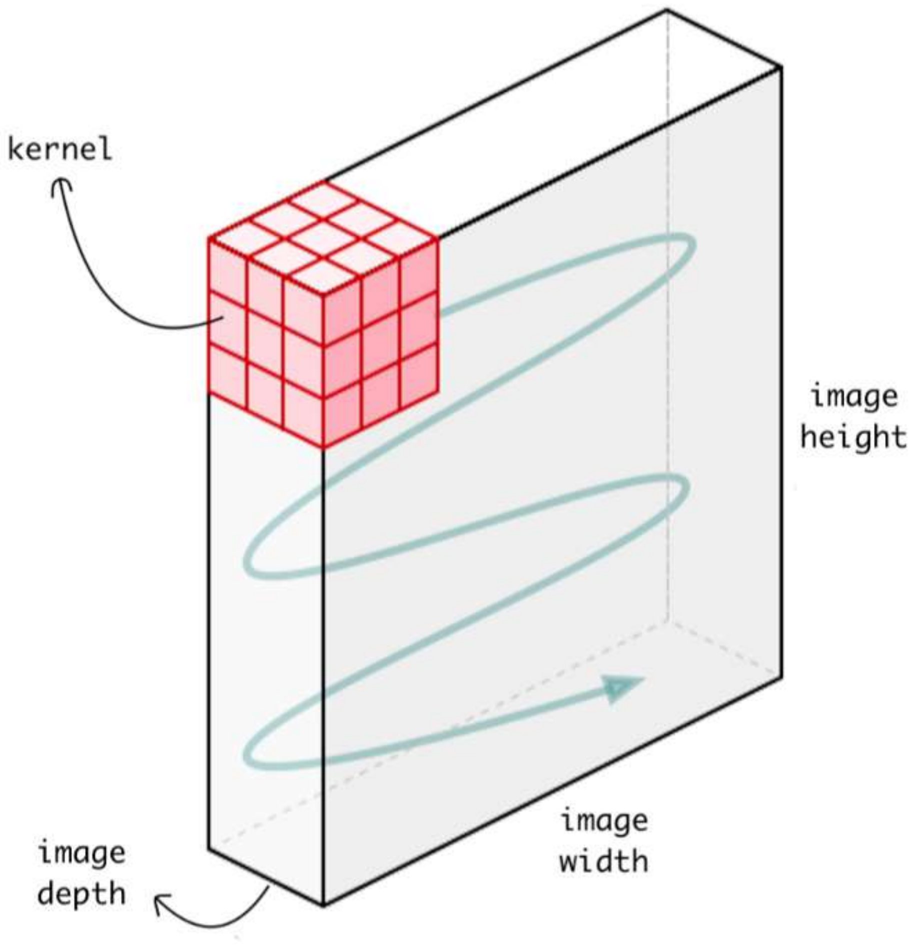


Figure 2.3: Kernel sliding over the image. Figure borrowed from medium.com.

The benefit of using CNN is that it can extract the local features from the data using its kernel, which other networks can't. For instance, CNN can detect edges of an image, distribution of colours etc which makes these networks very effective in image classification and other similar tasks which contain spatial properties.

**1D CNN**

In 1D CNN, kernel slides in one dimension. Hence, 1D CNN is effective for sequential data like time-series data, audio and text data. Operation of 1D CNN layer on sequential data is depicted in Figure 2.4.

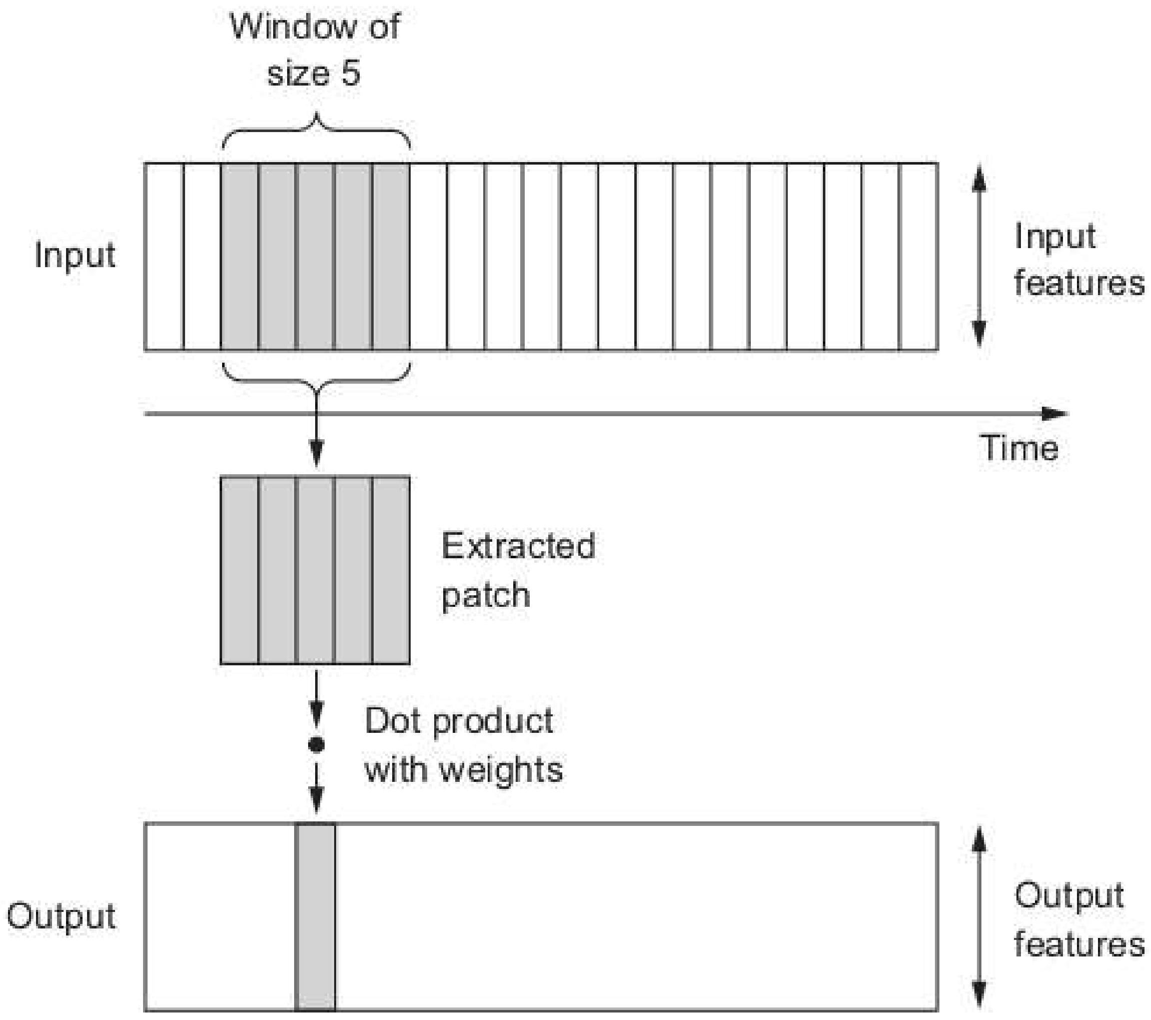


Figure 2.4: Each output timestep is obtained from a temporal patch in the input sequence. Figure borrowed from learnremote.medium.com.

### 2.4.4 Loss function

The objective function is used to evaluate candidate solutions in the context of optimization algorithms. In general, we want to minimize error in case of neural networks. In this case, the objective function is referred to as the loss function.

The three most popular loss functions for binary classifications are:

**Binary Cross-Entropy Loss** Let, there are $N$ samples in a dataset and $y_i$ denote the label (0 or 1) of the $i^{th}$ sample. Then, binary cross-entropy loss of these $N$ samples is calculated as follows:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log p_i + (1 - y_i) \log (1 - p_i)), \tag{2.5}$$

where, $p_i$ is the predicted probability of the $i^{th}$ sample to belong to class 1.

**Hinge Loss** For computation of hinge loss, the outputs must be in the set $\{-1, 1\}$. If the expected output is denoted by $t\ ( \ = \pm 1)$ and the output of the classifier is $y$, the hinge loss of the prediction $y$ is computed by the formula:

$$l(y) = max(0, 1 - t \cdot y) \tag{2.6}$$

**Squared Hinge Loss** When the hinge loss does not achieve expected result on a binary classification problem, it is often found that a squared hinge loss may be suitable to use. For computation of squared hinge loss, the outputs must be in the set $\{-1, 1\}$. If the expected output is denoted by $t\ ( \ = \pm 1)$ and the output of the classifier is $y$, the squared hinge loss of the prediction $y$ is computed by the formula:

$$l(y) = max(0, 1 - t \cdot y)^2 \tag{2.7}$$

### 2.4.5 Data leakage

Data leakage occurs when any information from outside the training dataset is used for the training of the model. With this additional information, the model may achieve a performance which would have otherwise been impossible to achieve. This is a common phenomenon in the machine learning era which everyone should be careful about to construct a credible model.

## 2.5 Performance evaluation

Some popular metrics for binary classification problems are Specificity (SP), Sensitivity (SN), Accuracy (ACC), Matthews Correlation Coefficient (MCC). The equations to determine these metrics are as follows.

$$Specificity = \frac{TN}{TN + FP}$$

$$Sensitivity = \frac{TP}{TP + FN}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FP}$$

$$MCC = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP + FN) \times (TN + FP) \times (TP + FP) \times (TN + FN)}}$$

Here, $TP$, $TN$, $FP$, and $FN$ corresponds to True Positive, True Negative, False Positive, and False Negative, respectively.

### 2.5.1 Sensitivity-specificity tradeoff

SN is the fraction of positive samples correctly classified whereas SP is the fraction of positive samples correctly classified. If a model blindly predicts each of the samples as positive (negative), the SN (SP) will be 1 and SP (SN) will be 0. Therefore, it is important to compute both of these metrics while evaluating a binary classifier. Whether higher SN is more important than higher SP or the opposite is context-dependent. For example, for spam detection of mails, it is more important to detect spams even if it causes some non-spams being detected as spams.

In our thesis, we are concerned with predicting lysine residues as succinylated or non-succinylated. Although predicting both succinylated residues as succinylated and non-succinylated residues as non-succinylated are important, the main users of our tool (i.e., the biologists) won't want to miss out on the succinylated residues even if that results in a relatively higher number of false positives. However, a very high SN with high false positive rate is also not desired because that will make the biologists do laboratory experiments for higher number of lysine residues. We will refer to this phenomenon as the **"Sensitivity-Specificity Tradeoff"**.

### 2.5.2 Receiver operating characteristic (ROC) curve

ROC curve is a graph depicting the performance of a classification model at various thresholds. This curve plots two parameters:

- True Positive Rate (TPR)
- True Positive Rate (FPR)

TPR is a synonym for SN. FPR is defined as follows.

$$FPR = \frac{FP}{FP + TN}$$

ROC curve plots TPR vs. FPR at different thresholds. Decreasing the value of threshold makes the model to classify more samples as positive, hence increasing both False Positives and True Positives. Figure 2.5 shows a typical ROC curve.

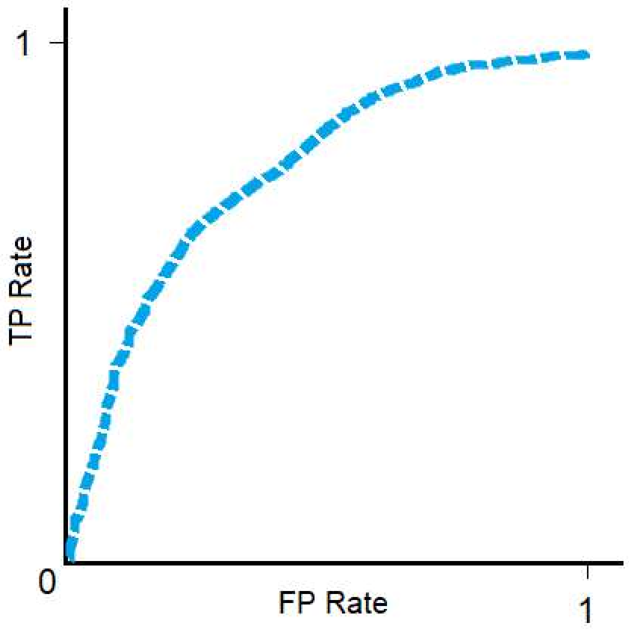
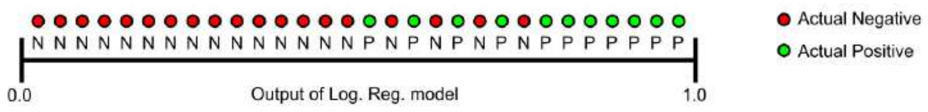


Figure 2.5: A sample ROC curve.

### 2.5.3 Area under the ROC curve (AUC)

AUC measures the area under the ROC curve that spans from $(0, 0)$ to $(1, 1)$. AUC provides an accumulated measurement of performance across all possible thresholds. One way of interpreting AUC is as the probability that the model ranks a random positive example more highly than a random negative example. For instance, provided the examples in Figure 2.6, which are arranged from left to right in ascending order of logistic regression predictions: AUC represents the



N N N N N N N N N N N N N N N N P N P N P N P N P P P P P P P
Actual Negative
Actual Positive
0.0
Output of Log. Reg. model
1.0

Figure 2.6: Predictions sorted in ascending order of logistic regression score. Figure borrowed from developers.google.com.

probability that a random positive (green) sample is positioned to the right of a random negative (red) sample. AUC ranges from 0 to 1. A model with AUC of $0.0$ misclassifies all the samples while an AUC of $1.0$ means the model is $100\%$ correct.

## 2.6 Conclusion

In this chapter, we have elaborately discussed the concepts and techniques used in our thesis. In the next chapter, we will go through the existing works related to SLR prediction and point out the gaps of those works in order to understand the rooms for improvements.

# Chapter 3

# Literature review

Previous chapter has laid the foundation of this thesis by presenting the background materials required to understand the underlying concepts and leveraged techniques. In this chapter, we critically review the literature with a goal to present the state-of-the-art for the task at hand, i.e., in silico succinylation site prediction. Since 2015, many prediction tools for detecting succinylation sites have been developed. These tools can be broadly categorized into two categories, namely, Traditional ML based tools and Deep learning based tools. Below we elaborate on these works from the literature by presenting them in thse two groups.

## 3.1 Traditional ML based tools

To the best of our knowledge, the first succinylation site predictor, named iSuc-PseAAC, was presented in [22]. This predictor used Support Vector Machine (SVM) to compute score of sites using the position specificity and count of amino acids in the peptide chain. As this was the first predictor, they could not do any comparative analysis with any existing predictor. They could only report cross-validation score on CPLM dataset [39] and the reported sensitivity was only around 50%. That means almost half of the succinylated lysine residues were being misclassified by this predictor.

Around the same time, another group independently developed another predictor named SuccFind [40]. It used sequence-derived features like amino acid composition (AAC) and composition of $k$-spaced amino acid pairs (CKSAAP). They run SVM with default parameters on the biophysico properties to evaluate each property's ability to distinguish succinylated lysine residues from non-succinylated ones. They found isoelectric point to be the best performing property and selected this property as another sequence-based feature. In addition to that, they extracted short peptides around the lysine residues and searched for similar peptides in the whole dataset and extracted features from those similar peptides which they termed as the *evolutionary-based features*. They used a two-step feature selection strategy via SVM to filter out the important features for better prediction of succinylation. Due to lack of existing works, they could not do comparative analysis too. However, they only reported the AUC values of cross-validation experiments. But, AUC can not alone depict the true performance of a succinylation predictor. Detecting succinylated sites is important even at the cost of a few more false positives. Another predictor, SucPred [41] also exploits sequence based features like SuccFind. It used four types of sequence-based features, including auto-correlation functions (ACF), the encoding based on grouped weight (EBGW), the normalized van der waals volume (VDWV) and the position weight amino acids composition (WAAC), to transform each succinylated residue into a feature vector. The semi-supervised machine learning algorithm (using only positive samples) with SVM was used to build the succinylation site predictor. They reported very good cross-validation performance. But, they only reported sensitivity score in the independent test set and excluded specificity score. Without specificity, sensitivity is of little value because a model can easily predict all the samples as positive to achieve a sensitivity of 1 at the cost of a specificity of 0.

One year later, iSuc-PseOpt [23] was proposed which adopted pseudo amino-acid composition (PseAAC) [42, 43] to encode the peptides surrounding the lysine residues. They made a mistake while preprocessing the dataset to make it a balanced dataset. For each of the negative samples, they computed its three nearest neighbours. If at least one of these three nearest neighbours is positive, they discarded that negative sample. However, later on while evaluating their proposed predictor, they mentioned that it was sufficient to conduct a jackknife [44] or K-fold cross validation test because the gathered performance thereof will be the result of testing the predictor on numerous indepentdent test datasets. Although we mentioned that during the preprocessing step, they used the complete dataset which nullifies the possibility of any part of the dataset to be independent test dataset. iSuc-PseOpt did a comparative analysis with iSuc-PseAAC and showed significant improvement. However, the data leakage in the preprocessing stage of building iSuc-PseOpt invalidates the credibility of the reported performance. Within a very short period, the same authors came up with another predictor, pSuc-Lys [24], which was free from the limitations discussed about the former work. In the preprocessing stage, rather than discarding some negative samples, they chose random subsets of negative samples with equal cardinality of the positive samples' set and trained a Random Forest (RF) for each combination of positive and negative samples' sets. Finally, they trained an ensemble classifier combining the individual random forests. They reported significantly better performance than iSuc-PseAAC and SucPred.

In 2017 and 2018, Dehzangi *et al.* proposed four succinylation predictors as follows.

**SucStruct [45]** It used several secondary-structure based features to build a decision tree based classifier.

**PSSM-Suc [46]** It adopted position-specific scoring matrix (PSSM) calculated by PSI-BLAST [47] as the only features and trained a pruned decision tree.

**Success [48]** It combined the features used by SucStruct and PSSM-Suc to build an SVM based classifier.

**SSEvol-Suc [49]** It integrated PSSM with probability of secondary structure of amino acids to build an Adaboost classifier.

All four of the above-mentioned classifiers made the same mistake done by the authors of [23]. During the preprocessing of the dataset, they discarded negative samples based on k-nearest neighbour (KNN) algorithm where they had to use the information of positive samples. Later during evaluation, these positive samples can not act as the test set. This error abolishes the results reported by all these four predictors.

On the other hand, Hasan *et al.* proposed three predictors in three consecutive years at around the same period (2016, 2017 and 2018) as follows.

**SuccinSite [11]** It used several sequence based features like CKSAAP, one-hot encoding and 350 most informative biophysico properties from AAindex [28]. Minimum redundancy maximum relevance (MRMR) approach was used to select the most useful features from each of these categories and built an RF based classifier.

**SuccinSite2.0 [2]** The difference between Succinsite2.0 and SuccinSite is that in SuccinSite2.0, the authors computed CKSAAP from the PSSM matrix of the amino acid sequence rather than directly from the amino acid sequence and SuccinSite2.0 did not use the biophysico properties. And for handling high dimensional feature space, SuccinSite2.0 used the information gain (IG) optimization of features rather than the MRMR approach.

**GPSuc [25]** GPSuc took twelve high quality biophysico properties from AAindex in addition to AAC, one-hot encoding, PSSM encoding and pCKSAAP (CKSAAP from PSSM) encoding. Wilcoxon rank-sum test was conducted to filter the most important features. A logistic regression model was trained to combine the scores from different RF classifiers.

Although the three above-mentioned works are unique in the sense that they applied several feature selection strategies to reduce the dimesionality of the feature space. However, the train dataset is not the same as the ones used by the previous works. So, the comparison made in these works with other existing works is not valid.

Psucce [26] used one-hot encoding and AAC in addition to top ten biophysico properties. Information gain was used as the feature selection strategy. Several SVM classifiers were ensembled to create the final Psucce predictor. Inspector [50] incorporated several sequence based features like bi-profile bayes (BPB), double BPB, position-specific di amino-acid propensity (PSDAAP) etc. and achieved an improved sensitivity with respect to SuccinSite2.0, Psucce and GPSuc. A very recently developed tool, predML-Site [51] used several sequence-based features including autocorrelation function (ACF), several biophysico properties, pseudo amino acid composition (PseAAC) and used the SVM algorithm for training.

We compile all the traditional-ML based succinylation predictors and their brief descriptions in Table 3.1.

Table 3.1: List of traditional ML based succinylation predictors.

| Predictor | Authors | Main technique | Dataset | Features | Performance |
|---|---|---|---|---|---|
| iSuc-PseAAC [22] | Yan Xu *et al.* | SVM | CPLM [39] | Position Specific Amino Acid Propensity (PSAAP) [52, 53] | Reported cross-validation performance with SN around 50%. |
| SuccFind [40] | Hao-Dong Xu *et al.* | Information gain to filter out important features and SVM for classification | UniProtKB / Swiss-Prot and CPLM databases | Composition of k-spaced amino acid pairs (CKSAAP), Amino acid composition (AAC), Isoelectric point from AAindex [28] and local sequence clusters | Reported AUC values with 4-, 6-, 8-, 10-fold cross-validation. |
| SucPred [41] | Xiaowei Zhao *et al.* | Semi-supervised using SVM with only positive samples | CPLM | Four sequence based features: Auto-correlation functions (ACF), the encoding based on grouped weight (EBGW), the normalized van der waals volume (VDWV) and the position weight amino acids composition (WAAC) | Reported only sensitivity of around 0.9 without mentioning anything about specificity and performed comparison with other works only with respect to sensitivity |
| iSuc-PseOpt [23] | Jianhua Jia *et al.* | k-nearest neighbor (KNN) for making the dataset balanced and Random Forest for training the classifier | CPLM | Pseudo amino-acid composition (PseAAC) | Performed comparative analysis with iSuc-PseAAC and showed around 70% sensitivity with the cross-validation experiments |
| pSuc-Lys [24] | Jianhua Jia *et al.* | Ensemble of Random Forests trained on different sets of negative samples' set combined with the set of positive samples | CPLM | PseAAC | Performed comparative analysis with iSuc-PseAAC and SucPred and reported sensitivity around 77%. |
| SucStruct [45] | Yosvany López *et al.* | Decision Tree | CPLM and CPLA 1.0 [54] | Accessible surface area (ASA), backbone torsion angles, probability of amino acid contribution to local structure conformations (helix, strand and coil) | Conducted 10-fold cross validation and reported a sensitivity of 80%. Performed comparative analysis iSuc-PseOpt and SuccinSite. |
| PSSM-Suc [46] | Abdollah Dehzangi *et al.* | KNN to handle the imbalance of dataset and Pruned Decision Tree as the classifier | CPLM and CPLA 1.0 | PSSM with bigram | Reported a sensitivity of around 82% |
| Success [48] | Yosvany López *et al.* | KNN to handle the imbalance of dataset and SVM as the classifier | CPLM and CPLA 1.0 | PSSM, ASA, secondary structure and backbone torsion angles | Reported a sensitivity of around 86% |
| SuccinSite [11] | Md. Mehedi Hasan *et al.* | MRMR for feature selection and random forest as the classifier | UniProtKB / Swiss-Prot and NCBI protein sequence | CKSAAP, one-hot and 350 most informative biophysico properties from AAindex | Reported a sensitivity of 37% on the independent dataset |
| SuccinSite2.0 [2] | Md Mehedi hasan *et al.* | Information gain for feature selection and random forest as the classifier | UniProtKB / Swiss-Prot and NCBI protein sequence | CKSAAP with PSSM, one-hot encoding | Reported a sensitivity of 46% on the independent dataset |
| GPSuc [25] | Md. Mehedi Hasan and Hiroyuki Kurata | Wilcoxon rank-sum test for feature selection and logistic regression on several random forest classifiers to predict succinylation | UniProtKB / Swiss-Prot and NCBI protein sequence | AAC, one-hot encoding, 12 high quality biophysico properties from AAindex, PSSM, CKSAAP with PSSM | Reported a sensitivity of 50% on the independent dataset |
| Psucce [26] | Qiao Ning *et al.* | Information gain to select important features and ensembling several SVM classifiers for prediction of succinylation | UniProtKB / Swiss-Prot and NCBI protein sequence | AAC, one-hot encoding and top ten properties from AAindex | Psucce shows a sensitivity of 37% with a threshold of 0.9 which is significantly better than other contemporary predictors |
| Inspector [50] | Yan Zhua *et al.* | ENN undersampling and ADASYN oversampling for balancing the dataset and RF for classification | UniProtKB / Swiss-Prot and NCBI protein sequence | Bi-profile Bayes (BPB), Double BPB (DBPB), Position-specific di-amino acid propensity (PSDAAP), PseAAC, Position-weight amino acid (PWAA) composition, Enhanced grouped amino acid composition (EGAAC), CKSAAP | Reported a sensitivity of 69% on the independent dataset |
| predML-Site [51] | Sabit Ahmed *et al.* | SVM | UniProtKB / Swiss-Prot and NCBI protein sequence | ACF, several biophysico properties, PseAAC | Reported a sensitivity of 10% on the independent dataset |

## 3.2 Deep learning based tools

A novel deep-learning based predictor named MUscADEL [55] was developed using recurrent neural network (RNN) for predicting eight types of post-translational modifications including succinylation. It did a comparative analysis with only two existing works, namely, iSuc-PseAAC and SuccinSite2.0 using an independent testing dataset. But, nothing has been clearly mentioned about the independent dataset. For doing comparison with other works, all the works should be trained on the same training dataset and then should be tested on an independent test set. However, the authors only used the independent test set to gather the performances of other works from the published webservers which is not the correct way of comparison. Moreover, the training and testing datasets have not been made public. The codebase of the implementation of MUscADEL has not been shared either which made it impossible to compare this work with any other work.

Later on, HybridSucc [56] was developed by integrating traditional ML algorithms like Penalized Logistic Regression (PLR), SVM, Random Forest with Deep Neural Networks (DNNs). It achieved significantly better AUC values compared to several existing predictors, namely, MUscADEL, SuccinSite, SuccFind, iSuc-PseAAC, iSuc-PseOpt etc. Informatively, the online servers provided by the existing works were used to compute the performances of these works, which may raise an eyebrow because HybridSucc compared its 10-fold cross validation performance with the performance of the existing works on the test dataset although the other predictors were not trained on the same training dataset.

One of the pioneering works in this field, DeepSuccinylSite [27], experimented with both one hot vectors and embedding vectors and fed them into a convolutional neural network comprising of a 2D convolution layer, maxpooling layer and a densely connected neural network. This tool reported significantly better performance compared to previously mentioned predictors. However, the authors in [27] undersampled the test set before evaluating its performance whereas other existing works did not do that. This might have put DeepSuccinylSite in an advantegeous situation.

In 2021, LSTMCNNsucc [57] was proposed which was developed using a deep learning framework consisting of a combination of CNN and LSTM layers. There are numerous hyperparameters in a deep learning model (e.g., embedding dimension, number of filters in convolution, dropout rate, learning rate in optimizer etc.) which need to be tuned with the help of a validation dataset. However, the authors did not mention the use of any validation dataset which puts a question mark on its achieved results. If the test dataset has been used for tuning the hyperparameters, the proposed model is practically useless because of data leakage (details of data leakage is discussed in Section 2.4.5). The authors did a comprehensive comparative analysis with some of the previous works. LSTMCNNsucc achieved a higher Matthews Correlation Coefficient (MCC) compared to DeepSuccinylSite. However, the sensitivity is very low compared to the sensitivity achieved by DeepSuccinylSite. So, the model is performing poorly on the actual context (i.e., succinylation site prediction) at the cost of efficiently detecting non-succinylated sites.

Finally, during the write-up stage of this thesis, the publication of a new predictor, called DeepSucc [1] came to our knowledge. The authors of [1] extracted CKSAAP, ACF, BLOSUM62, AAindex and one-hot features and experimented with a number of architectures, namely, LSTM, CNN and various combination thereof (e.g., LSTM+CNN and CNN+LSTM). One interesting technique used by the authors to improve upon the performance of their individual architectures is that the outputs of different models were combined through a weighted ensembling approach with the weights learned through a densely connected neural network. While the reported performance is significantly better than all other previous predictors, upon careful scrutiny we found that the codebase shared by the authors contained multiple discrepancies. This has led us to look at the results with a pinch of salt. A more elaborate discussion on this will be presented in a forthcoming section (Section 5.3.5).
We compile all the deep learning based succinylation predictors and their brief descriptions in Table 3.2.

Table 3.2: List of deep learning based succinylation predictors.

| **Predictor** | **Authors** | **Technique** | **Dataset** | **Features** | **Performance** |
|---|---|---|---|---|---|
| MUscADEL [55] | Zhen Chen *et al.* | RNN | PhosphoSite-Plus [58] | No hand crafted features used | Although MUscADEL reported better performance than iSuc-PseAAC and SuccinSite2.0, the other predictors were trained on a different dataset, hence the comparison is not credible |
| HybridSucc [56] | Wanshan Ning *et al.* | Integrated traditional ML like penalized Logistic Regression (PLR), SVM, Random Forest with Deep Neural Networks (DNNs) | PLMD 3.0 [59], PhosphoSitePlus and dbPTM [60] | PseAAC, CKSAAP, Position-specific amino acids (OBC), AAindex, ACF, Position-weighted similarity of amino acids (GPS), PSSM, ASA, Secondary structure, and continuous angle information of the local conformation of protein (BTA) | Performed comparative analysis from seven existing predictors, however the online servers were used although the existing works a different training set to train their models |
| DeepSuccinylSite [27] | Niraj Thapa *et al.* | CNN | UniProtKB / Swiss-Prot and NCBI protein sequence | One hot encoding | Undersampled test dataset and performed comparative analysis with several previous works, however the previous did not undersample the test dataset. Hence, the comparison is void |
| LSTMCNNsucc [57] | Guohua Huang *et al.* | LSTM and CNN | PLMD | No hand crafted features | Achieved an MCC of 25% on the independent dataset and had a better SP compared to DeepSuccinylSite at the cost of very low SN |
| DeepSucc [1] | Die Zhang and Shunfang Wang | Different combinations of CNN and LSTM, DNN for combining results of different architectures | UniProtKB / Swiss-Prot and NCBI protein sequence | CKSAAP, ACF, one-hot, BLOSUM62, 31 top biophysico properties from AAindex | Achieved a high specificity and moderate sensitivity on the test dataset. Hence, the MCC is higher than all previous predictors |

## 3.3 Existing works and biophysico properties

As one of the main objectives of our thesis is to find suitable combinations of biophysico properties for better prediction of succinylation, we would like to highlight the existing works that have adopted the biophysico properties as features to predict succinylated sites.

**SuccFind [40]** Only one biophysico property, namely isoelectric point was used. Information gain was used to find this property.

**SuccinSite [11]** 350 most informative biophysico properties from AAindex were adopted. MRMR approach was used to select these top 350 properties.

**GPSuc [25]** 12 high quality biophysico properties from AAindex were elected. Wilcoxon rank-sum test was used for feature selection.

**Psucce [26]** Selected the top 10 biophysico properties by ranking them with respect to their abilities to distinguish between succinylated and non-succinylated lyesine residues.

**HybridSucc [56]** No mention of how many properties were used or how they used them.

**DeepSucc [1]** 31 top biophysico properties from AAindex

## 3.4 Conclusion

In this chapter, we critically reviewed the existing works and discussed about the research gaps in the context of SLR prediction. In the next chapter, we will illustrate how we select better combination of biophysico properties, assess the statistical significane of the selected properties and show the performance on succinylation site prediction with the selected combination of properties.

# Chapter 4

# Selection of biophysico properties

In the previous chapter, we have critically reviewed the existing literature on succinylation site prediction in general and we have also discussed the state of the art with respect to the use of biophysico properties in this context. In this chapter, we concentrate on the selection of suitable combination of biophysico properties with a goal to efficiently predict succinylated sites. Additionally, we assess the statistical significance of the features being used for prediction.

The 3D structures and biological functions of proteins are determined by the combination of the 20 different amino acids as specified by the genetic code [28]. Numerous experimental and theoretical research have been performed to characterize biophysico properties of individual amino acids. Each derived property is represented by a set of 20 numerical values that is called the amino acid index. AAindex database [28] is a flat file database which currently (Last checked: Sunday 22$^{\text{nd}}$ May, 2022) contains 566 amino acid indices representing various biophysico properties of amino acids.

## 4.1 Introduction

The collection of amino acid indices is a significant resource for empirical analyses correlating sequence information with structural and functional properties of proteins [61]. The amino acid indices have been successfully adopted to predict numerous functional and structural properties of proteins. For instance, Han *et al.* in [62] used amino acid indices to predict short and long disordered regions in proteins, Liu *et al.* in [63] converted protein sequences in numeric vectors through amino acid indices for efficient detection of protein remote homology. The amino acid indices have been used for the prediction of several PTM sites (e.g. malonylation [64], S-sulfenylation [65]).

As has already been discussed in Chapter 3, several works have used these biophysico properties for predicting succinylation sites. Recall that, Hasan *et al.* in [11] chose 350 most informative amino acid indices via the minimum redundancy maximum relevance (mRMR) feature selection approach. To add to that, the same authors in [25] used 12 high-quality amino acid indices to generate the feature vectors from amino acid sequences. Furthermore, Ning *et al.* in [26] ranked the amino acid indices according to their abilities to distinguish succinylated residues from non-succinylated residues and took the top ten amino acid indices. Very recently, Zhang *et al.* in [1] selected 31 amino acid indices mentioning that these are the most common amino acid indices. However, none of these works have taken the inter-dependency among the biophysico properties into consideration.

There are total $2^{566} - 1$ (-1 for excluding the combination with no properties) possible combinations of biophysico properties. Exhaustive search for the best combination among these for efficient prediction of succinylation is a computationally infeasible task. Hence, adopting metaheuristics based algorithms for finding sub-optimal combinations is a suitable choice in this regard.

## 4.2 Methods

In this section, we discuss the methodologies adopted in order to search for suitable combinations of biophysico properties. First, we discuss about the datasets that we will use for evaluating our chosen combinations and compare with the existing works. Then, we talk about how the lysine residues will be represented considering its neighboring amino acids. This representation will turn out to be one of the crucial factors that will shape our computational predictor. The following section will discuss how we will reduce our search space by discarding the biophysico properties that are not relevant to the prediction of SLRs. We then discuss about a special crossover algorithm called *Orthogonal array crossover* which we will use as the crossover operation of the genetic algorithm (see Section 2.1.1 to know more about genetic algorithm). Finally, we describe the IBCGA algorithm, a variant of genetic algorithm which leverages the *orthogonal array crossover* to efficiently search for the suitable combinations of biophysico properties.

### 4.2.1 Datasets

Succinylated proteins are collected from the Protein Lysine Modification Database (PLMD) [59]. There are in total 6377 succinylated proteins in this dataset. To remove proteins having above threshold sequence similarity, we apply CD-Hit with cutoff set to 40%. Removing homology, we obtain 3560 proteins. The experimentally verified lysine residues are considered as the positive sites, and all other lysine residues in the same proteins are considered as the negative sites.

We randomly divide the proteins into training and test set in the ratio 4:1 resulting in 2848 proteins in the training set and 712 proteins in the test set. There are 6812 succinylated lysine residues in the training set. But, the number of non-succinylated lysine residues is more than

9 times higher. We randomly discard negative samples in order to make the training dataset balanced. Thus, in the processed training dataset, we have 6812 non-succinylated lysine residues from 2848 proteins. We separate out 300 proteins from the training set as the validation set for tuning several hyper-parameters. So, the final training set contains 2548 proteins. The detailed description of the training, validation and test datasets are given in Table 4.1. We will refer to this dataset as **D1** in subsequent sections.

Table 4.1: PLMD dataset, referred to as D1 dataset henceforth

| **Type** | **Positive Samples** | **Negative Samples** |
|---|---|---|
| Training | 5816 | 5826 |
| Validation | 696 | 686 |
| Test | 1479 | 16457 |

In addition to the above-mentioned dataset, there is another dataset with training and independent test set compiled by [11]. We use this dataset for comparing our model with other existing works. This dataset has been compiled as follows. At first, 10,000 succinylated proteins are collected from nine species. Then, redundancy is removed by applying CD-HIT with cutoff set to 30%. The training dataset contains 2,198 proteins with 4,750 succinylated and 9,500 non-succinylated lysine residues. We randomly separate out 3000 samples from the training set for tuning several hyper-parameters, of which 960 are succinylated and 2040 are non-succinylated lysine residues. The test dataset has 124 proteins with 254 succinylated and 2,977 non-succinylated lysine residues. The detailed description of the training, validation and testing samples for this dataset is given in Table 4.2. We will refer to this dataset as **D2** in subsequent sections.

Table 4.2: Dataset compiled by [2]

| **Type** | **Positive Samples** | **Negative Samples** |
|---|---|---|
| Training | 3790 | 7460 |
| Validation | 960 | 2040 |
| Test | 254 | 2977 |

### 4.2.2 Representation of samples

We refer to the succinylated residues as positive samples and all other lysine residues in the same proteins as negative samples. Each of the lysine residue can be represented by a number amino acids on both its upstream and downstream. For example, if we consider 16 amino acids both on the downstream and upstream of the lysine residue, we will be considering 33 amino acids altogether. We will call this parameter as the *context window*. If there are less number of amino acids on any side of the concerned lysine residue, mirror effect is used to keep the *context window* fixed for each sample. The detailed procedure is depicted in Figure 4.1.

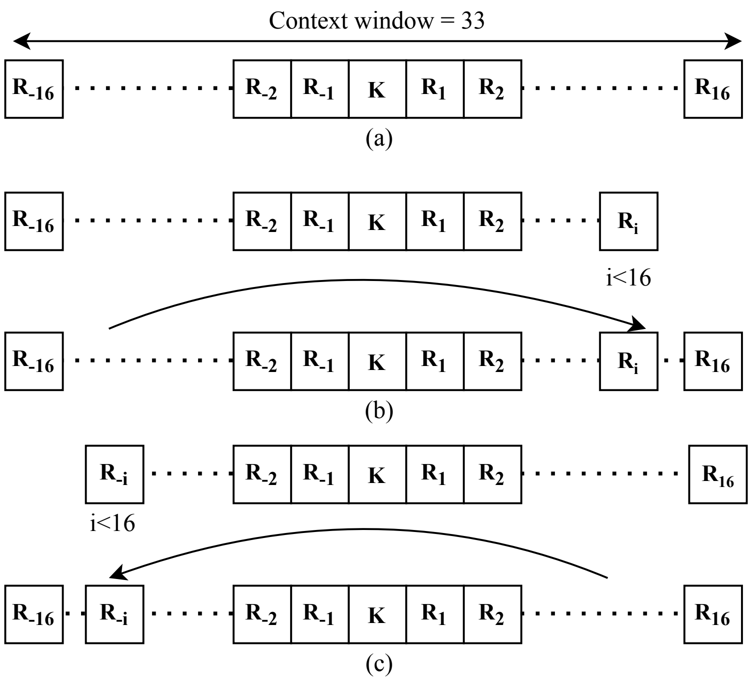


Figure 4.1: Creating positive and negative samples with the neighboring amino acids of the lysine residue. (a) The lysine residue has enough neighbors both on its downstream and upstream. (b) The lysine residue lacks neighbors on its downstream. So, mirror effect is used to bring amino acids from its upstream. (c) The lysine residue lacks neighbors on its upstream. So, mirror effect is used to bring amino acids from its downstream.

### 4.2.3 Reducing the search space

Many of the 566 biophysico properties may not be relevant for the prediction of succinylated sites. Therefore, for each of the 566 properties, a random forest classifier is trained on the training set by replacing each amino acid with its value for the corresponding property. The classifier is then evaluated on the validation set with respect to Matthews Correlation Coefficient (MCC). The 95% percentile of the 566 MCC values is computed and only those features are considered for which the MCC values are greater than the 95% percentile. So, we are remained with $566 \times 0.95 \approx 29$ features. In what follows, we only focus on these 29 features.

### 4.2.4 Orthogonal array crossover

We use orthogonal array crossover as the crossover operation in the inheritable bi-objective genetic algorithm (discussed in the following section) inspired by the concept of orthogonal array (details about orthogonal array is discussed in Section 2.2). Let's introduce some notations for ease of explanation of the algorithm.

- If $S = \{s_i \mid 1 \leq i \leq n\}$, then $S[p : q] = \{s_i \mid p \leq i \leq q\}$.
- $\zeta(S)$ denote the number of 1's in $S$.

- If two samples $S_1, S_2$ are constructed according to the procedure mentioned in Section 4.2.2, the cut point $C = \{c_i \mid 1 \leq i \leq n\}$ of $S_1, S_2$ is defined such that for each element $c_i$ of $C$,
  - $S_1[c_{i-1} : c_i] \neq S_2[c_{i-1} : c_i]$
  - $\zeta(S_1[c_{i-1} : c_i]) = \zeta(S_2[c_{i-1} : c_i])$ (if $i = 1$, then $c_{i-1} = 1$)

  And, no other set with cardinality greater than the cardinality of $C$ has the same properties.

During crossover of two samples $S_1$ and $S_2$, we first calculate their cut point $C$. We consider each element $c_i$ of $C$ as a factor with two levels 1 and 2. 1 means we will consider $S_1[c_{i-1} : c_i]$ to create the child, 2 means we will consider $S_2[c_{i-1} : c_i]$ to create the child. We then create orthogonal array and perform orthogonal experiment and find out the best level combination. The best level combination gives us one child. The second child is created by toggling the value of the worst factor.

### 4.2.5 Inheritable bi-objective genetic algorithm

Inheritable bi-objective genetic algorithm (IBCGA) [66] consists of an intelligent genetic algorithm [35] with an inheritable mechanism. The algorithm adopts a divide and conquer strategy with orthogonal array crossover to solve optimization problems with large number of parameters [34].

In the context of IBCGA, each solution will be called a chromosome containing genes. Each of the biophysico properties works as a binary gene constructing the chromosome. A value of 1 (0) for any binary gene means it is being considered (not considered) for the prediction of succinylation. The feature space is constructed by taking the biophysico properties having value 1 and concatenating the values of these properties for all of the amino acids in the *context window*. The centered lysine residue is excluded in this step because it is the same across all the samples. The detailed steps to calculate the fitness of an individual is demonstrated in Figure 4.2.

At each iteration, the IBCGA algorithm maintains a number of solutions each having $r$ genes with value 1, where $r_{start} \leq r \leq r_{end}$. Here, $r$ is the number of 1's in each chromosome. The steps of the algorithm with the given values of $r_{start}$ and $r_{end}$ are as follows [35].

Step 1. Generate $N_{pop}$ random individuals (chromosomes) each having $r$ genes with value 1.

Step 2. Compute the performance of the individuals using the procedure described in Fig. 4.2.

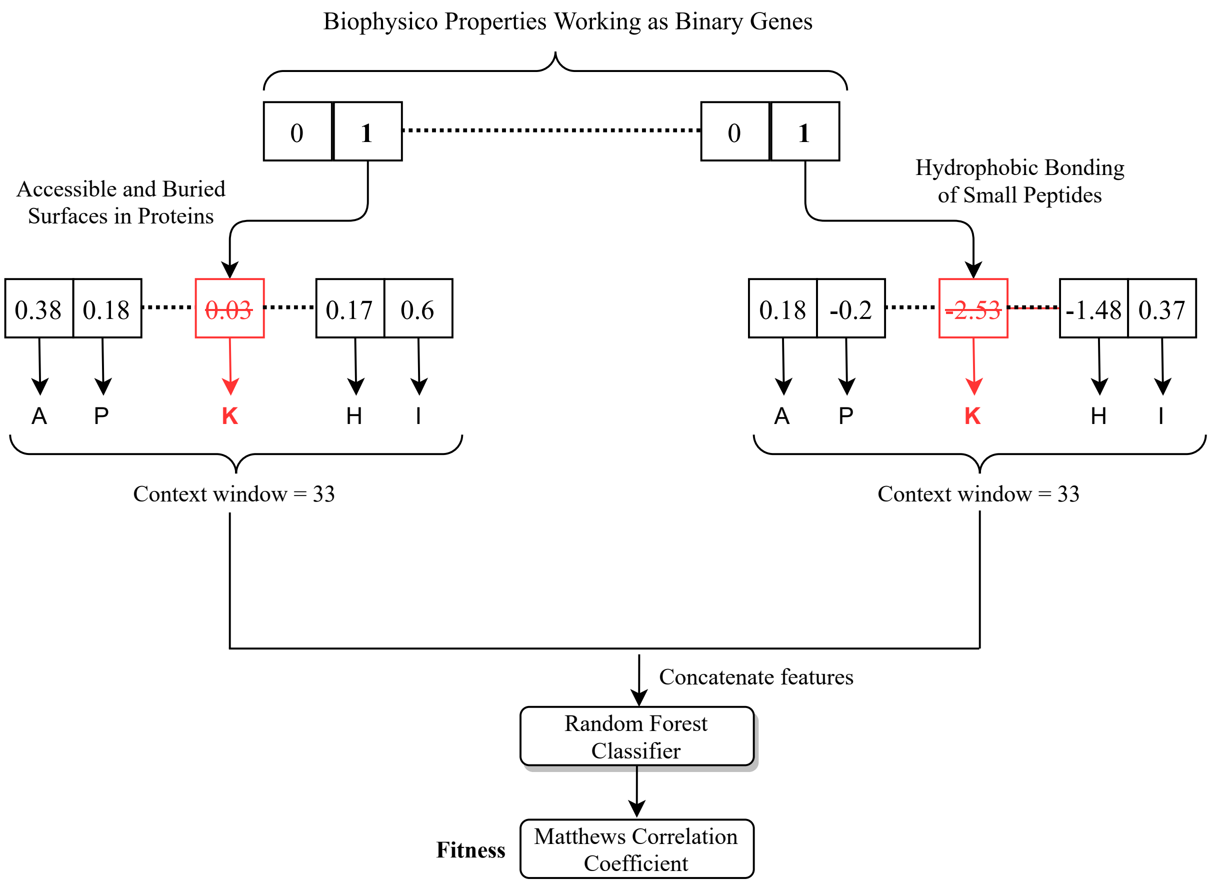

Biophysico Properties Working as Binary Genes
0
1
0
1
Accessible and Buried Surfaces in Proteins
Hydrophobic Bonding of Small Peptides
0.38
0.18
~~0.03~~
0.17
0.6
0.18
-0.2
~~-2.53~~
-1.48
0.37
A
P
K
H
I
A
P
K
H
I
Context window = 33
Context window = 33
Concatenate features
Random Forest Classifier
Fitness
Matthews Correlation Coefficient

Figure 4.2: Evaluation of fitness value for an individual from the chromosome. An individual consists of a number of amino acids, the amino acids from the upstream and downstream of the lysine residue. The lysine residue itself is ignored because it is the same across all the samples. The biophysico features corresponding to the binary genes having value 1 are considered for each of these amino acids. These features are concatenated and random forest classifier is run on the feature space. The Matthews Correlation Coefficient (MCC) on the validation set is considered the fitness function as we are interested in improving both sensitivity and specificity.

Step 3. Select $P_c \times N_{pop}$ individuals from the current population by the 2-way tournament selection algorithm (for more details about tournament selection, please refer back to Section 2.1.1) in pairs and perform orthogonal array crossover (see Section 4.2.4) on each of these pairs. Here, $P_c$ is the crossover probability.

Step 4. Two genes' values are swapped as part of mutation for $P_m \times N_{pop}$ individuals. This mutation is not applied on the best individual for any specific $r$ in order to preserve the best fitness value.

Step 5. Repeat Steps 2 to 4 $N_{iter}$ times. In the $(N_{iter} + 1)th$ time, go to Step 6.

Step 6. Randomly change one binary gene's value from 0 to 1 for each of the $N_{pop}$ individuals in order to increase the value of $r$ by 1. If $r \leq r_{end}$ go to Step 2. Else, terminate the algorithm.

## 4.3 Results and discussions

### 4.3.1 Performance of individual biophysico properties

We set *context window* $= 33$ because the current state-of-the-art [27] found this value to be "optimal". We run random forest on the training set of both D1 and D2 (see Section 4.2.1). For D1, we use the RandomForestClassifier class of sklearn library (version 1.0.2). As D2 is an imbalanced dataset, we use BalancedRandomForestClassifier class of imblearn library (version 0.8.0) for D2. We set *min_samples_leaf = 5*, a value greater than 1 will reduce the depth of each single tree and will degrade the performance of that tree. But for the same reason, the random forest should generalize better. We also set *max_features = "log2"*. Forcing separate trees to use different sets of features will enable the random forest to learn different interactions among the features. The performance of top performing 29 features with respect to MCC (to know about the 29 features, refer to Section 4.2.3) is given in Table 4.3 and 4.4.

### 4.3.2 Performance of combination of biophysico properties

The 29 features derived from Section 4.3.1 is the universal set of properties for the IBCGA algorithm (see Section 4.2.5). We set $r_{start} = 1, r_{end} = 20, N_{pop} = 50, N_{iter} = 5$. We take the values of both $P_m$ and $P_c$ from the set $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ resulting in 25 different combination of $P_m$ and $P_c$. The best MCC values obtained from each value of $r$ for both D1 and D2 is shown in Figure 4.3.

We observe that using all the top 29 biophysico features is giving us comparatively poorer performance if compared with the performances of $r = 2$ to $20$. This proves the necessity of

Table 4.3: Performance of top 29 (with respect to MCC) single biophysico properties on the validation set of D1 with *context window* set to 33. The 0-based **Index** column denotes the serial number of the property according to the AAindex database.

| Index | Sensitivity | Specificity | Accuracy | MCC |
|---|---|---|---|---|
| 33 | 0.6939655172 | 0.6239067055 | 0.6591895803 | 0.3186993207 |
| 35 | 0.6968390805 | 0.6311953353 | 0.6642547033 | 0.3287872541 |
| 67 | 0.683908046 | 0.639941691 | 0.6620839363 | 0.3241890713 |
| 69 | 0.7227011494 | 0.6020408163 | 0.6628075253 | 0.3272185533 |
| 93 | 0.7227011494 | 0.610787172 | 0.6671490593 | 0.335680593 |
| 95 | 0.7112068966 | 0.6034985423 | 0.6577424023 | 0.3166180621 |
| 112 | 0.658045977 | **0.6545189504** | 0.6562952243 | 0.3125619816 |
| 126 | 0.6968390805 | 0.6370262391 | 0.6671490593 | 0.3345048766 |
| 127 | 0.7183908046 | 0.6311953353 | 0.6751085384 | 0.3509926907 |
| 128 | 0.6738505747 | 0.639941691 | 0.6570188133 | 0.3139896168 |
| 129 | 0.7112068966 | 0.6413994169 | 0.6765557164 | 0.3535237888 |
| 130 | 0.7112068966 | 0.6326530612 | 0.6722141823 | 0.3449857207 |
| 145 | 0.7255747126 | 0.6005830904 | 0.6635311143 | 0.3288266938 |
| 150 | 0.6997126437 | 0.6311953353 | 0.6657018813 | 0.3317344258 |
| 152 | 0.724137931 | 0.6137026239 | 0.6693198263 | 0.3400048032 |
| 197 | 0.6709770115 | 0.6428571429 | 0.6570188133 | 0.3139709632 |
| 208 | 0.6767241379 | 0.6355685131 | 0.6562952243 | 0.3125792033 |
| 212 | 0.7183908046 | 0.6326530612 | 0.6758321274 | 0.3524105256 |
| 319 | 0.7212643678 | 0.6137026239 | 0.6678726483 | 0.3370027546 |
| 320 | 0.7126436782 | 0.6297376093 | 0.6714905933 | 0.3436271954 |
| 333 | **0.742816092** | 0.6020408163 | 0.6729377713 | 0.3484425706 |
| 355 | 0.6939655172 | 0.6355685131 | 0.6649782923 | 0.3301357925 |
| 447 | 0.6781609195 | 0.638483965 | 0.6584659913 | 0.3169158141 |
| 487 | 0.7284482759 | 0.5903790087 | 0.6599131693 | 0.3220073811 |
| 517 | 0.6982758621 | 0.6151603499 | 0.6570188133 | 0.3145767902 |
| 518 | 0.7083333333 | 0.6472303207 | **0.6780028944** | **0.3562773624** |
| 521 | 0.7054597701 | 0.6180758017 | 0.6620839363 | 0.3248373089 |
| 522 | 0.7011494253 | 0.6224489796 | 0.6620839363 | 0.3246578336 |
| 525 | 0.691091954 | 0.6326530612 | 0.6620839363 | 0.324336206 |

Table 4.4: Performance of top 29 (with respect to MCC) single biophysico properties on the validation set of D2 with *context window* set to 33. The 0-based **Index** column denotes the serial number of the property according to the AAindex database.

| Index | Sensitivity | Specificity | Accuracy | MCC |
|---|---|---|---|---|
| 34 | 0.6945273632 | 0.637593985 | 0.6566666667 | 0.3139519386 |
| 35 | 0.7004975124 | 0.6325814536 | 0.6553333333 | 0.3146978789 |
| 67 | 0.6855721393 | 0.6516290727 | 0.663 | 0.3192676272 |
| 69 | 0.6915422886 | 0.6380952381 | 0.656 | 0.3116490054 |
| 76 | 0.6786069652 | 0.6551378446 | 0.663 | 0.3162386269 |
| 87 | 0.7084577114 | 0.6250626566 | 0.653 | 0.3149488696 |
| 92 | 0.7064676617 | 0.6546365915 | 0.672 | 0.3416508939 |
| 93 | 0.7054726368 | 0.6471177945 | 0.6666666667 | 0.333399895 |
| 94 | 0.7054726368 | **0.6636591479** | **0.6776666667** | **0.3495737979** |
| 95 | 0.6895522388 | 0.6380952381 | 0.6553333333 | 0.3097907568 |
| 117 | **0.7184079602** | 0.634085213 | 0.6623333333 | 0.3329173315 |
| 126 | 0.6885572139 | 0.6395989975 | 0.656 | 0.3103199199 |
| 127 | 0.6875621891 | 0.6370927318 | 0.654 | 0.3069612945 |
| 128 | 0.6915422886 | 0.6360902256 | 0.6546666667 | 0.3097086758 |
| 130 | 0.6925373134 | 0.645112782 | 0.661 | 0.3193900337 |
| 145 | 0.7064676617 | 0.6541353383 | 0.6716666667 | 0.3411613366 |
| 152 | 0.6845771144 | 0.6471177945 | 0.6596666667 | 0.313928429 |
| 197 | 0.6825870647 | 0.6546365915 | 0.664 | 0.3194438135 |
| 212 | 0.6905472637 | 0.6446115288 | 0.66 | 0.3170471141 |
| 319 | 0.7014925373 | 0.6401002506 | 0.6606666667 | 0.3228827749 |
| 320 | 0.6736318408 | 0.6521303258 | 0.6593333333 | 0.3086568048 |
| 333 | 0.7154228856 | 0.6285714286 | 0.6576666667 | 0.3248395923 |
| 355 | 0.6955223881 | 0.6511278195 | 0.666 | 0.3280358609 |
| 391 | 0.7154228856 | 0.6421052632 | 0.6666666667 | 0.3378369344 |
| 400 | 0.7074626866 | 0.6380952381 | 0.6613333333 | 0.3265259822 |
| 489 | 0.7154228856 | 0.6270676692 | 0.6566666667 | 0.3234029544 |
| 518 | 0.6895522388 | 0.637593985 | 0.655 | 0.3093051686 |
| 521 | 0.6845771144 | 0.6466165414 | 0.6593333333 | 0.3134389713 |
| 525 | 0.6835820896 | 0.6441102757 | 0.6573333333 | 0.3100672087 |

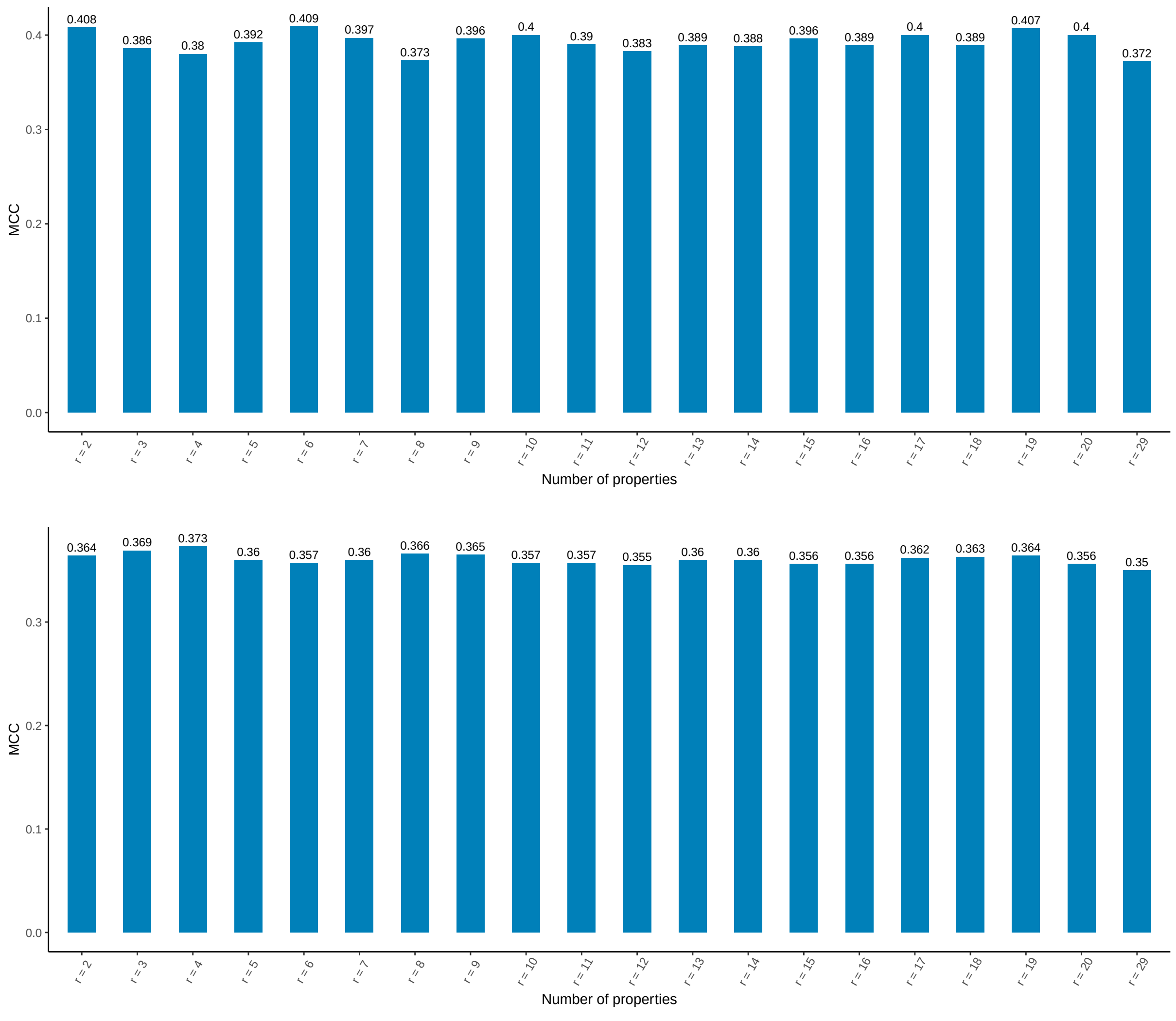


Figure 4.3: The IBCGA algorithm was run for 25 different pairs of values for $P_c$ and $P_m$. The best MCC value obtained among the 25 values for each value of $r$ is recorded. These are the histogram plots of the best MCC values against each value of $r$ from 2 to 20 for D1 and D2. $r = 29$ means we are using all the 29 biophysico properties.

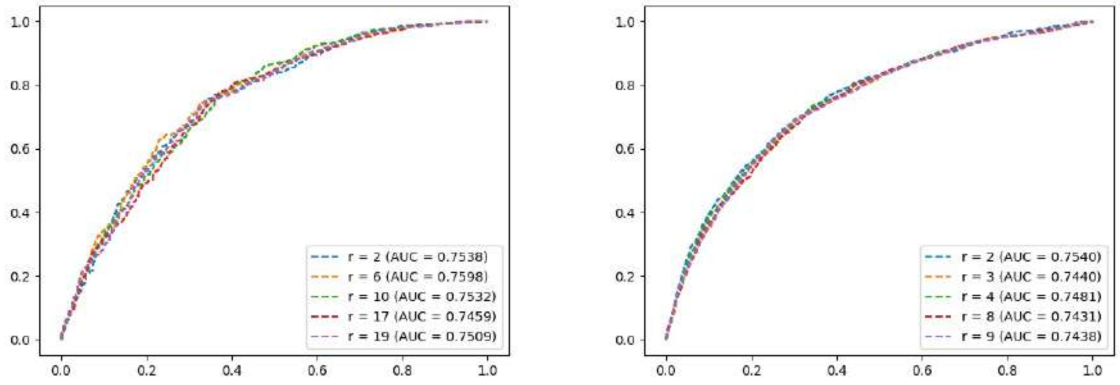


Figure 4.4: We rank the 29 biophysico properties in descending order with respect to the achieved MCC value on the validation dataset of D1 and D2. These are the ROC curves along with AUC values on the validation set of D1 and D2 for the top 5 among the 29 properties.

searching for suitable combinations rather than using all the better performing features as has been done by several works in the literature [1, 11, 25, 26]. Hence, we obtain 19 models trained by Random Forest (Concept of Random Forests are discussed in 2.4.1) for $r = 2$ to $r = 20$.

Notably, we have been able to produce very competitive result with only 2 properties (i.e., $r = 2$) for D1. These two properties are **Average accessible surface area** [67] (126th biophysico property according to the AAindex database. See Table 4.3) and **Net charge** [68] (145th biophysico property according to the AAindex database. See Table 4.3). If we look at their individual performances in Table 4.3, we will see that none of the individual performance metrics are in the higher side. But, together they are performing better than most other combinations of two properties. This again proves that the inter-dependency of the properties should be taken into consideration for any prediction task.

The ROC curves along with the AUC values are computed on the validation dataset of D1 and D2 and plotted in Figure 4.4. To avoid congestion, we choose the top five properties according to the MCC on the validation set. We observe that the performance of each combination is somewhat similar.

### 4.3.3 Correlation among the biophysico properties

If two features are correlated, one feature can be predicted from the other. Hence, it is desired that the features being used for a machine learning task will have low correlation among them. For $r = 2$ properties on D1 (those mentioned in Section 4.3.2), there are $32 \times 2 = 64$ features (2 values for each amino acid because $r = 2$, $context\ window = 33$ means there are 32 amino acids excluding the lysine residue in the center). We compute the pairwise Spearman Correlation Coefficient (see Section 2.3.1) with the training dataset of D1 and draw a heatmap in Figure 4.5. For avoiding congestion, we have drawn the triangular matrix rather than the square matrix as the correlation coefficient is symmetric.

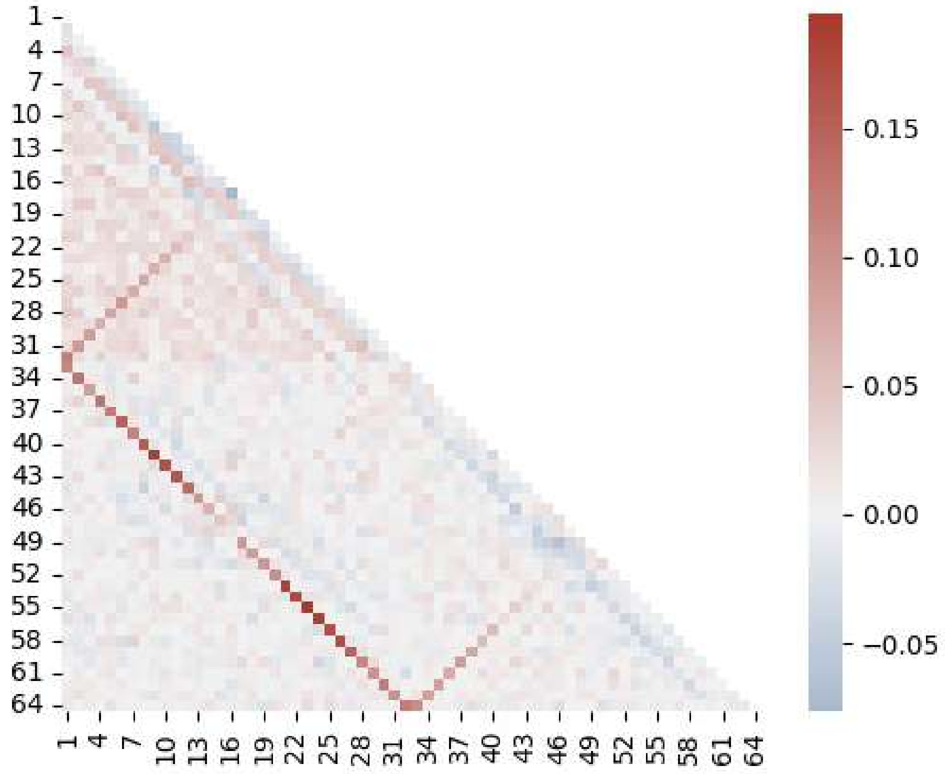


Figure 4.5: Heatmap of pairwise Spearman correlation coefficients of 64 features (2 features per amino acid) for the best combination with $r = 2$ properties found by the IBCGA algorithm for D1 (see Section 4.3.2). Here 1 denotes the value of the first property for the leftmost amino acid, . . . , 32 denotes the value of the first property for the rightmost amino acid, 33 denotes the value of the second property for the leftmost amino acid, . . . , 64 denotes the value of the second property for the rightmost amino acid.

We observe that all correlation values are $< 0.2$. This indicates that the features have insignificant correlation among them. We observe a mid-level correlation between the pair (1,33). This is intuitive because 1 and 33 both represent the leftmost amino acid in the *context window*; recall that in these experiments, our context window size is 32 (excluding the lysine residue). For the same reason, we are noticing a comparatively higher correlation between pairs (2,34), (3,35), . . . , (32,64).

### 4.3.4 Importance of the biophysico properties in a combination

We use *permutation feature importance* (see Section 2.3.2) to compute the importance of each feature. We use the *permutation_importance* class of sklearn library with $n\_repeats = 10$. Hence, we get 10 importance values for each of the feature. The distribution of these values along with the standard deviation is plotted in Figure 4.6 for the 64 features for D1 with $r = 2$ mentioned in Section 4.3.2.

We notice that the importance of the features which correspond to amino acids nearer to the

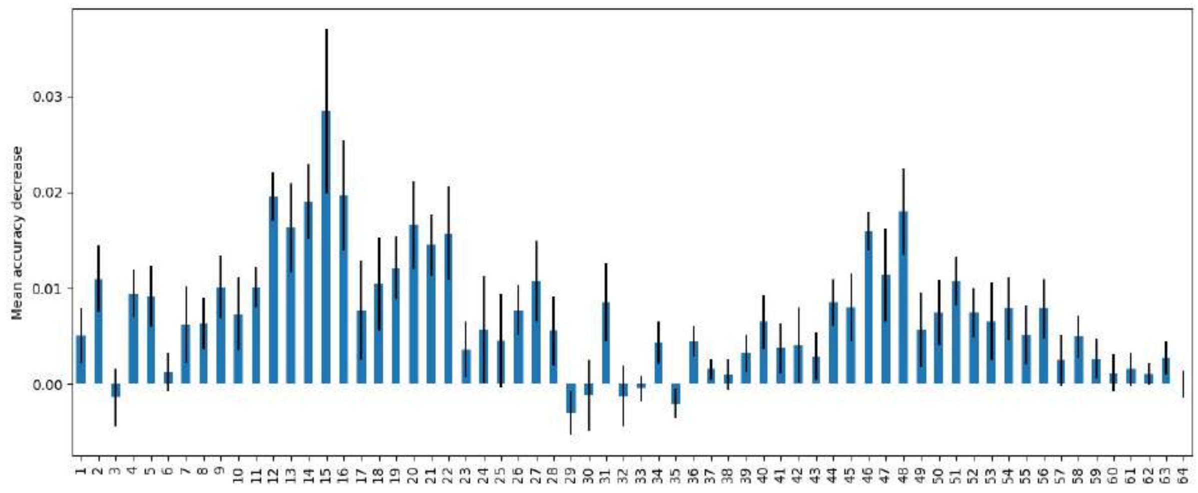


Figure 4.6: Distribution of importance values for D1 for each of the 64 features along with the standard deviation (vertical black bar).

lysine residue are comparatively higher than the features which correspond to amino acids that are far away. As a result, we observe a somewhat normal distribution of feature importance values for features 1 to 32 (which correspond to $1^{st}$ property) and for features 33 to 64 (which correspond to $2^{nd}$ property).

However, the amino acids that are far from the lysine residue can not be completely ignored. We can occassionally see some important features corresponding to amino acids that are farther from the lysine residue. For instance, $2^{nd}$ feature (14 amino acids away from the lysine residue) shows a relatively higher importance compared to its downstream 9 amino acids which are nearer to the lysine residue.

## 4.4 Conclusion

The biophysico properties have been proven useful for predicting different protein functions and structural properties. Several works have used these properties for predicting succinylation sites but the inter-dependency among the properties has not been considered. We propose a variant of genetic algorithm for finding suitable combination of biophysico properties for efficient prediction of succinylation sites. We show that the combinations found by our algorithm achieve better performance than the combination with all the good properties.

In the following chapter, we will explore different deep learning architectures for succinylated site prediction. We will try to combine the random forest classifiers trained on the suitable biophysico combinations with the deep learning architectures to improve the performance of our proposed model. Finally, we will perform an extensive comparative analysis with the existing works.

# Chapter 5

# Succinylation prediction

In the previous chapter, we have presented the methodology for selecting suitable combinations of biophysico properties for efficient prediction of succinylation sites. We have also illustrated the obtained results with the selected combinations and performed statistical tests. In this chapter, we describe the undertaken experiments with different deep learning architectures. Moreover, ensembling different deep learning models in combination with the models obtained using combination of biophysico properties will be discussed. We also present a differential evolution based algorithm (this algorithm is elaborately discussed in Section 2.1.2) for finding an appropriate threshold to optimize the ensemble classifiers.

## 5.1 Introduction

In Section 3.2, we have discussed the existing works that leveraged state-of-the-art techniques of deep learning to come up with efficient succinylation site predictors. Although the claimed performance seems satisfactory in most of these works, careful scrutiny reveals that proper procedure has not been followed by most of them particularly in connection with their comparative analysis with other existing works. More specifically, a few works have data leakage problem (details about this problem has been discussed in Section 2.4.5) and in some, there is no use of validation set for tuning hyperparameters. Hence, a sound methodology to develop model for succinylation site prediction with high efficacy is direly needed. Another crucial issue in this regard is the so called sensitivity-specificity (SN-SP) trade-off as discussed in Section 2.5.1. DeepSuccinylSite [27] showed a competitive SN at the cost of a relatively low SP. Later on, LSTMCNNsucc [57] achieved a better MCC by improving the SP significantly, however the SN was very low compared to other existing works. Very recently, DeepSucc [1] outperformed all the previous works by achieving a very high MCC with a moderately high SP, but the SN was comparatively lower. Therefore, it is of paramount importance to develop a model where the SN and SP can be controlled by the bioinformaticians.

Before going into the methodical details, an important discussion is in order. All deep-learning models require initialization of weights of different layers which in turn require generation of random numbers. Hence, it is evident that the results will vary for different runs even with the same architecture and same dataset. One way to avoid this is to set a seed so that each time the sequence of generation of random numbers remains the same. Another way is to run the code $n$ times and gather a population of performances. We adopt the second method and create several models from the same architecture.

## 5.2 Methods

### 5.2.1 One-hot encoding

The samples are first pre-processed according to the procedure mentioned in Section 4.2.2. Then, each amino acid is mapped to a unique value ranging from 0 to 19 (as there are 20 possible amino acids). After that, each integer value is converted into a 20 dimensional vector with all zeros except a 1 at that integer valued index. For example, if the lysine residue 'K' is mapped to the integer 11, the final 20 dimensional vector from the lysine residue will be $\{0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0\}$. Hence, if we use *context window* = 33, the dimension of input to the deep learning model will be $33 \times 20$.

### 5.2.2 Deep learning architecures

We initially experiment with two different architectures, where we leverage the power of CNN and Bi-LSTM architectures. Our two basic settings are **Bi-LSTM+CNN** (will be referred to as *BLC* henceforth for brevity) and **CNN+Bi-LSTM** (will be referred to as *CBL* henceforth for brevity). As the name indicates, the main difference between BLC and CBL models lies in the order the two constituent deep neural network architectures have been connected to each other. We then use the combination of these two architectures to build a better predictor (will be referred to as *CBL_BLC* henceforth for brevity). The reason for choosing CNN and Bi-LSTM as our fundamental models is not arbitrary. As protein sequence is a sequential data, both 1D-CNN and LSTM can extract features from the protein sequence. However, CNN's main function is to extract local features whereas LSTM can capture the long-range dependency in the sequence. LSTM and CNNs have been successfully used in conjunction in different fields. For example, Xia *et al.* used the combination of LSTM and CNN in [69] for human activity recognition. Wang *et al.* used a regional CNN-LSTM model in [70] for sentiment analysis. Zhang *et al.* experimented with CNN-LSTM, LSTM-CNN architectures in [1] for building succinylation site predictor. This motivates us to use these two celebrated models in combination.

We use the Keras version 2.6.0 to implement the architectures. The architecture of *CBL* is shown

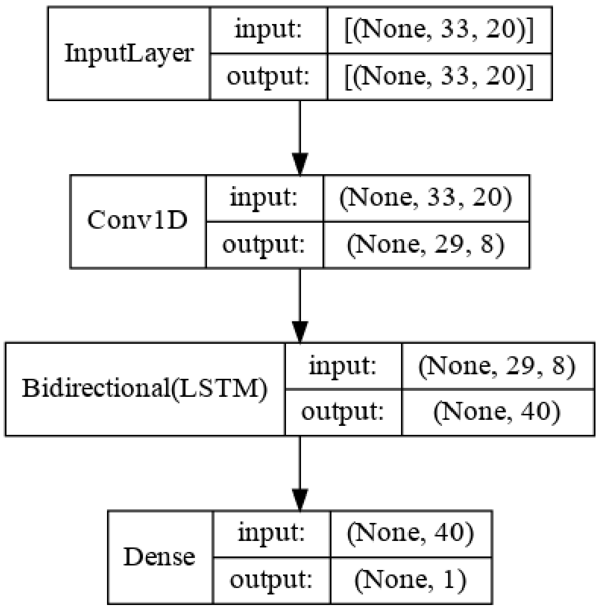


Figure 5.1: The one-hot encoded protein sequences are first fed into 1D convolution layer. The local features extracted from the CNN layer are then fed into the Bidirectional LSTM layer. The output of the bidirectional LSTM layer is fed into a fully connected dense layer with 1 neuron which will output the probability of the centered lysine residue of the input protein sequence to be succinylated.

in Figure 5.1. The *None* in each cell represents the *batch size*. The input layer takes as input the one-hot encoded protein sequences. 1D-CNN layer has been used with $kernel\ size = 5$, $strides = 1$ and $filters = 8$. Hence, $33 \times 20$ dimensional vector is converted into a 29-dimensional vector ($33 - 5 + 1 = 29$) for each of the filters resulting in a $29 \times 8$ dimensional vector. Then, the bidirectional LSTM layer is applied with $units = 20$ resulting in a 40-dimensional (20 for forward direction, 20 for backward direction) vector. Finally, a 1 neuron dense layer with sigmoid activation function is applied which outputs a value between 0 and 1 representing the probability of the lysine residue to be succinylated.

The architecture of *BLC* is shown in Figure 5.2. After the input layer, we use the bidirectional LSTM layer with *return_sequences* set to *True* so that each LSTM cell outputs the hidden states for each timestep. Therefore, the bidirectional LSTM layer outputs a $33 \times 40$ (there are 33 amino acids in a sample each of which represents a timestep for every LSTM unit) dimensional vector. Similar to the *CBL*, 1D-CNN layer outputs a $29 \times 8$ dimensional vector. As we will be feeding this vector into the feed forward network, we flatten the vector and feed it into the fully connected layer with 1 neuron as is done in *CBL*.

The architecture of *CBL_BLC* is shown in Figure 5.3. There are two branches in this architecture. One branch is similar to *CBL* except for the fully connected layer, another branch is similar to *BLC* except for the fully connected layer. The outputs of the two branches are concatenated and fed into a fully connected network with two layers. The final layer has 1 neuron similar to the final layer of *CBL* and *BLC*.

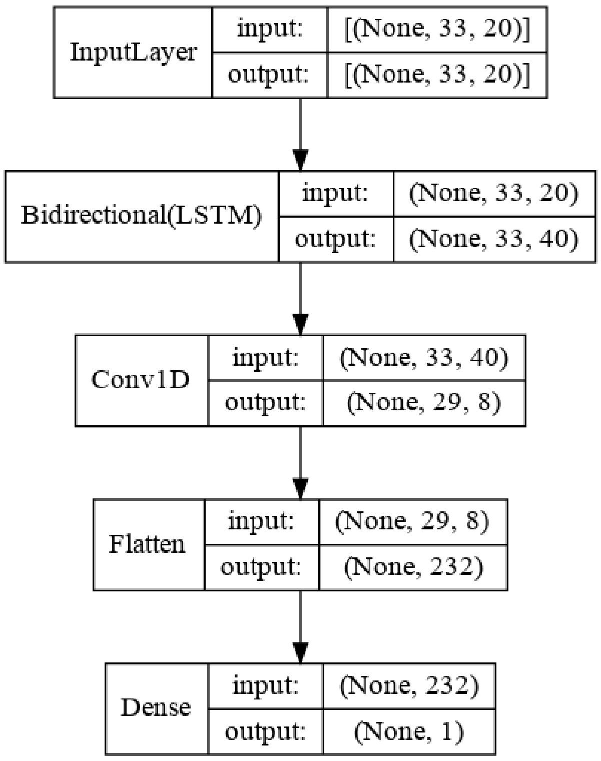


Figure 5.2: The one-hot encoded protein sequences are first fed into bidirectional LSTM layer. 1D-CNN layer follows this layer. The output of the CNN layer goes through a flatten layer. Finally, a fully connected dense layer outputs the probability in the same way as *CBL*.

### 5.2.3 Loss function

We need to choose a loss function (details about loss functions are discussed in Section 2.4.4) that is appropriate for binary classification as predicting succinylation is a binary classification problem. After experimenting with the three loss functions, namely binary cross-entropy, hinge loss and squared hinge loss, we choose binary cross-entropy as this gives better results than the other two.

During training the deep learning architectures with **D2** dataset, we use *weighted binary cross-entropy* because the number of negative samples is twice the number of positive samples. So, we penalize the model two times more for predicting a 1 as 0 than predicting a 0 as 1. The formula for weighted binary cross-entropy is as follows:

$$Loss = -\frac{1}{N}\sum_{i=1}^{N}(w_1 y_i \log p_i + w_0(1-y_i)\log(1-p_i)), \tag{5.1}$$

where, $(w_0, w_1) = (1, 2)$.

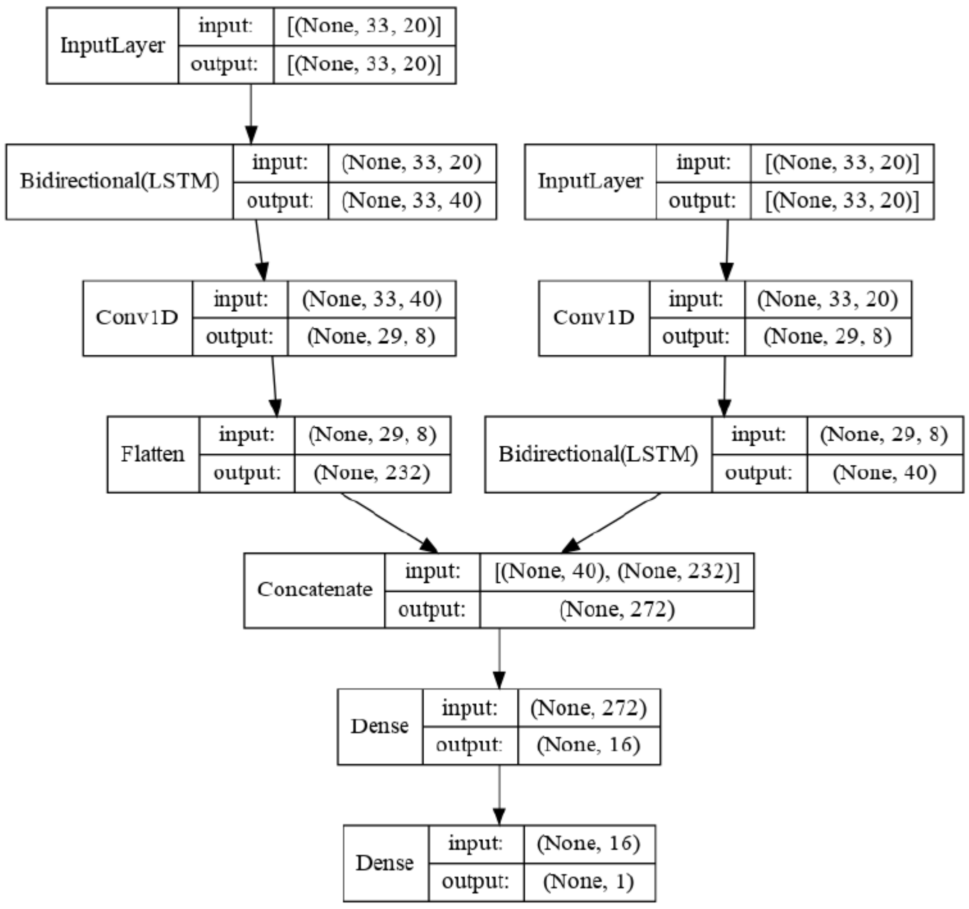


Figure 5.3: The *CBL_BLC* architecture uses two branches. One branch is similar to the architecture of *CBL* and another branch is similar to the architecture of *BLC*.

### 5.2.4 Optimizer

After the loss is calculated for a set of samples through a loss function, the weights of the neural network need to be updated so as to minimize that loss. There exist several optimizers in the literature. We adopt Adam optimizer [71] because it combines the advantages of AdaGrad [72] to deal with sparse gradients, and the ability of RMSProp (another unpublished but popular optimizer) to deal with non-stationary objectives. We try to find the appropriate learning rate using the technique mentioned in [73]. However, the results degraded following this procedure. Hence, we use the default learning rate of 0.001.

### 5.2.5 Checkpoint

We run our model with *batch size* set to 128 and for 80 epochs. We monitor the loss on the validation dataset after each epoch and save a checkpoint of the model if the calculated loss is smaller than the smallest loss found so far. After the training is complete, we use the model that has had the smallest validation loss across the 80 epochs.

### 5.2.6 Ensembling of different models

We train each of the three architectures for 5 times. This results in 5 different versions for each architecture. We calculate the average probability of a sample's belonging to class 1 according to the following equation:

$$Probability = \frac{1}{N}\sum_{i=1}^{N} p_i, \tag{5.2}$$

where, $p_i$ is the predicted probability from the $i^{th}$ classifier. If this probability is less than a pre-defined *threshold*, the ensemble classifier classifies the sample as 0, otherwise the sample is classified as 1. We initially set the *threshold* value to 0.5 to evaluate the performance of the ensemble classifiers.

### 5.2.7 Tuning the threshold for ensemble classifiers

As has been discussed in Section 5.2.6, we have a parameter called *threshold* which is initially set to 0. However, if we decrease (increase) the threshold value, the SN (SP) will increase compromising the SP (SN). Hence, this parameter can be tuned to produce 'better' results under different circumstances where it may be more desirable to achieve a better SP or SN for that matter.

We use differential evolution algorithm (this algorithm is elaborately discussed in Section 2.1.2) to find an optimal value for the threshold to improve the performance. Each individual in this algorithm refers to a threshold value. The fitness of an individual is the MCC value obtained by the ensemble classifier on the validation dataset.

## 5.3 Results and discussions

### 5.3.1 Performance of deep learning architectures

We train the architectures *CBL, BLC* and *CBL_BLC* with both the training datasets of D1 & D2. In case of D2, we use weighted binary cross-entropy loss (see Subsection 5.2.3). We perform the training of each of the architectures five times and report the average performance on the validation dataset in Table 5.1.

Table 5.1: Performance of CBL, BLC and CBL_BLC on the validation dataset of D1 and D2. Five independent runs are conducted for each architecture and dataset combination, and the average performance values are reported. The SN of CBL is the best for both datasets. For the other three metrics, CBL_BLC is the winner for both datasets.

| **Method** | **D1** | | | | **D2** | | | |
|---|---|---|---|---|---|---|---|---|
| | SN | SP | ACC | MCC | SN | SP | ACC | MCC |
| CBL | **0.752** | 0.62 | 0.686 | 0.375 | **0.703** | 0.736 | 0.725 | 0.421 |
| BLC | 0.698 | 0.679 | 0.688 | 0.377 | 0.6985 | 0.78 | 0.753 | 0.466 |
| CBL_BLC | 0.73 | **0.684** | **0.707** | **0.415** | 0.685 | **0.8** | **0.76** | **0.475** |

We observe that the CBL_BLC is the winner with respect to SP, ACC & MCC for both datasets. CBL achieved the highest SN for both datasets at the cost of the lowest SP.

### 5.3.2 Performance of ensemble classifiers

As has been already mentioned in Section 5.2.6, we have run each of the model for 5 times. We can ensemble these 5 models as we expect that even if some of the 5 models misclassify a sample, majority will correctly classify the sample. We have obtained 19 models from the combination of biophysico properties (see Section 4.3.2). We will refer to the collection of these models as *BP*. We denote the ensemble classifier of any architecture by appending an '-E'. Hence, CBL-E will represent the ensemble classifier of CBL and so on. If we ensemble classifiers from different architectures, we will denote the ensemble classifier as the addition of different architectures appended by an '-E'. For example, the ensemble of CBL and BLC classifiers will be denoted as (CBL+BLC)-E. The ensemble of all the deep learning and traditional ML based models will be denoted as **DeepBiophysico**. The performance of the ensemble classifiers are shown in Table 5.2.

CBL_BLC-E dominates CBL-E and BLC-E both for datasets D1 and D2 except with respect to SN. In case of (CBL+BLC)-E, (BLC+CBL_BLC)-E and (CBL+BLC+CBL_BLC)-E, the

Table 5.2: Performance of different ensemble classifiers on the validation dataset of D1 and D2. DeepBiophysico achieves best SN, ACC and MCC in D1 and (CBL+BLC)-E achieves best SN in D2. (BLC+CBL_BLC)-E achieves best SP in D1. CBL_BLC-E achieves best SP, ACC, MCC in D2.

| **Architecture** | **D1** | | | | **D2** | | | |
|---|---|---|---|---|---|---|---|---|
| | SN | SP | ACC | MCC | SN | SP | ACC | MCC |
| CBL-E | 0.764 | 0.624 | 0.695 | 0.392 | 0.702 | 0.744 | 0.73 | 0.429 |
| BLC-E | 0.717 | 0.685 | 0.701 | 0.402 | 0.712 | 0.791 | 0.765 | 0.49 |
| CBL_BLC-E | 0.744 | 0.691 | 0.718 | 0.436 | 0.6985 | **0.81** | **0.772** | **0.498** |
| BP-E | 0.745 | 0.64 | 0.695 | 0.391 | 0.721 | 0.66 | 0.681 | 0.361 |
| (CBL+BLC)-E | 0.759 | 0.657 | 0.708 | 0.418 | **0.724** | 0.775 | 0.758 | 0.483 |
| (BLC+CBL_BLC)-E | 0.7299 | **0.692** | 0.711 | 0.423 | 0.707 | 0.8 | 0.769 | 0.496 |
| (CBL+BLC+CBL_BLC)-E | 0.748 | 0.678 | 0.713 | 0.428 | 0.717 | 0.787 | 0.764 | 0.49 |
| DeepBiophysico | **0.772** | 0.681 | **0.726** | **0.454** | 0.723 | 0.766 | 0.752 | 0.472 |

performances are not improving much from the individual ensemble classifiers (i.e., CBL-E, BLC-E and CBL_BLC-E).

If we observe the performance of BP-E both on D1 & D2, we note that the SN ($0.745$ on validation set of D1, $0.705$ on validation set of D2) is competitive if compared to the SNs of CBL-E, BLC-E and CBL_BLC-E. However, BP-E is lagging way behind with respect to SP which makes the other metrics (ACC and MCC) poor too. However, we achieve the highest SN, ACC and MCC on the dataset D1 with DeepBiophysico which shows the usefulness of the suitable combinations of biophysico properties. However, DeepBiophysico is not the best with respect to any metric on the dataset D2 although the achieved SN is the second best and very close to the best one. Still, the comparable results of BP-E on dataset D2 shows the effectiveness of the chosen combinations. We choose the DeepBiophysico for dataset D1 and (CBL+BLC+CBL_BLC)-E for dataset D2 to be the best models. We also take DeepBiophysico for dataset D2 because it is the second best and the ensemble of more number of classifiers, hence the probability for an unseen sample to be misclassified is low. We try to tune the *threshold* parameter in the next subsection to improve the performance of these models.

### 5.3.3 Tuning threshold parameter

We use the differential evolution class from the scipy.optimize (scipy version 1.2.0) package to optimize the MCC value. We keep the lower bound of the *threshold* as 0.4 and upper bound as 0.6. The *tol* (tolerance) parameter is set to 1e-7. Other parameters are kept as the default ones. The algorithm is run 10 times and the threshold value that gives the best MCC value is recorded. The performance of the tuned models are reported in Table 5.3. For D1, as we choose DeepBiophysico, we observe that the SN increased from 0.772 to 0.795 with the tuned value. As a result, the SP decreased from 0.681 to 0.666. ACC increased from 0.726 to 0.731 and MCC increased from 0.454 to 0.465. For D2, as we choose (CBL+BLC+CBL_BLC)-E, we observe that the SN increased from 0.717 to 0.835 with the tuned value. As a result, the SP decreased from 0.787 to 0.699. ACC decreased from 0.764 to 0.745 and MCC increased from 0.49 to 0.505.

Table 5.3: Performance of tuned ensemble classifiers. For dataset D1, we use the model DeepBiophysico and for dataset D2, we use the model (CBL+BLC+CBL_BLC)-E. For each model, the obtained threshold is also shown which produces best MCC across the 10 runs.

| Dataset | Threshold | SN | SP | ACC | MCC |
|---|---|---|---|---|---|
| D1 | 0.491 | 0.795 | 0.666 | 0.731 | 0.465 |
| D2 | 0.402 | 0.835 | 0.699 | 0.745 | 0.505 |

### 5.3.4 Comparison with existing works

The two datasets D1 and D2 that we have used throughout the thesis, have been used by most of the existing works. Hence, we can directly utilize the reported result for a work during comparison if the comparison is being conducted on the same dataset. In the case when we have to compute the result of a predictor on a different dataset, we re-implement the predictor and train with the same training data that is being used to train our model. The training and testing dataset of D1 has been directly used by LSTMCNNsucc [57]. Although DeepSuccinylSite [27] used solely the training and testing dataset of D2, we re-implemented this predictor to re-calculate the performance of DeepSuccinylSite because there was an issue with their reported results (see Section 3.2). Hence, we are able to calculate its performance on D1 too. The results of these predictors along with the performance of both DeepBiophysico and tuned DeepBiophysico on the test dataset of D1 are shown in Table 5.4.

Table 5.4: Performance of some existing works on the testing dataset of D1 along with the performance of DeepBiophysico (with threshold=0.5 and tuned threshold)

| Method | SN | SP | ACC | MCC | Remarks |
|---|---|---|---|---|---|
| LSTMCNNSucc | 0.592 | **0.796** | **0.779** | **0.251** | Low SN |
| DeepSuccinylSite | 0.725 | 0.593 | 0.604 | 0.176 | Low SP |
| DeepBiophysico | 0.751 | 0.677 | 0.683 | 0.246 | High SN |
| Tuned DeepBiophysico | **0.769** | 0.624 | 0.639 | 0.24 | Highest SN,low SP |

We observe that both of our methods are performing better than DeepSuccinylSite. However, LSTMCNNSucc has a significantly better SP at the cost of a very poor SN. As the negative samples are the majority class in the test dataset, the ACC is higher for LSTMCNNSucc too. But, this method is performing poorly on the actual task which is to predict the positive sites as positive.

The tools Succinsite, SuccinSite2.0, GpSuc, Psucce, Inspector, DeepSuccinylSite and DeepSucc used the training and testing dataset of D2. The results of these predictors along with the performance of (CBL+BLC+CBL_BLC)-E and DeepBiophysico (with different thresholds) on the test dataset of D2 are shown in Table 5.5.

Table 5.5: Performance of several existing works on the testing dataset of D2 along with the performance of (CBL+BLC+CBL_BLC)-E and DeepBiophysico (with threshold=0.5 and tuned threshold)

| Method | Threshold | SN | SP | ACC | MCC | Remarks |
|---|---|---|---|---|---|---|
| Succinsite | - | 0.371 | 0.882 | 0.842 | 0.199 | Low SN |
| Succinsite2.0 | - | 0.457 | 0.884 | 0.85 | 0.263 | Low SN |
| GPsuc | - | 0.499 | 0.883 | 0.853 | 0.296 | Low SN |
| Psucce | - | 0.375 | 0.886 | 0.845 | 0.204 | Low SN |
| Inspector | - | 0.693 | 0.717 | 0.715 | 0.238 | Average SN Average SP |
| predML-Site | - | 0.094 | **0.918** | **0.854** | 0.013 | Extremely Low SN |
| DeepSuccinylSite | - | 0.7874 | 0.687 | 0.695 | 0.268 | Good SN Average SP |
| DeepSucc | - | 0.705 | 0.823 | 0.814 | **0.3437** | Unreliable |
| (CBL+BLC+CBL_BLC)-E | 0.5 | 0.697 | 0.753 | 0.748 | 0.269 | Average SN Good SP |
| Tuned (CBL+BLC+CBL_BLC)-E | 0.402 | **0.811** | 0.653 | 0.665 | 0.257 | High SN Low SP |
| **DeepBiophysico** | 0.5 | 0.748 | 0.73 | 0.731 | 0.278 | Good SN Good SP |
| Tuned **DeepBiophysico** | 0.481 | 0.776 | 0.702 | 0.708 | 0.272 | High SN Average SP |

We observe that the performance of (CBL+BLC+CBL_BLC)-E is very sensitive to the threshold value. (CBL+BLC+CBL_BLC)-E with a threshold of 0.5 achieves a moderate SN and SP and as a result, has a higher MCC value than the MCC value of all other methods except the MCC achieved by DeepSucc and GPSuc. Tuned (CBL+BLC+CBL_BLC)-E achieves the highest SN among the existing works at the cost of a low SP. DeepBiophysico achieves a very good SN and SP both with default and tuned threshold values, as a result it achieves a better MCC value. However, the performance of DeepSucc is better in all cases due to a very high SP at the cost of a moderate SN. We will present a more detaild discussion on this method in the following section.

### 5.3.5 A discussion on DeepSucc [1]

during the write-up stage of this thesis, the publication of a new predictor, called DeepSucc [1] came to our knowledge. We have thoroughly investigated the codebase shared by the paper and found some issues which are discussed below.

- They performed cross-validation to compare among different architectures. But in their code, we find that after each fold of cross-validation was performed, the authors mistakenly did not re-initialize the variables which has resulted in a data leakage. Hence, the claimed results are completely unreliable. And our repeated communications with the authors did not get any response.

- The code which was used for performing testing on the dataset D2 is completely absent in the repository. Although there is a folder named “Test” inside the repository, all the codes correspond to the cross-validation codes.
- We re-implemented their architectures but the produced results were far worse than the claimed ones.

For the above-mentioned reasons, the DeepSucc’s results seem unreliable at best. Therefore, although we have reported the results of DeepSucc (as reported in their paper), we actually exclude those from our comparative analysis and discussion.

## 5.4 Conclusion

In this chapter, we discussed about the application of several deep learning architectures and found that the deep learning models have been capable of dominating the traditional ML based models that are built upon hand crafted features. We ensembled the deep learning models with traiditional ML based classifiers to produce better results than the result of a single deep learning model. Additionally, we tuned the threshold of the ensemble classifier to control the sensitivity and specificity of the model and showed that we could achieve better results than the state-of-the-art by changing the value of the threshold. In the next chapter, we will discuss the further research scopes from where we left off.

# Chapter 6

# Conclusion

Lysine succinylation (Ksucc) is involved in diverse biological processes including cell cycle, growth & signal transduction pathways. It is significant because it is one of the largest (100 Da) PTMs. It also changes the net charge of the lysine residue from positive to negative. Aberrant succinylations have also been shown to cause diseases including metabolism disease, nervous system diseases and cancers. Therefore, it is crucial to identify succinylated sites to understand several functions of proteins and develop relevant drugs as remedy of diseases caused by succinylation.

Several experimental techniques (e.g., mass spectrometry, liquid chromatography) exist to detect PTM sites but all these techniques are costly, lengthy and cumbersome. Hence, computational models to efficiently detect succinylated sites will significantly reduce the labor of the biologists.

## 6.1 Our contributions

Please recall that, the aim of our thesis is to construct computational model to predict whether a lysine residue is succinylated or not. The objectives of our thesis are to search for suitable combinations of biophysico properties for efficient prediction of succinylation, assess the statistical significance of the features being used for prediction, leverage the power of deep learning to build strong predictors and incorporate deep learning with models trained on biophysico properties to construct stronger predictors.

In Chapter 4, we used the IBCGA algorithm and gathered 19 subsets of combinations from 29 biophysico properties that are capable of efficiently detecting succinylated sites. The MCC value achieved by each of these combinations ranges from 0.38 to 0.4 for D1 and 0.355 to 0.38 for D2 while the MCC value achieved by taking all the 29 properties is 0.37 for D1 and 0.35 for D2. It showed the importance of searching for suitable combinations of biophysico properties rather than using all of the top performing properties. We computed the pairwise Spearman correlation coefficient of the features for one of the 19 combinations on D1 and found that the correlation ranges from -0.1 to 0.2 which means that the features have very low correlation among themselves.

In Chapter 5, we experimented with CNN+Bi-LSTM (CBL) and Bi-LSTM+CNN (BLC) architectures. We also combined these two architectures and found that the combination achieves better SP, ACC and MCC for both D1 and D2 datasets compared to CBL and BLC architectures. Additionally, we performed ensembling of different classifiers and found that ensemble classifiers in general perform better than the single classifiers. For example, CBL achieved an SN of 0.752, SP of 0.62, ACC of 0.686 and MCC of 0.375, whereas ensemble of CBL classifier achieves an SN of 0.764, SP of 0.624, ACC of 0.695 and MCC of 0.392. Notably, the ensemble of the 19 models (these 19 models are collectively referred to as BP) trained on 19 combinations of biophysico properties (BP-E) achieved a competitive SN of 0.745 and SP of 0.64 on the validation dataset of D1. Although, the achieved SN (0.721) by BP-E is reasonable on the validation dataset of D2, the SP is relatively lower compared to the deep learning models. Predictably, the ensemble of all the deep learning models along with BP models achieved the best performance on the validation dataset of D1. However, only the ensemble of deep learning models achieved better performance than the ensemble of all the models on the validation dataset of D2 although the ensemble of all the models is not far behind. Moreover, we applied DE algorithm to tune the threshold of the ensemble classifiers. We were able to increase the MCC value from 0.454 to 0.465 for the best model on the validation dataset of D1 and from 0.49 to 0.505 for the best model on the validation dataset of D2.

Finally, we performed comparative analysis with existing works with the test dataset of D1 and D2. We observed that our model with and without tuned threshold achieves the best SN on the test dataset of both D1 and D2 compared to all other existing works, however, at the cost of a comparatively lower SP. But the novelty is that we provide a tunable parameter called threshold which the users can play with to control the sensitivity and specificity of the predictor.

We observed that the traditional ML based models trained with the chosen combinations of biophysico properties achieved a competitive SN even in comparison with the deep learning models. Ensembling of different deep learning models along with the traiditional ML models further improved the result. We also showed that the results of the state-of-the-art could be surpassed by varying the threshold value of the ensemble classifiers.

## 6.2 Further research

We now discuss several ways of extending the research works presented in this thesis as follows.

- We only used the biophysico properties as the biological features along with deep learning models. But, there exist numerous other biological features in the literature, including but not limited to, Position Specific Amino Acid Propensity, Amino acid composition, Auto-correlation functions, PSSM, Accessible Surface Area etc. These features can be incorporated with the proposed combinations of biophysico properties to see if that improves the performance.

- We did not apply self attention techniques in deep learning models although this is being widely used on sequential data. Attention can be included in the proposed deep learning models to inspect whether that improves the performance.
- We have only experimented with context window $= 33$. However, several other values have been used in the literature. Other values should be explored with our proposed methodology to see if any other value of context window produces improved performance.

# References

[1] D. Zhang and S. Wang, “A protein succinylation sites prediction method based on the hybrid architecture of LSTM network and CNN,” *Journal of Bioinformatics and Computational Biology*, p. 2250003, 2022.

[2] M. M. Hasan, M. S. Khatun, M. N. H. Mollah, C. Yong, and D. Guo, “A systematic identification of species-specific protein succinylation sites using joint element features information,” *International Journal of Nanomedicine*, vol. 12, p. 6303, 2017.

[3] M. Mann and O. N. Jensen, “Proteomic analysis of post-translational modifications,” *Nature Biotechnology*, vol. 21, no. 3, pp. 255–261, 2003.

[4] Y. Xu and K.-C. Chou, “Recent progress in predicting posttranslational modification sites in proteins,” *Current Topics in Medicinal Chemistry*, vol. 16, no. 6, pp. 591–603, 2016.

[5] G. A. Khoury, R. C. Baliban, and C. A. Floudas, “Proteome-wide post-translational modification statistics: frequency analysis and curation of the swiss-prot database,” *Scientific Reports*, vol. 1, no. 1, pp. 1–5, 2011.

[6] A. P. Oliveira and U. Sauer, “The importance of post-translational modifications in regulating *Saccharomyces cerevisiae* metabolism,” *FEMS Yeast Research*, vol. 12, no. 2, pp. 104–117, 2012.

[7] F. Tripodi, R. Nicastro, V. Reghellin, and P. Coccetti, “Post-translational modifications on yeast carbon metabolism: Regulatory mechanisms beyond transcriptional control,” *Biochimica et Biophysica Acta (BBA)-General Subjects*, vol. 1850, no. 4, pp. 620–627, 2015.

[8] P. Vlastaridis, A. Papakyriakou, A. Chaliotis, E. Stratikos, S. G. Oliver, and G. D. Amoutzias, “The pivotal role of protein phosphorylation in the control of yeast central metabolism,” *G3: Genes, Genomes, Genetics*, vol. 7, no. 4, pp. 1239–1249, 2017.

[9] P. Cohen, “The role of protein phosphorylation in human health and disease. The Sir Hans Krebs Medal Lecture.,” *European Journal of Biochemistry*, vol. 268, no. 19, pp. 5001–5010, 2001.

[10] S. Timmermann, H. Lehrmann, A. Polesskaya, and A. Harel-Bellan, "Histone acetylation and disease," *Cellular and Molecular Life Sciences CMLS*, vol. 58, no. 5, pp. 728–736, 2001.

[11] M. M. Hasan, S. Yang, Y. Zhou, and M. N. H. Mollah, "SuccinSite: A computational tool for the prediction of protein succinylation sites by exploiting the amino acid patterns and properties," *Molecular BioSystems*, vol. 12, no. 3, pp. 786–795, 2016.

[12] Z. Zhang, M. Tan, Z. Xie, L. Dai, Y. Chen, and Y. Zhao, "Identification of lysine succinylation as a new post-translational modification," *Nature Chemical Biology*, vol. 7, no. 1, pp. 58–63, 2011.

[13] A. Sreedhar, E. K. Wiese, and T. Hitosugi, "Enzymatic and Metabolic Regulation of Lysine Succinylation," *Genes & Diseases*, vol. 7, no. 2, pp. 166–171, 2020.

[14] M. Yang, Y. Wang, Y. Chen, Z. Cheng, J. Gu, J. Deng, L. Bi, C. Chen, R. Mo, X. Wang, *et al.*, "Succinylome Analysis Reveals the Involvement of Lysine Succinylation in Metabolism in Pathogenic *Mycobacterium tuberculosis* H37Rv," *Molecular & Cellular Proteomics*, vol. 14, no. 4, pp. 796–811, 2015.

[15] L. Sun, Z. Yao, Z. Guo, L. Zhang, Y. Wang, R. Mao, Y. Lin, Y. Fu, and X. Lin, "Comprehensive analysis of the lysine acetylome in *Aeromonas hydrophila* reveals cross-talk between lysine acetylation and succinylation in LuxS," *Emerging Microbes & Infections*, vol. 8, no. 1, pp. 1229–1239, 2019.

[16] M. J. Rardin, W. He, Y. Nishida, J. C. Newman, C. Carrico, S. R. Danielson, A. Guo, P. Gut, A. K. Sahu, B. Li, *et al.*, "SIRT5 Regulates the Mitochondrial Lysine Succinylome and Metabolic Networks," *Cell Metabolism*, vol. 18, no. 6, pp. 920–933, 2013.

[17] K. N. Papanicolaou, B. O'Rourke, and D. B. Foster, "Metabolism leaves its mark on the powerhouse: recent progress in post-translational modifications of lysine in mitochondria," *Frontiers in Physiology*, vol. 5, p. 301, 2014.

[18] G. Tannahill, A. Curtis, J. Adamik, E. Palsson-McDermott, A. McGettrick, G. Goel, C. Frezza, N. Bernard, B. Kelly, N. Foley, *et al.*, "Succinate is an inflammatory signal that induces IL-1$\beta$ through HIF-1$\alpha$," *Nature*, vol. 496, no. 7444, pp. 238–242, 2013.

[19] Y. Yang and G. E. Gibson, "Succinylation Links Metabolism to Protein Functions," *Neurochemical Research*, vol. 44, no. 10, pp. 2346–2359, 2019.

[20] G. E. Gibson, H. Xu, H.-L. Chen, W. Chen, T. T. Denton, and S. Zhang, "Alpha-ketoglutarate dehydrogenase complex-dependent succinylation of proteins in neurons and neuronal cell lines," *Journal of Neurochemistry*, vol. 134, no. 1, pp. 86–96, 2015.

[21] Y. Xiangyun, N. Xiaomin, *et al.*, “Desuccinylation of pyruvate kinase M2 by SIRT5 contributes to antioxidant response and tumor growth,” *Oncotarget*, vol. 8, no. 4, p. 6984, 2017.

[22] Y. Xu, Y.-X. Ding, J. Ding, Y.-H. Lei, L.-Y. Wu, and N.-Y. Deng, “iSuc-PseAAC: predicting lysine succinylation in proteins by incorporating peptide position-specific propensity,” *Scientific Reports*, vol. 5, no. 1, pp. 1–6, 2015.

[23] J. Jia, Z. Liu, X. Xiao, B. Liu, and K.-C. Chou, “iSuc-PseOpt: Identifying lysine succinylation sites in proteins by incorporating sequence-coupling effects into pseudo components and optimizing imbalanced training dataset,” *Analytical Biochemistry*, vol. 497, pp. 48–56, 2016.

[24] J. Jia, Z. Liu, X. Xiao, B. Liu, and K.-C. Chou, “pSuc-Lys: Predict lysine succinylation sites in proteins with PseAAC and ensemble random forest approach,” *Journal of Theoretical Biology*, vol. 394, pp. 223–230, 2016.

[25] M. M. Hasan and H. Kurata, “GPSuc: Global Prediction of Generic and Species-specific Succinylation Sites by aggregating multiple sequence features,” *PloS One*, vol. 13, no. 10, p. e0200283, 2018.

[26] Q. Ning, X. Zhao, L. Bao, Z. Ma, and X. Zhao, “Detecting succinylation sites from protein sequences using ensemble support vector machine,” *BMC Bioinformatics*, vol. 19, no. 1, pp. 1–9, 2018.

[27] N. Thapa, M. Chaudhari, S. McManus, K. Roy, R. H. Newman, H. Saigo, and D. B. Kc, “DeepSuccinylSite: a deep learning based approach for protein succinylation site prediction,” *BMC Bioinformatics*, vol. 21, no. 3, pp. 1–10, 2020.

[28] S. Kawashima and M. Kanehisa, “AAindex: Amino Acid Index Database,” *Nucleic Acids Research*, vol. 28, no. 1, pp. 374–374, 2000.

[29] K. Sörensen and F. Glover, “Metaheuristics,” *Encyclopedia of Operations Research and Management Science*, vol. 62, pp. 960–970, 2013.

[30] B. L. Miller, D. E. Goldberg, *et al.*, “Genetic algorithms, tournament selection, and the effects of noise,” *Complex Systems*, vol. 9, no. 3, pp. 193–212, 1995.

[31] R. Storn, “On the Usage of Differential Evolution for Function Optimization,” in *Proceedings of North American Fuzzy Information Processing*, pp. 519–523, IEEE, 1996.

[32] R. Storn and K. Price, “Differential Evolution–A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[33] Q. Wu, "On the optimality of orthogonal experimental design," *Acta Mathematicae Applacatae Sinica*, vol. 1, no. 4, pp. 283–299, 1978.

[34] C.-W. Tung and S.-Y. Ho, "POPI: predicting immunogenicity of MHC class I binding peptides by mining informative physicochemical properties," *Bioinformatics*, vol. 23, no. 8, pp. 942–949, 2007.

[35] S.-Y. Ho, L.-S. Shu, and J.-H. Chen, "Intelligent Evolutionary Algorithms for Large Parameter Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 6, pp. 522–541, 2004.

[36] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[37] A. Fisher, C. Rudin, and F. Dominici, "All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously.," *J. Mach. Learn. Res.*, vol. 20, no. 177, pp. 1–81, 2019.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[39] Z. Liu, Y. Wang, T. Gao, Z. Pan, H. Cheng, Q. Yang, Z. Cheng, A. Guo, J. Ren, and Y. Xue, "CPLM: a database of protein lysine modifications," *Nucleic Acids Research*, vol. 42, no. D1, pp. D531–D536, 2014.

[40] H.-D. Xu, S.-P. Shi, P.-P. Wen, and J.-D. Qiu, "SuccFind: a novel succinylation sites online prediction tool via enhanced characteristic strategy," *Bioinformatics*, vol. 31, no. 23, pp. 3748–3750, 2015.

[41] X. Zhao, Q. Ning, H. Chai, and Z. Ma, "Accurate in silico identification of protein succinylation sites using an iterative semi-supervised learning technique," *Journal of Theoretical Biology*, vol. 374, pp. 60–65, 2015.

[42] K.-C. Chou, "Prediction of Protein Cellular Attributes Using Pseudo-Amino Acid Composition," *Proteins: Structure, Function, and Bioinformatics*, vol. 43, no. 3, pp. 246–255, 2001.

[43] K.-C. Chou, "Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes," *Bioinformatics*, vol. 21, no. 1, pp. 10–19, 2005.

[44] C. K. Z. CT, "Review: Prediction of protein structural classes," *Critical Reviews in Biochemistry and Molecular Biology*, vol. 30, p. 275349, 1995.

[45] Y. López, A. Dehzangi, S. P. Lal, G. Taherzadeh, J. Michaelson, A. Sattar, T. Tsunoda, and A. Sharma, "SucStruct: Prediction of succinylated lysine residues by using structural properties of amino acids," *Analytical Biochemistry*, vol. 527, pp. 24–32, 2017.

[46] A. Dehzangi, Y. López, S. P. Lal, G. Taherzadeh, J. Michaelson, A. Sattar, T. Tsunoda, and A. Sharma, "PSSM-Suc: Accurately predicting succinylation using position specific scoring matrix into bigram for feature extraction," *Journal of Theoretical Biology*, vol. 425, pp. 97–102, 2017.

[47] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, no. 17, pp. 3389–3402, 1997.

[48] Y. López, A. Sharma, A. Dehzangi, S. P. Lal, G. Taherzadeh, A. Sattar, and T. Tsunoda, "Success: evolutionary and structural properties of amino acids prove effective for succinylation site prediction," *BMC Genomics*, vol. 19, no. 1, pp. 105–114, 2018.

[49] A. Dehzangi, Y. López, S. P. Lal, G. Taherzadeh, A. Sattar, T. Tsunoda, and A. Sharma, "Improving succinylation prediction accuracy by incorporating the secondary structure via helix, strand and coil, and evolutionary information from profile bigrams," *PloS One*, vol. 13, no. 2, p. e0191900, 2018.

[50] Y. Zhu, C. Jia, F. Li, and J. Song, "Inspector: a lysine succinylation predictor based on edited nearest-neighbor undersampling and adaptive synthetic oversampling," *Analytical Biochemistry*, vol. 593, p. 113592, 2020.

[51] S. Ahmed, A. Rahman, M. A. M. Hasan, J. Rahman, M. K. B. Islam, and S. Ahmad, "predML-Site: Predicting Multiple Lysine PTM Sites with Optimal Feature Representation and Data Imbalance Minimization," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[52] Y. Xu, J. Ding, L.-Y. Wu, and K.-C. Chou, "iSNO-PseAAC: Predict Cysteine S-Nitrosylation Sites in Proteins by Incorporating Position Specific Amino Acid Propensity into Pseudo Amino Acid Composition," *PloS One*, vol. 8, no. 2, p. e55844, 2013.

[53] Y.-R. Tang, Y.-Z. Chen, C. A. Canchaya, and Z. Zhang, "GANNPhos: a new phosphorylation site predictor based on a genetic algorithm integrated neural network," *Protein Engineering, Design & Selection*, vol. 20, no. 8, pp. 405–412, 2007.

[54] Z. Liu, J. Cao, X. Gao, Y. Zhou, L. Wen, X. Yang, X. Yao, J. Ren, and Y. Xue, "CPLA 1.0: an integrated database of protein lysine acetylation," *Nucleic Acids Research*, vol. 39, no. suppl_1, pp. D1029–D1034, 2011.

[55] Z. Chen, X. Liu, F. Li, C. Li, T. Marquez-Lago, A. Leier, T. Akutsu, G. I. Webb, D. Xu, A. I. Smith, *et al.*, "Large-scale comparative assessment of computational predictors for lysine post-translational modification sites," *Briefings in Bioinformatics*, vol. 20, no. 6, pp. 2267–2290, 2019.

[56] W. Ning, H. Xu, P. Jiang, H. Cheng, W. Deng, Y. Guo, and Y. Xue, “HybridSucc: A Hybrid-learning Architecture for General and Species-specific Succinylation Site Prediction,” *Genomics, Proteomics & Bioinformatics*, vol. 18, no. 2, pp. 194–207, 2020.

[57] G. Huang, Q. Shen, G. Zhang, P. Wang, and Z.-G. Yu, “LSTMCNNsucc: A Bidirectional LSTM and CNN-Based Deep Learning Method for Predicting Lysine Succinylation Sites,” *BioMed Research International*, vol. 2021, 2021.

[58] P. V. Hornbeck, J. M. Kornhauser, V. Latham, B. Murray, V. Nandhikonda, A. Nord, E. Skrzypek, T. Wheeler, B. Zhang, and F. Gnad, “15 years of PhosphoSitePlus®: integrating post-translationally modified sites, disease variants and isoforms,” *Nucleic Acids Research*, vol. 47, no. D1, pp. D433–D441, 2019.

[59] H. Xu, J. Zhou, S. Lin, W. Deng, Y. Zhang, and Y. Xue, “PLMD: An updated data resource of protein lysine modifications,” *Journal of Genetics and Genomics*, vol. 44, no. 5, pp. 243–250, 2017.

[60] K.-Y. Huang, T.-Y. Lee, H.-J. Kao, C.-T. Ma, C.-C. Lee, T.-H. Lin, W.-C. Chang, and H.-D. Huang, “dbPTM in 2019: exploring disease association and cross-talk of post-translational modifications,” *Nucleic Acids Research*, vol. 47, no. D1, pp. D298–D308, 2019.

[61] K. Nakai, A. Kidera, and M. Kanehisa, “Cluster analysis of amino acid indices for prediction of protein structure and function,” *Protein Engineering, Design and Selection*, vol. 2, no. 2, pp. 93–100, 1988.

[62] P. Han, X. Zhang, and Z.-P. Feng, “Predicting disordered regions in proteins using the profiles of amino acid indices,” *Bmc Bioinformatics*, vol. 10, no. 1, pp. 1–8, 2009.

[63] B. Liu, X. Wang, Q. Chen, Q. Dong, and X. Lan, “Using Amino Acid Physicochemical Distance Transformation for Fast Protein Remote Homology Detection,” 2012.

[64] C.-R. Chung, Y.-P. Chang, Y.-L. Hsu, S. Chen, L.-C. Wu, J.-T. Horng, and T.-Y. Lee, “Incorporating hybrid models into lysine malonylation sites prediction on mammalian and plant proteins,” *Scientific Reports*, vol. 10, no. 1, pp. 1–12, 2020.

[65] Y. Xu, J. Ding, and L.-Y. Wu, “iSulf-Cys: Prediction of S-sulfenylation Sites in Proteins with Physicochemical Properties of Amino Acids,” *PLoS One*, vol. 11, no. 4, p. e0154237, 2016.

[66] S.-Y. Ho, J.-H. Chen, and M.-H. Huang, “Inheritable Genetic Algorithm for Biobjective 0/1 Combinatorial Optimization Problems and its Applications,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 609–620, 2004.

[67] J. Janin, S. Wodak, M. Levitt, and B. Maigret, "Conformation of Amino Acid Side-chains in Proteins," *Journal of Molecular Biology*, vol. 125, no. 3, pp. 357–386, 1978.

[68] P. Klein, M. Kanehisa, and C. DeLisi, "Prediction of protein function from sequence properties: Discriminant analysis of a data base," *Biochimica et Biophysica Acta (BBA)-Protein Structure and Molecular Enzymology*, vol. 787, no. 3, pp. 221–226, 1984.

[69] K. Xia, J. Huang, and H. Wang, "LSTM-CNN Architecture for Human Activity Recognition," *IEEE Access*, vol. 8, pp. 56855–56866, 2020.

[70] J. Wang, L.-C. Yu, K. R. Lai, and X. Zhang, "Dimensional sentiment analysis using a regional CNN-LSTM model," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 225–230, 2016.

[71] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ArXiv Preprint ArXiv:1412.6980*, 2014.

[72] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization.," *Journal of Machine Learning Research*, vol. 12, no. 7, 2011.

[73] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 464–472, IEEE, 2017.