Features and Algorithms for Embedded Protein Sequence Classification with Class Imbalance

doi: https://doi.org/10.3389/fgene.2022.885929

Dissertation

Features and Algorithms for Embedded Protein Sequence Classification with Class Imbalance

埋め込み表現されたタンパク質配列の不均衡な分類のための特徴およびアルゴリズム

Graduate School of

Natural Science & Technology

Kanazawa University

Division of Electrical Engineering and Computer Science

Student ID No. 1924042009

Name: Fatma Indriani

Chief Advisor: Professor Kenji Satou

Date of Submission: June, 2022

Abstract

With machine learning, we can learn hidden information from large-scale biological data to be applied in various prediction or identification tasks. Many of these applications involve the study of protein sequence data. We identify two major challenges in dealing with protein sequence data in the bioinformatics domain. The first is how to represent the protein data so that it is suitable for machine learning tasks. A recent approach is the protein sequence embedding methods borrowed from natural language processing (NLP) research. The second challenge is class imbalance, which is a problem because standard classification algorithms are not designed to handle imbalanced data.

Previous studies of protein embedding methods for protein classification are still limited, especially in tasks that have class imbalance. In this study, we investigated the best approach in using protein embedding methods for imbalanced class protein classification tasks. Two different protein classification tasks are investigated for this aim: (1) Protein-protein Interaction: Identification of Human-Virus PPI, and (2) PTM classification: Lysine glutarylation prediction.

For the first task, it is found that modification to the feature formulation improved the classification performance. Another process that improved performance is applying random over-sampling. However, combining them both at once improved the classification evaluation further from originally 0.9414 to 0.9448.

For the second task, we improved lysine glutarylation prediction by combining embedding features (ProtT5-XL-UniRef50) with non-embedding features (enhanced

amino acid composition encoding and distribution encoding) and applying random under-sampling. The performance evaluations obtained from this model for recall, specificity, and AUC are 0.7864, 0.6286, and 0.7075, respectively. Compared to other models using the same dataset, this model outperformed the existing model in terms of recall and AUC score and could potentially be used to complement previous models to reveal new glutarylated sites.

Keywords: protein classification, protein embedding, class imbalance, doc2vec, BERT, protein-protein interaction, post translational modification

Acknowledgments

I am very grateful to my supervisor, Professor Kenji Satou, and other professors in the Bioinformatics Laboratory at Kanazawa University for giving me the opportunity and for providing constant support and guidance during the course of this study.

Next I would like to thank my husband, Ely Fahrudin, and my daughter, Alisha Mumtaz Fahrudin for their encouragement, understanding, and general support in finishing this research.

I am also extremely indebted to my parents who has always supported my study since I am still a kid, and my siblings for their love, prayers, and support.

Special thanks goes to my lab colleagues Kunti, Mera, and Bedy for the discussions and encouragement, especially during the early years of my study.

I would also like to thank and acknowledge the Directorate General of Higher Education, Research, and Technology; Ministry of Education, Culture, Research, and Technology of The Republic of Indonesia for providing the BPP-LN scholarship during the duration of this research.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Background

In the past, before the development of machine learning, bioinformatics algorithms had to be programmed individually. With machine learning, we can learn hidden information from large scale biological data, to be applied in various prediction or identification task. For example, it has led advances in personalized medicine, adapting treatment based on personal health record and personal genes. Another area is pharmaceutical research, designing new drugs to combat infectious diseases, and in developing new vaccines. Another application of machine learning is genetics, particularly in fighting illnesses caused by individual genes that have been affected or inherited. In addition, comparing genomic data adds to our understanding of Earth's evolutionary history. Another area of application is in agriculture: the study of proteomics and genetics helps better crops to be developed that are more resistant to drought and to pests. Many of these applications involved with the study of protein sequence data.

We identify two major challenges in dealing with protein sequence data in the bioinformatics domain. The first is how to represent the protein data so that it is suitable for machine learning tasks. Various methods have been proposed in this area and this is a highly active research area. A recent approach is the protein sequence embedding methods, borrowed from natural language processing (NLP) research, in which a protein model is learned from large scale protein data, similar to how a language model is learned from large text corpora [1]. Later this trained protein model can be used to extract features to represent different protein sequences intrinsically, even those sequences that have not been seen in the training process. Previous studies have investigated the use of pre-trained

language models to show its effectiveness as protein sequence representation for protein classification, such as ([2], [3]). However, it is essential to investigate this approach further, especially in conjunction with different preprocessing methods and different classifiers.

The second challenge is the class imbalance problem. Many data in the bioinformatics domain are naturally imbalanced in real world scenario. For example in the protein classification problems, certain types of proteins is far more common than other types of protein; a few protein-protein pairs may have interactions, but many other pairs revealed no interaction; post-translational modifications happen rarely compared to no modifications, etc. In these cases, one of the class has significantly more samples than the other class. This issue is a problem because standard classification algorithms are not designed to handle imbalanced data. The result may be biased towards the majority class.

1.2 Objective

Previous studies of protein embedding methods for protein classification is still limited, especially in tasks that have class imbalance. The aim of this study is find the best approach in using protein embedding methods for imbalanced class protein classification tasks. Two different protein classification tasks are investigated for this objective:

1. Protein-protein Interaction: Identification of Human-Virus PPI
2. PTM classification: Lysine glutarylation prediction

1.3 Contribution

Below is a description of the study's contributions.

1. Improving human-virus PPI identification using Doc2Vec encoding by modifying the feature formulation method and applying random oversampling
2. Improving lysine glutarylation prediction by combining embedding features (ProtT5-

XL-UniRef50) with non-embedding features (enhanced amino acid composition encoding and distribution encoding) and applying random undersampling.

1.4 Thesis Organization

This thesis consists of five chapters.

Chapter 1 describes the research's background and rationale. Additionally, this chapter explained the objectives and contribution of the research.

Chapter 2 presents related work regarding various encoding methods to represent protein sequence data. We divide it into two major approach, traditional methods (non-embedded) and embedding methods. We also highlight some background knowledge about class imbalance methods used in this research. Finally we describe the workflow to build the best classification model and the evaluating its performance.

Chapter 3 presents the study to improve human-virus PPI identification using doc2Vec encoding. We investigated 6 different feature formulation methods, 4 different feature transformations, 6 different data resampling, and 7 different classification algorithms. It is revealed a new feature formulation method combined with random oversampling improved the result.

Chapter 4 presents the study to find the best model for lysine glutarylation prediction. We investigated combinations of 7 non-embedding features, 6 embedding features, 5 classification algorithms. It is revealed that best features are a combination of embedding features (ProtT5-XL-UniRef50) with non-embedding features (enhanced amino acid composition encoding and distribution encoding) with XGBoost classifier, with random undersampling to balance the data.

Chapter 5 summarizes the study findings and suggests directions for further research in this topic.

Chapter 2 Literature Review

In this chapter, we presented well-known and state-of-the-art protein features methods, divided into two types: non-embedding methods and embedding methods. In the second section, we explained resampling methods commonly used to handle class imbalance. Finally, we presented a common framework to build a classification model and evaluating its performance.

2.1 Protein Sequence Features Extraction Methods

Protein sequence has variable length, commonly between 50 and 2000. Traditional machine learning algorithms need fixed length input and cannot handle sequence directly. Many algorithms also prefer numerical input. Various methods has been proposed to extract features from protein sequence so that it becomes suitable for machine learning application. In this chapter, we divide the methods utilized in this study into two approaches: (1) traditional (non-embedding based) methods, and (2) embedding based methods.

2.1.1 Non-embedding based Features Extraction Methods

1. Amino acid composition (AAC) and Enhanced Amino Acid Composition (EAAC)

The AAC method encodes a protein sequence-based on the frequency of each amino acid [4]. For this type of feature, we used two variants. The first variant is the basic AAC, in which the protein sequence is converted into a vector of length 20, representing the frequency of the 20 amino acids ("*ACDEFGHIKLMNPQRSTVWY*"). Each element is calculated according to Equation 1, as follows:

$$\mathrm{f}(t) = \frac{N(t)}{N} \tag{1}$$

where *t* is the amino acid type, *N(t)* is the total number of amino acids *t* appearing in the sequence, and *N* is the length of the sequence.

The second variant is EAAC, introduced by [5]. In this encoding, the EAAC was calculated using sliding windows, that is, from a fixed window size, moving from left to right. To calculate the frequency of each amino acid in each window, see Equation 2:

$$\mathrm{f}(t, win) = \frac{N(t,win)}{N(win)} \quad (2)$$

where *N(t,win)* represents the number of amino acids *t* that appear in the window *win* and *N(win)* represents the length of the window. To develop our model, a default window size of five was used.

2. Composition/Transition/Distribution (CTD)

The CTD method encodes a protein sequence-based on various structural and physicochemical properties [6], [7]. Thirteen properties were used to build the features. Each property was divided into three groups (see Table 1). For example, the attribute "Hydrophobicity_PRAM900101" divides the amino acids into polar, neutral, and hydrophobic groups.

Table 1. Physicochemical attributes and its division of the amino acids

Attribute	Division		
Hydrophobicity_PRAM900101	Polar: RKEDQN	Neutral: GASTPHY	Hydrophobicity: CLVIMFW
Hydrophobicity_ARGP820101	Polar: QSTNGDE	Neutral: RAHCKMV	Hydrophobicity: LYPFIW
Hydrophobicity_ZIMJ680101	Polar: QNGSWTDERA	Neutral: HMCKV	Hydrophobicity: LPFYI

Hydrophobicity_PONP930101	Polar: KPDESNQT	Neutral: GRHA	Hydrophobicity: YMFWLCVI
Hydrophobicity_CASG920101	Polar: KDEQPSRNTG	Neutral: AHYMLV	Hydrophobicity: FIWC
Hydrophobicity_ENGD860101	Polar: RDKENQHYP	Neutral :SGTAW	Hydrophobicity: CVLIMF
Hydrophobicity_FASG890101	Polar: KERSQD	Neutral: NTPG	Hydrophobicity: AYHWVMFLIC
Normalized van der Waals volume	Volume range: 0-2.78 GASTPD	Volume range: 2.95-94.0 NVEQIL	Volume range: 4.03-8.08 MHKFRYW
Polarity	Polarity value: 4.9-6.2 LIFWCMVY	Polarity value: 8.0-9.2 PATGS	Polarity value: 10.4-13.0 HQRKNED
Polarizability	Polarizability value: 0-1.08 GASDT	Polarizability value: 0.128-120.186 GPNVEQIL	Polarizability value: 0.219-0.409 KMHFRYW
Charge	Positive: KR	Neutral: ANCQGHILMF PSTWYV	Negative: DE
Secondary structure	Helix: EALMQKRH	Strand: VIYCWFT	Coil: GNPSD
Solvent accessibility	Buried: ALFCGIVW	Exposed:	Intermediate: MPSTHY

The CTD feature comprises three parts: composition (CTDC), transition (CTDT), and distribution (CTDD). For composition, an attribute contributes to three values, representing the global distribution (frequency) of the amino acids in each of the three groups of attributes. The composition is computed as follows:

$$\mathrm{C}(r) = \frac{N(r)}{N} \tag{3}$$

where *N(r)* is the number of occurrences of type *r* amino acids in the sequence and *N* is the length of the sequence.

For transition, an attribute also contributes to three values, each representing the number of transitions between any pair of groups. The transition is calculated as follows:

$$\mathrm{T}(r,s) = \frac{N(r,s)+N(s,r)}{N-1} \tag{4}$$

where *N(r,s)* represents the number of occurrences amino acid type *r* transit to type *s* (i.e. it appeared as "rs" in the sequence), and *N* is the length of the sequence. Similarly, *N(s,r)* is the reverse, that is, the number of "sr" occurrences in the sequence.

The distribution feature consists of five values per attribute group, each of which corresponds to the fraction of the sequence length at five different positions in the group: first occurrence, 25%, 50%, 75%, and 100%.

3. Pseudo amino acid composition

Pseudo amino acid composition feature was proposed by [8]. For protein sequence P with L amino acid residues $P = [R_1R_2R_3\ldots R_L]$, the PAAC features can be formulated as

$$P= [p_1, p_2, \ldots, p_{20}, p_{20+1}, \ldots, p_{20+\lambda}]^T , (\lambda< L) \tag{5}$$

where

$$p_u = \begin{cases} \frac{f_u}{\sum_{i=1}^{20} f_i + w\sum_{k=1}^{\lambda} \tau_k}, (1 \le u \le 20) \\ \frac{w\tau_{u-20}}{\sum_{i=1}^{20} f_i + w\sum_{k=1}^{\lambda} \tau_k}, (20+1 \le u \le 20+\lambda) \end{cases} \quad (6)$$

w is the weight factor and τ_k is the k-the tier correlation factor, defined as

$$\tau_k = \frac{1}{L-k}\sum_{i=1}^{L-k} J_{i,i+k}, (k < L) \quad (7)$$

and

$$J_{i,i+k} = \frac{1}{\Gamma}\sum_{q=1}^{\Gamma}\left[\Phi_q R_{i+k} - \Phi_q R_i\right]^2 \quad (8)$$

where $\Phi_q(R_i)$ is the q-th function of the amino acid R_i, and Γ the total number of functions. In here Γ=3 and the functions used are hydrophobicity value, hydrophilicity value, and side chain mass of amino acid R_i.

A variant of PAAC called amphiphilic pseudo amino acid composition (APAAC) proposed in [9]. A protein sample P with L amino acid residues P = [$R_1R_2R_3\ldots R_L$], is formulated as

$$\mathrm{P}= [p_1, p_2, \ldots, p_{20}, p_{20+1}, \ldots, p_{20+\lambda}, p_{20+\lambda, \ldots}, p_{2\lambda}]^{\mathrm{T}}, (\lambda < \mathrm{L}) \quad (9)$$

where

$$p_u = \begin{cases} \frac{f_u}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda} \tau_j}, (1 \le u \le 20) \\ \frac{w\tau_{u-20}}{\sum_{i=1}^{20} f_i + w\sum_{j=1}^{2\lambda} \tau_j}, (20+1 \le u \le 20+2\lambda) \end{cases} \quad (10)$$

τ_j is the j-tier sequence-correlation factor calculated using the equations:

$$\begin{cases} \tau_1 = \frac{1}{L-1}\sum_{i=1}^{L-1} H_{i,i+1}^1 \\ \tau_2 = \frac{1}{L-1}\sum_{i=1}^{L-1} H_{i,i+1}^2 \\ \tau_3 = \frac{1}{L-2}\sum_{i=1}^{L-2} H_{i,i+2}^1 \\ \tau_4 = \frac{1}{L-2}\sum_{i=1}^{L-2} H_{i,i+2}^2, \lambda < L \\ \cdots \\ \tau_{2\lambda-1} = \frac{1}{L-1}\sum_{i=1}^{L-\lambda} H_{i,i+\lambda}^1 \\ \tau_{2\lambda} = \frac{1}{L-1}\sum_{i=1}^{L-\lambda} H_{i,i+\lambda}^1 \end{cases} \quad (11)$$

where $H_{i,j}^1$ and $H_{i,j}^2$ are hydrophobicity and hydrophilicity values of the i-th amino acid,

described by the following equation:

$$H_{i,j}{}^{1} = h^{1}(R_i) \cdot h^{1}(R_j)$$

$$H_{i,j}{}^{2} = h^{2}(R_i) \cdot h^{2}(R_j) \quad (12)$$

2.1.2 Embedding-based Features Extraction Methods

1. Doc2Vec

Doc2vec is a method to represent textual documents as numerical vectors proposed by Mikolov and Le [10]. It is based from another method called Word2Vec [11]. Training the doc2vec model required a set of documents. For each word appearing in the document, a word vector *W* is generated. For each document, a document vector *D* is generated. Weights for a softmax hidden layer is also trained in the model. When the trained model is presented with a new document, these weights are used to calculate the document vector.

The doc2vec model, originally proposed for textual documents, has been adapted for the biological sequence by [12] (Figure 1). Each protein sequence represented a sentence, and is broken into k-mers as words. Then, each word was mapped to a unique vector. All word vectors (W_1, W_2, W_3, … W_M) constitute a matrix $\mathbf{W}_{M\times V}$, where M words constitute M rows, and the column number is the number of hidden layer neurons (V). Similar to words, each sentence (a protein sequence) in the document containing N sequences was also mapped to a unique vector, represented by a row in matrix $\mathbf{D}_{N\times V}$. Further, the average of the sentence and word vectors was used to predict the output word vectors in a text window and formed the matrix $\mathbf{W'}_{V\times M}$. Stochastic Gradient Descent (SGD) and backpropagation were used to update weights. Resulting sentence vectors then can be used to for features in a machine learning model.

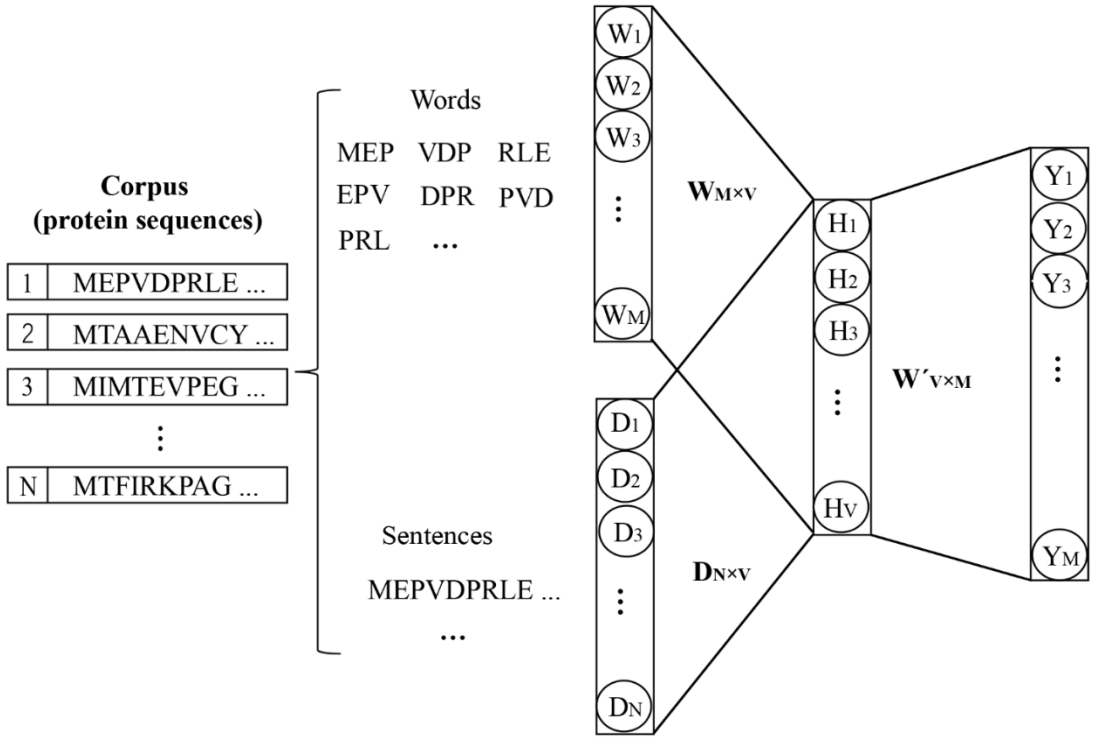

Figure 1. Framework of the doc2vec model training method

2. Transformer-Based Models

A transformer model is a neural network that learns context and meaning in sequential data such as text data or biological sequence by tracking relationships. Transformer models utilizes a set of mathematical techniques, named self-attention, to detect influence and dependencies between distant data elements in a sequence. It is originally presented by for machine translation task [13], and has been evolved with different variations. It has also been applied to other areas such as image recognition and bioinformatics domain.

Protein language models has been trained from large protein corpora, using the state-of-the-art Transformer models from the latest NLP research [14]. In chapter 5, six of the models were applied to extract features for our task of predicting glutarylation sites.

- **ProtBERT** and **ProtBert-BFD** are derived from the BERT model [15], trained on UniRef100 and BFD corpora, respectively.

- **ProtT5-XL-UniRef50** and **ProtT5-XL-BFD** are derived from the T5 model [16], trained on UniRef50 and BFD corpora, respectively.
- **ProtAlbert** is derived from the Albert model [17] trained on UniRef100 corpora.
- **ProtXLNet** is derived from the XLNet model [18], trained on UniRef100 corpora.

Protein embeddings (features) can be extracted from the last layer of this protein language model to be used for subsequent supervised training. This layer is a 2-dimensional array with a size of 1024 × length of sequence, except for the ProtAlbert model with an array size of 4096 × length of sequence. For the glutarylation prediction problem in Chapter 5, this feature is simplified by summing the vectors along the length of the sequence; hence, each feature group is now one-dimensional, with a length of 4096 for ProtAlbert and 1024 for the rest.

2.2 Imbalanced Data Handling

Many real-world datasets have imbalances. Class imbalance occurs when one class have far less number of samples compared to the other class. The class with less samples is called the minority class (or positive class), while the other class with more samples is called the majority class (or negative class). In most problems, we are interested in the rare cases or the minority class compared to the abundant (or normal) cases. Standard classifiers are not designed to handle highly imbalance data. This imbalance case may cause classifier to bias towards the majority class. Thus, class imbalance should be handled to improve detection of minority samples. A common approach for dealing with imbalance is preprocessing at the data level. The data is balanced before being processed in the classification algorithm.

2.2.1 Undersampling

In the undersampling technique, the majority class is reduced by removing some instances.

This method will achieve reducing the data from being highly imbalanced (for example 1:100 ratio) to a lesser ratio of imbalance (for example 1:10 or 1:3 or 1:1). One of the simplest method to undersample the majority class, is by random undersampling. In this method, a certain number of instance from the majority class is removed randomly. This method is very simple, yet has the drawback of potentially removing important samples.

2.1.2 Oversampling

In the oversampling technique, the minority class is increased through various means. This method will also achieve reducing the imbalance ratio of the classes. The simplest method to increase the minority class is random oversampling. The random oversampling technique involves selecting random examples from the minority class with replacement, and adding them in the training dataset. Another approach to increase the minority class is by generating artificial examples. Methods that fall into this type include Synthetic Minority Oversampling Technique (SMOTE) [19] and its variations, as well as Adaptive Synthetic (ADASYN) [20].

SMOTE

In the SMOTE method, for each sample in the minority class, artificial samples is created along the line segments between the sample and *k* minority class nearest neighbors. This neighbors are randomly chosen from the k nearest neighbors. The artificial samples are created in the following method: For the sample under consideration, find the difference between it and its chosen nearest neighbor. Multiply this difference by a random number [0-1], then add it to the original sample vector. This is basically selecting a random point between two nearest neighbor (minority) samples.

ADASYN

ADASYN assigned weighted distribution for different minority class examples based on

their level of difficulty in learning. For minority class examples that are more difficult to learn, more synthetic data will be generated compared to those minority samples that are easier to learn. The ADASYN method reduces the bias due to the class imbalance, as well as adaptively moving the classification decision boundary towards the hard examples.

2.3 Workflow for Supervised Classification

In finding the best classification model a general workflow for the model is presented in Figure 2 below. First, raw data is obtained from the real world. This dataset is usually not ready to be processed with classification algorithms, and need one or more preprocessing steps. These preprocessing steps may include feature extraction, data cleaning, and data transformation.

After the preprocessing step, the data is then split into two groups, training data set and testing data set. Generally the training data set is around 60-90% of the whole dataset. The training dataset is then used to find the best classification model. This model building may include a training and validation steps, either separately or combined in a validation system such as k-fold cross validation (Section 2.4.1). Validation process is important since classification algorithms need tweakings and parameter optimization. The output of model building is the best classification model.

Finally the testing dataset is used to evaluate the model. Using a separate test set ensured that the model is evaluated using data unseen during the training process. The predicted label of the test data then can be measured using various performance metric (Section 2.4.2).

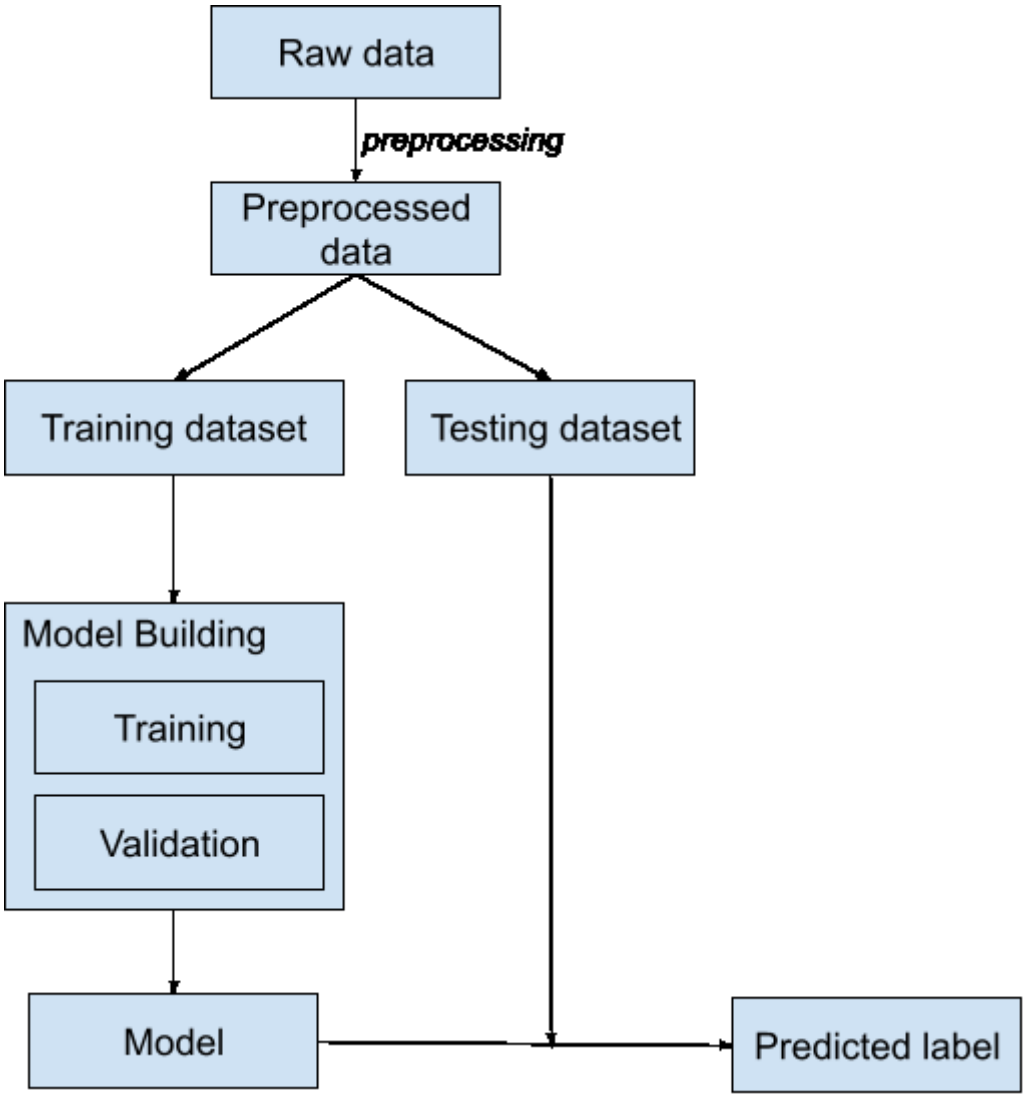

Figure 2. General workflow for building a classification model

2.4 Model Evaluation

During and after a model is trained, it needs to be evaluated. During the training, the evaluation is performed to fine tune the model and/or choose the best among several models. After the training process, evaluation is performed to estimate the model's performance in the future or in the real world.

2.4.1 k-Fold Cross Validation

A simple train/test split to validate a classification model can have bias due to the different

split that is possible. To handle this bias, k-fold cross validation can be used. In this method, the (training) data set is split into *k* equal sized groups (or folds). A fold is designated as the test set while the other *k*-1 folds is used for training the model. This training and test is repeated k times, each time with a different fold for testing. The evaluation of each fold is then averaged to get the overall performance of the model. An example of 5-fold cross validation is shown in Figure 3.

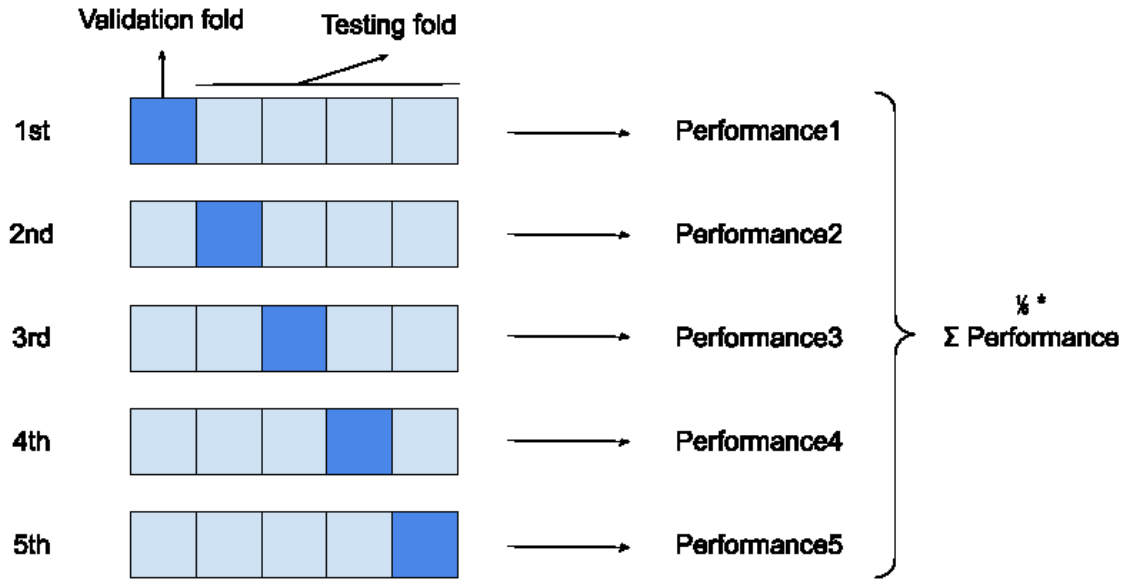

Figure 3. Example of 5-fold cross validation

2.4.2 Performance Metric

A confusion matrix is a summary of the prediction results of a test set on a classification task. For each class, correct and incorrect predictions are counted. A confusion matrix for binary classification problem is shown in Table 2. True Negative and True Positive happens when the predicted class matches the actual class. False Positive and False Negative happen when the predicted class are incorrect.

Table 2. Confusion matrix

	Predicted 0	Predicted 1
Actual 0	True Negative (TN)	False Positive (FP)
Actual 1	False Negative (FN)	True Positive (TP)

From the confusion matrix, various other performance metrics can be derived. The ones specifically used in this study are: recall (Rec), specificity (Spe), precision (Pre), accuracy (Acc), MCC, F1-score (F1), and area under the ROC curve (AUC). These six metrics were calculated with the following formulas:

$$\text{Rec} = \frac{TP}{TP + FN}$$

$$\text{Spe} = \frac{TN}{TN + FP}$$

$$\text{Pre} = \frac{TP}{TP + FP}$$

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

$$\text{F1} = 2 \times \frac{Rec.Pre}{Rec + Pre} \quad (12)$$

where *TP* is True Positive, *FP* is False Positive, *TN* is True Negative, and *FN* is False Negative.

Another metric, the AUC (Area Under Curve) is obtained by plotting recall against (1 – specificity) for every threshold and then calculating the area under the curve.

Chapter 3 Improving Human-Virus Protein-Protein Interaction Prediction

3.1 Introduction

Viral infection is a major health problem to humans, in addition to major cause of human death. As of July 2021, WHO reported there have been 196 million cases of COVID-19 and 4.2 million deaths confirmed [21]. In other news, chronic hepatitis is suffered by over 354 million people worldwide, with 8,000 new infections of hepatitis B and C every day, and more than one million deaths from liver disease and liver cancer occur every year [22]. Studying interactions between human and virus proteins is important in understanding host immune responses and viral infections [23]. This understanding will provide foundation in the development of strategies to prevent and combat viral diseases.

One main task in protein-protein interaction (PPI) identification is to predict whether a human protein interacts with another virus protein. Experimental techniques have been invaluable in compiling and collecting validated human-virus PPI. However, this approach is expensive and time-consuming and it is important to discover more PPI through computational approaches. Recently, research is focused on utilizing computational and machine learning methods to identify new or unknown protein-protein interactions [24].

One promising method for human-virus PPI identification is representing the protein-protein interaction as embedded protein sequence [12]. The authors showed that using doc2vec embedding on the proteins, combined with Random Forest classifier, they

achieved the best classification performance. Although the authors investigated the embedding parameters and experimented on 3 classifiers, other preprocessing methods have not been studied. Additional preprocessing before the classification stage may improve the performance.

3.2 Materials and methods

3.2.1 Dataset

We obtained the human-virus PPI dataset from [12] which is derived from Host-Pathogen Interaction Database (HPIDB) [25]. This dataset consisted of 249,184 pairs of human-virus PPI, and is imbalanced with a ratio 1:10 of positive to negative data. Next, a doc2vec model [10] is trained using proteins from Swiss-Prot [26]. The training scheme for the doc2vec model is based on previous work of ([12], [27]) where each protein is duplicated k times and split into non-overlapping k-mers with different groupings, before training the embedded protein model. For this study, 3-mer is chosen.

3.2.2 Workflow for Building the Model

The overall pipeline of our experiment is shown in Fig. 3. This pipeline is based on machine learning process with features based on protein embedding, but with a preprocessing step as emphasis. The human-virus PPI dataset and the doc2vec model were obtained as described in Section 3.2.1 above.

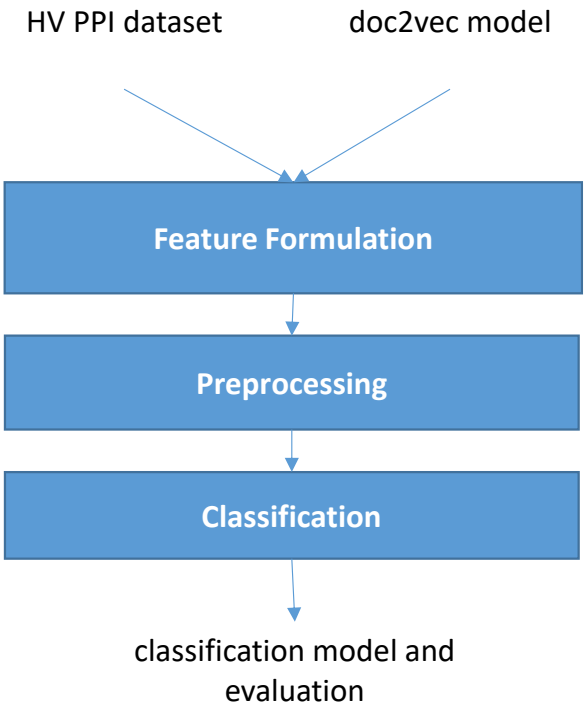

Figure 4. Workflow of the Experiment

For our experiment, there are 3 different parts in which various methods would be compared: (1) feature formulation, the methods used to create the feature vectors of the dataset, (2) data preprocessing, which consisted of various transformation methods and resampling methods, and (3) the classification method used for the classification task. In all evaluations of classification performance, the Area Under Curve (AUC) was applied. This metric is chosen in accordance with the previous study of the same dataset [12].

Before doing the various modifications, we introduce a basic procedure, in which a base method for each of the three processes is set. This basic procedure will serve as a comparison for the various modifications applied.

1. Base feature formulation method

For the feature formulation method, a basic method is used, in which each pair of human and virus protein is represented as summing all the representative human vectors,

summing all the virus vectors, and concatenating them (Fig. 3). An instance from the human-virus PPI dataset consisted of a pair of human and virus proteins with their label (either positive or negative interaction). These proteins are converted into feature vectors by getting their individual vector representations from the doc2vec model and combining them in certain ways. In our experiment 3-mer is used, so the model saved 3 different vector representations of each protein, with each vector having a length 16. To aid our explanation, let us name the three human protein vectors H1, H2, H3 respectively, while the virus vectors V1, V2, V3.

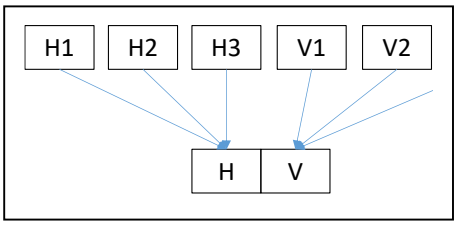

Figure 5. Combining Human Vectors H1, H2, H3 with Virus Vectors V1, V2, V3

2. Base data preprocessing method

In accordance with the previous study, no further data preprocessing is involved.

3. Base classifier

In accordance with the previous study, the classification method is Random Forest classifier, using the 5-fold cross-validation scheme. By several trials, the parameters used for the Random Forest are *nb_estimators*=1000 and *criterion='entropy'*.

This basic procedure for PPI classification will be used as a base comparison for modifications in any of the three steps: feature formulation, preprocessing, and classifier. The metric to compare is the Area Under Curve (AUC) value.

3.3 Result and Discussion

3.3.1 Experiments on Different Feature Formulation

We experimented on 6 different ways to combine the human and virus vector representations (Table 3). The first method is the base method. The fifth method is separated into 9 different combinations, in which one of the 3 human vectors is paired differently with one of the 3 virus vectors. Hence the total feature formulation methods investigated is 14. During this experiment, the preprocessing and classifier, are not changed i.e., the same as the base procedure. It is revealed that the best method is method 6, followed by method 1 (base method)

Table 3. Feature formulation methods

Method	Feature Formulation	Feature Length	AUC
1 (base)	concat(sum(H1,H2,H3), sum(V1,V2,V3))	32	0.9414
2	concat(H1,H2,H3,V1,V2,V3)	96	0.9408
3	sum(H1,H2,H3,V1,V2,V3)	16	0.8080
4	concat(sum(H1,V1), sum(H2,V2), sum(H3,V3))	48	0.8331
5-1	concat(H1,V1)	32	0.9322
5-2	concat(H1,V2)	32	0.9361
5-3	concat(H1,V3)	32	0.9366
5-4	concat(H2,V1)	32	0.9328
5-5	concat(H2,V2)	32	0.9373
5-6	concat(H2,V3)	32	0.9378
5-7	concat(H3,V1)	32	0.9320
5-8	concat(H3,V2)	32	0.9367

Method	Feature Formulation	Feature Length	AUC
5-9	concat(H3,V3)	32	0.9372
6	concat (concat(sum(H1,H2,H3), sum(V1,V2,V3)), concat(H1,H2,H3,V1,V2,V3))	128	**0.9418**

3.3.2 Experiments on Preprocessing Methods

For the second part of the experiments, two sets of modifications were investigated. The first is feature transformation (Table 4). PCA with all dimensions (32) performed slightly worse than no transformation. t-SNE and UMAP, even with parameter tuning, had much lower AUC. This showed that these transformations fail to improve the classification result.

Table 4. Classification result for different transformation methods

Transformation	Parameters	AUC
No transformation (base)	-	**0.9414**
PCA	dimension=32	0.9277
t-SNE	n=3, p=30	0.8689
UMAP	dimension=32, neighbors=50	0.8110

Another experiment on preprocessing was applying resampling methods to the dataset (see the result in Table 5). Random oversampling improved the AUC slightly. On the other hand, all undersampling methods performed worse than the original.

Table 5. Classification result for different sampling methods

Sampling	**Parameters**	**AUC**
No sampling (base)	-	0.9414
Random Oversampling	sampling_strategy=1	**0.9437**
	sampling_strategy=0.5	0.9434
	sampling_strategy=0.25	0.9420
Random Undersampling	sampling_strategy=1	0.9289
	sampling_strategy=0.5	0.9334
	sampling_strategy=0.25	0.9370
	sampling_strategy=0.2	0.9381
SMOTE	k=1	0.9370
	k=2	0.9365
	k=3	0.9357
	k=4	0.9355
	k=5	0.9354
	k=6	0.9350
	k=7	0.9346
	k=8	0.9349
	k=9	0.9340
	k=10	0.9342
BorderlineSMOTE	-	0.9344
ADASYN	n=1	0.9407
	n=3	0.9403
	n=5	0.9395

3.3.3 Experiments on Different Classification Algorithms

The next experiment investigated various classifiers and their best parameter (Table 6). The original classifier, Random Forest, gave the best performance with AUC 0.9414. This is in agreement with previous research. SVM and Multilayer Perceptron were the next best classifiers with AUC 0.8909 and 0.8862 respectively. The other classifiers performed poorly.

Table 6. Classification result for different classifiers

Classifier	**Best Parameters**	**AUC**
Random Forest (base)	nb_trees=1000	**0.9414**
SVM	kernel=rbf, *gamma=0.1, C=1*	0.8909
knn	k=6	0.8614
Naïve Bayes	default	0.7129
Logistic Regression	default	0.7100
Multi Layer Perceptron	default (hidden neurons=100)	0.8862
Decision Tree	default	0.7119

3.3.4 Combining Random Oversampling with Best Feature Formulation Method

From experimenting with feature formulation methods, the best method was Method 14. Furthermore, in the experiment with data preprocessing, Random Oversampling was the best. Next, we experimented in combining the two Method 6 with Random Oversampling (Method 6-ROS) and applied it to Random Forest and SVM (Table 7). For Random Forest classifier, both modifications increased the AUC even further from 0.9414 to 0.9448. For SVM, both modifications also increased the AUC from

0.8909 to 0.9147, but not better than the Random Forest classifier. Combining the best feature formulation and the best resampling method showed that the classification performance can be improved.

Table 7. Classification result for combination of feature formulation and transformation methods

Method Combination	**AUC**
Random Forest	
Method 1-NoTransformation-RandomForest	0.9414
Method 6-ROS-Random Forest	**0.9448**
SVM	
Method 1-NoTransformation-SVM	0.8909
Method 6-ROS-SVM	0.9147

3.4 Conclusion

In this chapter, we investigated various modifications to the process of human-virus PPI classification based on sequence-embedding and machine learning methods. It is found that modification to the feature formulation improved the classification performance. Another process that improved performance is applying random oversampling. However, combining them both at once improved the classification evaluation further from originally 0.9414 to 0.9448. We also confirm that Random Forest is the best classifier for this task and this dataset. For future study in this topic, we suggest to expand the modifications, such as experiment on other datasets, investigate different embedding scheme and/or different embedding parameters, investigate other

preprocessing methods, investigate simultaneous combinations of modifications as well as classification methods, in order to achieve better performance.

Chapter 4 Improving Lysine Glutarylation Prediction

4.1 Introduction

The post-translational modification (PTM) of amino acids dynamically changes the function of proteins and is actively studied in the field of molecular biology. Among various kinds of PTMs, lysine glutarylation is defined as an attachment of a glutaryl group to a lysine residue of a protein [28]. This modification was first detected via immunoblotting and mass spectrometry analysis and later validated using chemical and biochemical methods. It is suggested that this PTM may be a biomarker of aging and cellular stress [29]. Dysregulation of glutarylation is related to some metabolic diseases, including type 1 glutaric aciduria, diabetes, cancer, and neurodegenerative diseases [30]–[32]. Since the identification of glutarylated peptides using proteomics techniques is expensive and time-consuming, it is important to investigate computational models and predictors to rapidly identify glutarylation.

Based on a survey of previous research, various prediction models were proposed to distinguish glutarylation sites, including GlutPred [33], iGlu-Lys [34], MDDGlutar [35], RF-GlutarySite [36], and iGlu_Adaboost [37]. Although many models have been built to distinguish between positive and negative glutarylation sites, the performance of these methods remains limited. One challenge to this problem is finding a set of features to represent the protein subsequence, which enables a correct classification of glutarylation site. BERT models [15], and other transformer-based language models from natural language processing (NLP) research, show excellent performance for NLP tasks. These language models, having been adapted to biological sequences by treating them as sentences and then trained using large-scale protein corpora

[14], also show promise for various machine learning tasks in the bioinformatics domain.

Previous studies have investigated the use of pre-trained language models from BERT and BERT-like models to show its effectiveness as protein sequence representation for protein classification. For example, [2] proposed a new approach to predict flavin adenine dinucleotide (FAD) binding sites from transport proteins based on pre-training BERT, position-specific scoring matrix profiles (PSSM), and an amino acid index database (AAIndex). In another study, Liu built a predictor for protein lysine glycation sites using features extracted from pre-trained BERT models, which showed improved performance in terms of accuracy and AUC score compared to previous methods [38]. These studies demonstrate the suitability of utilizing BERT models to improve various protein classification tasks. Therefore, using embeddings from pre-trained BERT and BERT-like models has the potential to build an improved glutarylation prediction model.

In this chapter, we proposed a new prediction model to predict glutarylation sites (Figure 1) by incorporating features extracted from pre-trained protein models combined with features from handcrafted sequence-based features.

4.2 Materials and methods

4.2.1 Dataset

This study utilized unbalanced benchmark datasets compiled by [36] to build their predictor, RF-GlutarySite. This dataset collected positive glutarylation sites from various sources, including PLMD [39] and [32] and consisted of four different species (*Mus musculus, Mycobacterium tuberculosis, E. coli,* and HeLa cells), for a total of 749 sites from 234 proteins. Homologous sequences that showed ≥40% sequence identity were removed using the CD-HIT tool. The remaining proteins were converted into peptides with a fixed length of 23, with glutarylated lysine as the central residue, and 11

residues each upstream and downstream. Negative sites were generated in the same way, but the central lysine residue was not glutarylated. After removing homologous sequences, the final dataset consisted of 453 positive and 2043 negative sites. The distributions of the training and testing datasets are listed in Table 8. This dataset was also used to build the proposed predictor model iGlu_Adaboost [37].

Table 8. Number of positive and negative sites in training and test set

	Training set	Test set	
Positive sites	400	44	444
Negative sites	1703	203	1906
	2103	247	

4.2.2 Workflow for Building the Model

The workflow for building the model is presented in Figure 4. The extraction of numerical features from protein sequences or peptides is an important step before they can be utilized by machine learning algorithms. In this study, we investigated two types of features: classic sequence-based features and features derived from pre-trained transformer-based protein embeddings. Classic sequence-based features were extracted using the *iFeature* Python package [5].

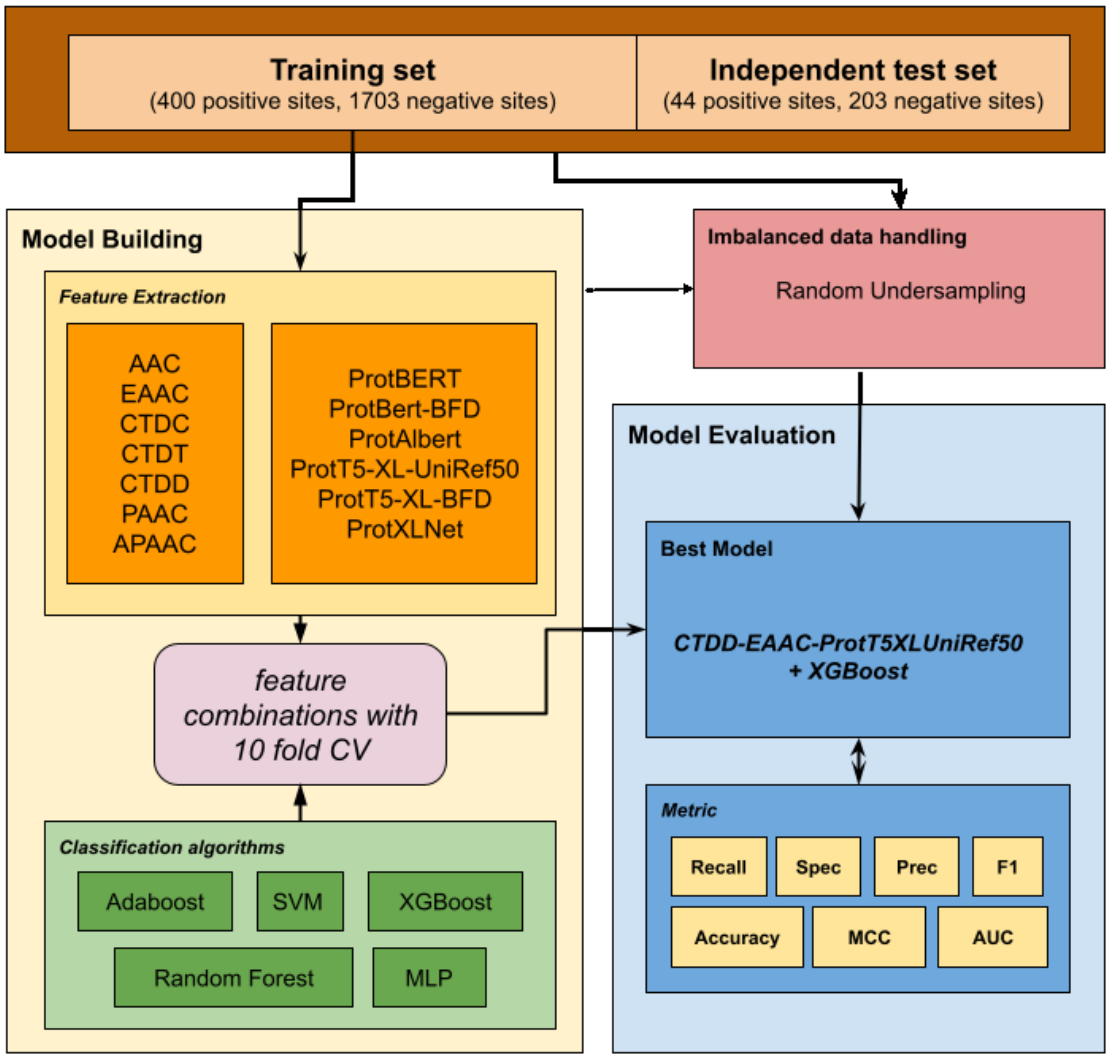

Figure 6. Workflow for building lysine glutarylation predictor

After preliminary experiments, seven feature groups were chosen for further investigation: AAC, EAAC, Composition/Transition/Distribution (CTD), pseudo-amino acid composition (PAAC), and amphiphilic pseudo-amino acid composition (APAAC). The second type of feature, embeddings from pre-trained transformer-based models, was extracted using models trained and provided by [14]. It consists of six feature sets from six protein models: ProtBERT, ProtBert-BFD, ProtAlbert, ProtT5-XL-UniRef50, ProtT5-XL-BFD, and ProtXLNet. The complete list of features investigated is shown in Table 3.

Table 9. Features investigated for method development

Group	Feature set	Length of features
Amino acid composition	AAC	20
	EAAC	380
C/T/D	CTDC	39
	CTDT	39
	CTDD	195
Pseudo amino acid composition	PAAC	35
	APAAC	50
Embeddings from pretrained transformer-based model	ProtBERT	1024
	ProtBert-BFD	1024
	ProtAlbert	4096
	ProtT5-XL-UniRef50	1024
	ProtT5-XL-BFD	1024
	ProtXLNet	1024

The next step was to combine two or more feature sets to evaluate further models, such as AAC-EAAC, AAC-CTDC, and AAC-ProtBert. For this, we limited the embedding features to a maximum of one in the combination. Five classification algorithms were included in the experiments: Adaboost, XGBoost, SVM (with RBF kernel), random forest (RF), and multilayer perceptron (MLP).

Our best model combines the features of CTDD, AAC, and ProtT5-XL-

UniRef50 with the XGBoost classification algorithm. This model, with the model of the best feature set from sequence-based feature groups and the model of the best feature set from the protein embedding feature group, was then evaluated with an independent dataset. For independent testing, the entire training set was used to develop a model. In both model building and independent testing, a random under-sampling method was used to balance the training dataset, while the testing dataset was not resampled to reflect performance in the real-world unbalanced scenario.

4.2.3 Imbalanced Data Handling

In the glutarylation dataset, the number of negative samples is nearly four times that of positive samples. This imbalance may affect the performance of classifiers because they are biased to misclassify a positive example as a negative one [40]. A common strategy to solve this problem is by data re-sampling, either adding minority samples (over-sampling) or reducing majority samples (under-sampling). In this study, we implemented a random under-sampling strategy [41] after preliminary experiments with various re-sampling methods.

4.3 Result and Discussion

4.3.1 Models Based on Sequence-based Feature Set

We calculated the cross-validation performance for each sequence-based feature set using five supervised classifiers: AdaBoost, MLP, RF, SVM, and XGBoost. The performances of these classifiers are shown in Table 4. It can be observed that no classifier is the best for all feature groups. For example, using AAC features, MLP performs the best based on the AUC score. However, using EAAC features, the RF model has the best performance, whereas MLP has the poorest. Among the six different feature sets, the best model achieved was using EAAC features combined with RF, with an AUC score of

0.6999. This model also had the best specificity, precision, and accuracy compared to the other models.

Table 10. Cross validation result of models from sequence-based features

Feature groups	Classifier	Rec	Spe	Pre	Acc	MCC	F1	AUC
AAC	Adaboost	0.6120	0.6013	0.2654	0.6033	0.1690	0.3700	0.6433
	MLP	0.6520	0.6192	0.2864	0.6255	0.2150	0.3977	0.6864
	Random Forest	0.6190	0.5809	0.2575	0.5881	0.1576	0.3635	0.6378
	SVM	0.6395	0.5969	0.2714	0.6050	0.1868	0.3808	0.6651
	XGBoost	0.5917	0.5482	0.2353	0.5565	0.1102	0.3362	0.6101
EAAC	Adaboost	0.5983	0.6015	0.2608	0.6009	0.1584	0.3629	0.6384
	MLP	0.5850	0.5946	0.2530	0.5928	0.1422	0.3529	0.6323
	Random Forest	0.6450	**0.6598**	**0.3089**	**0.6570**	0.2450	0.4171	0.6999
	SVM	0.5967	0.6434	0.2821	0.6345	0.1923	0.3827	0.6571
	XGBoost	0.6408	0.6385	0.2945	0.6389	0.2230	0.4030	0.6834
CTDC	Adaboost	**0.7050**	0.5518	0.2699	0.5809	0.2019	0.3901	0.6641
	MLP	0.6867	0.6034	0.2905	0.6193	0.2300	0.4073	0.6912
	Random Forest	0.6408	0.5676	0.2579	0.5815	0.1639	0.3676	0.6556
	SVM	0.6842	0.5657	0.2705	0.5882	0.1966	0.3874	0.6765
	XGBoost	0.6367	0.5754	0.2605	0.5871	0.1672	0.3693	0.6450
CTDT	Adaboost	0.6208	0.5762	0.2566	0.5847	0.1556	0.3627	0.6261
	MLP	0.6408	0.5756	0.2622	0.5880	0.1708	0.3717	0.6439
	Random Forest	0.6025	0.5982	0.2603	0.5990	0.1588	0.3633	0.6241

	SVM	0.6425	0.5841	0.2661	0.5952	0.1787	0.3760	0.6493
	XGBoost	0.5783	0.5668	0.2390	0.5690	0.1147	0.3378	0.6015
CTDD	Adaboost	0.6358	0.6046	0.2744	0.6106	0.1904	0.3831	0.6531
	MLP	0.5942	0.5365	0.2434	0.5475	0.1120	0.3297	0.6065
	Random Forest	0.6967	0.6164	0.2994	0.6316	**0.2476**	**0.4185**	0.6987
	SVM	0.6675	0.6111	0.2877	0.6218	0.2206	0.4017	0.6794
	XGBoost	0.6675	0.6201	0.2927	0.6291	0.2282	0.4064	0.6847
PAAC	Adaboost	0.5942	0.6052	0.2611	0.6031	0.1581	0.3626	0.6253
	MLP	0.5958	0.5717	0.2462	0.5763	0.1321	0.3482	0.6261
	Random Forest	0.6375	0.5809	0.2633	0.5917	0.1723	0.3723	0.6413
	SVM	0.6617	0.5905	0.2752	0.6041	0.1990	0.3885	0.6745
	XGBoost	0.6217	0.5731	0.2554	0.5823	0.1537	0.3615	0.6375
APAAC	Adaboost	0.6125	0.5976	0.2634	0.6004	0.1662	0.3682	0.6367
	MLP	0.5658	0.5904	0.2450	0.5857	0.1237	0.3416	0.6162
	Random Forest	0.6458	0.5831	0.2671	0.5950	0.1805	0.3776	0.6464
	SVM	0.6650	0.5970	0.2794	0.6099	0.2069	0.3932	0.6777
	XGBoost	0.6425	0.5694	0.2596	0.5833	0.1668	0.3695	0.6375

4.3.2 Models Based on Embeddings from Pre-trained Transformer Models

Based on the embeddings extracted from the pre-trained transformer models, we evaluated the same five supervised classifiers. The performance results of the models are presented in Table 5. The combination of the ProtBERT model and SVM can match the recall score with the classic sequence-based feature result. However, all other metrics were lower. In this experiment, the best model with respect to the AUC score was a

combination of features from the ProtAlbert model and SVM classifier (AUC = 0.6744). This model also had the highest cross-validation scores for precision, MCC, and F1-score. It can also be noted that out of the six models, SVM performed best on four of them compared to the other machine learning algorithms.

Table 11. Cross validation result of models from pre-trained transformer models

Feature groups	Classifier	Rec	Spe	Pre	Acc	MCC	F1	AUC
ProtBERT	Adaboost	0.5767	0.5680	0.2389	0.5697	0.1142	0.3374	0.5996
	MLP	0.5892	0.5608	0.2395	0.5662	0.1187	0.3396	0.6128
	Random Forest	0.5567	**0.6426**	0.2681	**0.6262**	0.1602	0.3616	0.6415
	SVM	**0.7042**	0.4775	0.2420	0.5207	0.1475	0.3578	0.6275
	XGBoost	0.6033	0.6007	0.2619	0.6012	0.1616	0.3649	0.6398
ProtBert-BFD	Adaboost	0.5433	0.5547	0.2231	0.5525	0.0773	0.3162	0.5776
	MLP	0.5900	0.5645	0.2420	0.5694	0.1218	0.3430	0.6076
	Random Forest	0.5383	0.6230	0.2510	0.6069	0.1289	0.3421	0.6122
	SVM	0.6242	0.5819	0.2595	0.5899	0.1626	0.3662	0.6420
	XGBoost	0.5908	0.5733	0.2453	0.5766	0.1295	0.3464	0.6142
ProtAlbert	Adaboost	0.5875	0.5753	0.2450	0.5776	0.1284	0.3456	0.6193
	MLP	0.5858	0.6189	0.2657	0.6126	0.1646	0.3615	0.6407
	Random Forest	0.5808	0.6316	0.2703	0.6220	0.1697	0.3687	0.6535
	SVM	0.6283	0.6136	**0.2767**	0.6164	**0.1919**	**0.3840**	**0.6744**

	XGBoost	0.6092	0.5927	0.2604	0.5958	0.1597	0.3646	0.6477
ProtT5-XL-UniRef50	Adaboost	0.5533	0.5655	0.2306	0.5632	0.0938	0.3254	0.5897
	MLP	0.6192	0.5633	0.2501	0.5739	0.1439	0.3558	0.6296
	Random Forest	0.5608	0.6171	0.2562	0.6064	0.1419	0.3515	0.6237
	SVM	0.6583	0.5710	0.2653	0.5876	0.1807	0.3777	0.6600
	XGBoost	0.5933	0.5807	0.2497	0.5831	0.1377	0.3509	0.6183
ProtT5-XL-BFD	Adaboost	0.5892	0.5600	0.2395	0.5656	0.1175	0.3405	0.5959
	MLP	0.6000	0.5768	0.2502	0.5812	0.1396	0.3529	0.6188
	Random Forest	0.5392	0.6163	0.2485	0.6017	0.1242	0.3399	0.6145
	SVM	0.6550	0.5625	0.2604	0.5801	0.1711	0.3724	0.6548
	XGBoost	0.5858	0.5862	0.2490	0.5862	0.1361	0.3489	0.6224
ProtXLNet	Adaboost	0.5125	0.5343	0.2057	0.5302	0.0369	0.2934	0.5421
	MLP	0.5325	0.5248	0.2081	0.5262	0.0450	0.2991	0.5463
	Random Forest	0.5050	0.5668	0.2152	0.5551	0.0568	0.3015	0.5511
	SVM	0.4742	0.5770	0.2103	0.5575	0.0408	0.2900	0.5460
	XGBoost	0.5642	0.5504	0.2274	0.5530	0.0902	0.3238	0.5652

4.3.3 Models Based on Combination of Sequence-based Feature and Pre-trained Transformer Models Feature Set

To obtain the best model, we tested various combinations of two or more feature

sets to evaluate further models, such as AAC-EAAC, AAC-CTDC, and AAC-ProtBert. For this, we limited the embedding features to a maximum of one set in the combination. Similar to previous experiments, five classification algorithms were used: AdaBoost, XGBoost, SVM (RBF kernel), RF, and MLP.

Our best model, ProtTrans-Glutar, uses a combination of the features CTDD, EAAC, and ProtT5-XL-UniRef50 with the XGBoost classification algorithm. The performance of this model is shown in Table 6, with comparison to the best model from sequence-based features (EAAC with RF classifier) and the best model from embeddings of the protein model (ProtAlbert with SVM classifier). According to the cross-validation performance on training data, this model has the best AUC and recall compared with models with features from only one group. These three models were then evaluated using an independent dataset (Figure 5). This test result shows that ProtTrans-Glutar outperformed the other two models in terms of AUC, recall, precision, MCC, and F1-score. However, it is severely worse in terms of specificity and slightly worse in terms of accuracy compared to the EAAC+RF model.

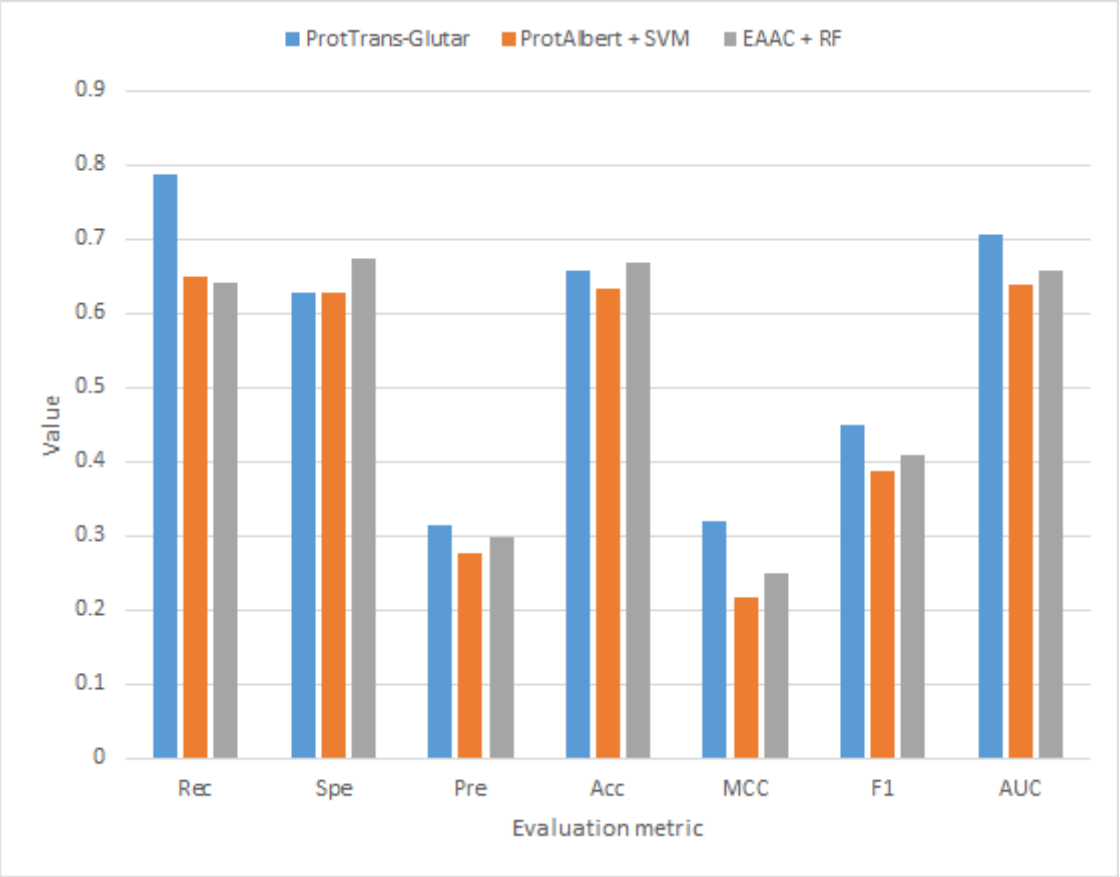

Figure 7. Evaluation of ProTrans-Glutar using independent test set

As shown in the ROC curves of the three models (Figure 6), EAAC+RF performed better for low values of FPR, but for larger values, ProtTrans-Glutar performed better. It is also noted that ProtAlbert+SVM performed worse for most values of FPR. Overall, ProtTrans-Glutar was the best model with an AUC of 0.7075.

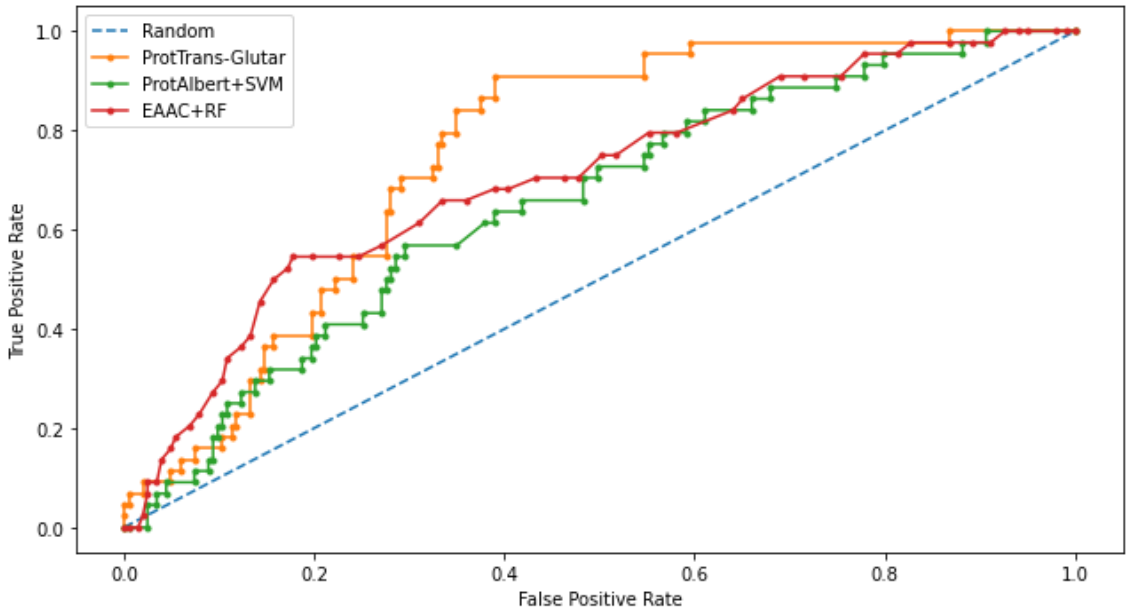

Figure 8. ROC Curves evaluation on independent test set

4.3.4 Discussion

From our study, it was shown that building prediction models from traditional sequence-based features only provided limited performance (Table 10). It was also shown that using only embeddings from pre-trained protein models gave slightly worse results, except that the recall performance was almost the same (Table 11). When we combined the features from these two groups, we found that the best performance was achieved by the combination of the features CTDD, EAAC, and ProtT5-XL-UniRef50 with the XGBoost classifier (independent test AUC = 0.7075). This indicated that ProtT5-XL-UniRef50 features on their own are not the best embedding model during the individual feature evaluation (see Table 5), but combined with CTDD and EAAC, it outperformed the other models. It is worth mentioning that Elnaggar et al. (2021), who developed and trained protein models, revealed that ProtT5 models outperformed state-of-the-art models in protein classification tasks, namely in prediction of localization (10-class classification) and prediction of membrane/other (binary classification), compared to other embedding

models.

For further evaluation, we compared our model with previous glutarylation site prediction models (Table 12). The first three models, GlutPred, iGlu-Lys, and MDDGlutar, used datasets that were different from our model and are shown for reference. The other model, iGlu_Adaboost, utilized the same public dataset as for our model and contained glutarylation sites from the same four species. ProtTrans-Glutar outperformed the other models in terms of the recall performance (Rec = 0.7864 for unbalanced data). This high recall suggests that this model can be useful for uncovering new and potential glutarylation sites.

Table 12. Performance comparison of existing models

Models	**Resources**	**Rec**	**Spe**	**Pre**	**Acc**	**MCC**	**F1**	**AUC**
GlutPred	PLMD	0.5179	0.7850	0.2397	0.7541	0.2238	n/a	0.7663
iGlu-Lys	PLMD	0.5143	0.9531	n/a	0.8853	0.52	n/a	0.8842
MDDGlutar	PLMD	0.652	0.739	n/a	0.71	0.38	n/a	n/a
iGlu_AdaBoost	PLMD, NCBI, Swiss-Prot	0.7273	0.7192	0.3596	0.7207	0.36	0.48	0.6300
ProtTrans-Glutar	PLMD, NCBI, Swiss-Prot	0.7822	0.6286	0.3147	0.6567	0.3196	0.4494	0.7075

Furthermore, we also evaluated our model by using a balanced training and testing dataset using random under-sampling for comparison with the RF-GlutarySite model (Table 13), which uses the same dataset but is balanced before evaluating performance. Because the authors of RF-GlutarySite did not provide their data after the resampling process, we performed the experiments 10 times to handle variance from the

under-sampling. The ProtTrans-Glutar model showed a higher recall score of 0.7864 compared to RF-GlutarySite (0.7410), in addition to a slightly higher accuracy, MCC, and F1-score. However, the specificity and precision scores were lower.

Table 13. Performance comparison with RF-GlutarySite using balanced train and test data

Models	Resources	Rec	Spe	Pre	Acc	MCC	F1	AUC
RF-GlutarySite*	PLMD, NCBI, Swiss-Prot	0.741	0.685	0.72	0.713	0.43	0.72	0.72
ProtTrans-Glutar (balanced)	PLMD, NCBI, Swiss-Prot	0.7864	0.6455	0.6955	0.7159	0.4388	0.7358	0.7159

**RF-GlutarySite model balanced the training and testing dataset using undersampling*

In summary, the model improved the recall score compared to the existing models but did not improve other metrics. However, we would like to point out that GlutPred, iGlu-Lys, and MDDGlutar based their glutarylation datasets on less diverse sources (two species only), whereas ProtTrans-Glutar with RF-GlutarySite and iGlu_Adaboost utilized newer datasets (four species). The more diverse source of glutarylation sites in the data may present more difficulty in improving performance, especially in terms of specificity and accuracy. Compared with iGlu_Adaboost, which used the same dataset, our model improved their recall and AUC scores. Despite this, the specificity is worse and will be a challenge for future research.

5.4 Conclusion

In this chapter, we presented a new glutarylation site predictor by incorporating embeddings from pretrained protein models as features. This method, which is termed ProtTrans-Glutar, combines three feature sets: EAAC, CTDD, and ProtT5-XL-UniRef50. Random under-sampling was used in conjunction with the XGBoost classifier to train the model. The performance evaluations obtained from this model for recall, specificity, and AUC are 0.7864, 0.6286, and 0.7075, respectively. Compared to other models using the same dataset of more diverse sources of glutarylation sites, this model outperformed the existing model in terms of recall and AUC score and could potentially be used to complement previous models to reveal new glutarylated sites. In the future, refinements can be expected through further experiments, such as applying other feature selection methods, feature processing, and investigating deep learning models.

Chapter 6 Summary and Future Research

6.1. Summary

To summarize, we investigated using protein embedding features for protein classification tasks with class imbalance using two different classification tasks.

1. We improved human-virus PPI classification by applying random oversampling and modifying some preprocessing steps

2. We proposed a new glutarylation site predictor, by combining 3 feature sets, random undersampling, and XGBoost algorithm.

6.2 Future Research

Further study is needed to improve protein classification tasks. Our suggestion is as follows.

1. More variations of feature extraction methods need to be explored.

2. Another promising direction is to experiment using deep learning algorithms for the classification.

Bibliography

[1] F. Cui, Z. Zhang, and Q. Zou, "Sequence representation approaches for sequence-based protein prediction tasks that use deep learning," *Brief. Funct. Genomics*, vol. 20, no. 1, pp. 61–73, Mar. 2021, doi: 10.1093/bfgp/elaa030.

[2] Q.-T. Ho, T.-T.-D. Nguyen, N. Q. Khanh Le, and Y.-Y. Ou, "FAD-BERT: Improved prediction of FAD binding sites using pre-training of deep bidirectional transformers," *Comput. Biol. Med.*, vol. 131, p. 104258, Apr. 2021, doi: 10.1016/j.compbiomed.2021.104258.

[3] S. M. Ali Shah, S. W. Taju, Q.-T. Ho, T.-T.-D. Nguyen, and Y.-Y. Ou, "GT-Finder: Classify the family of glucose transporters with pre-trained BERT language models," *Comput. Biol. Med.*, vol. 131, p. 104259, Apr. 2021, doi: 10.1016/j.compbiomed.2021.104259.

[4] M. Bhasin and G. P. S. Raghava, "Classification of Nuclear Receptors Based on Amino Acid Composition and Dipeptide Composition," *J. Biol. Chem.*, vol. 279, no. 22, pp. 23262–23266, May 2004, doi: 10.1074/jbc.M401932200.

[5] Z. Chen *et al.*, "iFeature: a Python package and web server for features extraction and selection from protein and peptide sequences," *Bioinformatics*, vol. 34, no. 14, pp. 2499–2502, Jul. 2018, doi: 10.1093/bioinformatics/bty140.

[6] C. Z. Cai, "SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence," *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3692–3697, Jul. 2003, doi: 10.1093/nar/gkg600.

[7] I. Dubchak, I. Muchnik, S. R. Holbrook, and S. H. Kim, "Prediction of protein folding class using global description of amino acid sequence.," *Proc. Natl. Acad. Sci.*, vol. 92, no. 19, pp. 8700–8704, Sep. 1995, doi: 10.1073/pnas.92.19.8700.

[8] K.-C. Chou, "Prediction of protein cellular attributes using pseudo-amino acid composition," *Proteins Struct. Funct. Genet.*, vol. 43, no. 3, pp. 246–255, May 2001, doi: 10.1002/prot.1035.

[9] K.-C. Chou, "Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes," *Bioinformatics*, vol. 21, no. 1, pp. 10–19, Jan. 2005, doi: 10.1093/bioinformatics/bth466.

[10] Q. V. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," *ArXiv14054053 Cs*, May 2014, Accessed: Jul. 31, 2021. [Online]. Available: http://arxiv.org/abs/1405.4053

[11] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed

Representations of Words and Phrases and their Compositionality," *ArXiv13104546 Cs Stat*, Oct. 2013, Accessed: Jul. 31, 2021. [Online]. Available: http://arxiv.org/abs/1310.4546

[12] X. Yang, S. Yang, Q. Li, S. Wuchty, and Z. Zhang, "Prediction of human-virus protein-protein interactions through a sequence embedding-based machine learning method," *Comput. Struct. Biotechnol. J.*, vol. 18, pp. 153–161, 2020, doi: 10.1016/j.csbj.2019.12.005.

[13] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[14] A. Elnaggar *et al.*, "ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3095381.

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *ArXiv181004805 Cs*, May 2019, Accessed: Feb. 23, 2022. [Online]. Available: http://arxiv.org/abs/1810.04805

[16] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *ArXiv191010683 Cs Stat*, Jul. 2020, Accessed: Feb. 23, 2022. [Online]. Available: http://arxiv.org/abs/1910.10683

[17] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *ArXiv190911942 Cs*, Feb. 2020, Accessed: Feb. 23, 2022. [Online]. Available: http://arxiv.org/abs/1909.11942

[18] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," *ArXiv190608237 Cs*, Jan. 2020, Accessed: Feb. 23, 2022. [Online]. Available: http://arxiv.org/abs/1906.08237

[19] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[20] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational*

Intelligence), Hong Kong, China, Jun. 2008, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

[21] "WHO Coronavirus (COVID-19) Dashboard." https://covid19.who.int (accessed Jul. 31, 2021).

[22] "'Hepatitis can't wait' - WHO commemorates World Hepatitis Day 2021." https://www.who.int/news/item/28-07-2021-hepatitis-can-t-wait---who-commemorates-world-hepatitis-day-2021 (accessed Jul. 31, 2021).

[23] M. D. Dyer, T. M. Murali, and B. W. Sobral, "The Landscape of Human Proteins Interacting with Viruses and Other Pathogens," *PLoS Pathog.*, vol. 4, no. 2, p. e32, Feb. 2008, doi: 10.1371/journal.ppat.0040032.

[24] H. Chen *et al.*, "Systematic evaluation of machine learning methods for identifying human–pathogen protein–protein interactions," *Brief. Bioinform.*, vol. 22, no. 3, p. bbaa068, May 2021, doi: 10.1093/bib/bbaa068.

[25] M. G. Ammari, C. R. Gresham, F. M. McCarthy, and B. Nanduri, "HPIDB 2.0: a curated database for host–pathogen interactions," *Database*, vol. 2016, p. baw103, 2016, doi: 10.1093/database/baw103.

[26] "reviewed:yes in UniProtKB." https://www.uniprot.org/uniprot/?query=reviewed:yes (accessed Jul. 31, 2021).

[27] E. Asgari and M. R. K. Mofrad, "Continuous Distributed Representation of Biological Sequences for Deep Proteomics and Genomics," *PLOS ONE*, vol. 10, no. 11, p. e0141287, Nov. 2015, doi: 10.1371/journal.pone.0141287.

[28] J. V. Lee *et al.*, "Akt-Dependent Metabolic Reprogramming Regulates Tumor Cell Histone Acetylation," *Cell Metab.*, vol. 20, no. 2, pp. 306–319, Aug. 2014, doi: 10.1016/j.cmet.2014.06.004.

[29] R. Harmel and D. Fiedler, "Features and regulation of non-enzymatic post-translational modifications," *Nat. Chem. Biol.*, vol. 14, no. 3, pp. 244–252, Mar. 2018, doi: 10.1038/nchembio.2575.

[30] C. Carrico, J. G. Meyer, W. He, B. W. Gibson, and E. Verdin, "The Mitochondrial Acylome Emerges: Proteomics, Regulation by Sirtuins, and Metabolic and Disease Implications," *Cell Metab.*, vol. 27, no. 3, pp. 497–512, Mar. 2018, doi: 10.1016/j.cmet.2018.01.016.

[31] B. Osborne, N. L. Bentley, M. K. Montgomery, and N. Turner, "The role of mitochondrial sirtuins in health and disease," *Free Radic. Biol. Med.*, vol. 100, pp. 164–174, Nov. 2016, doi: 10.1016/j.freeradbiomed.2016.04.197.

[32] M. Tan *et al.*, "Lysine Glutarylation Is a Protein Posttranslational Modification Regulated by SIRT5," *Cell Metab.*, vol. 19, no. 4, pp. 605–617, Apr. 2014, doi:

10.1016/j.cmet.2014.03.014.

[33] Z. Ju and J.-J. He, “Prediction of lysine glutarylation sites by maximum relevance minimum redundancy feature selection,” *Anal. Biochem.*, vol. 550, pp. 1–7, Jun. 2018, doi: 10.1016/j.ab.2018.04.005.

[34] Y. Xu, Y. Yang, J. Ding, and C. Li, “iGlu-Lys: A Predictor for Lysine Glutarylation Through Amino Acid Pair Order Features,” *IEEE Trans. NanoBioscience*, vol. 17, no. 4, pp. 394–401, Oct. 2018, doi: 10.1109/TNB.2018.2848673.

[35] K.-Y. Huang, H.-J. Kao, J. B.-K. Hsu, S.-L. Weng, and T.-Y. Lee, “Characterization and identification of lysine glutarylation based on intrinsic interdependence between positions in the substrate sites,” *BMC Bioinformatics*, vol. 19, no. S13, p. 384, Feb. 2019, doi: 10.1186/s12859-018-2394-9.

[36] H. J. AL-barakati, H. Saigo, R. H. Newman, and D. B. Kc, “RF-GlutarySite: a random forest based predictor for glutarylation sites,” *Mol. Omics*, vol. 15, no. 3, pp. 189–204, 2019, doi: 10.1039/C9MO00028C.

[37] L. Dou, X. Li, L. Zhang, H. Xiang, and L. Xu, “iGlu_AdaBoost: Identification of Lysine Glutarylation Using the AdaBoost Classifier,” *J. Proteome Res.*, vol. 20, no. 1, pp. 191–201, Jan. 2021, doi: 10.1021/acs.jproteome.0c00314.

[38] Y. Liu, Y. Liu, G.-A. Wang, Y. Cheng, S. Bi, and X. Zhu, “BERT-Kgly: A Bidirectional Encoder Representations From Transformers (BERT)-Based Model for Predicting Lysine Glycation Site for Homo sapiens,” *Front. Bioinforma.*, vol. 2, p. 834153, Feb. 2022, doi: 10.3389/fbinf.2022.834153.

[39] H. Xu, J. Zhou, S. Lin, W. Deng, Y. Zhang, and Y. Xue, “PLMD: An updated data resource of protein lysine modifications,” *J. Genet. Genomics*, vol. 44, no. 5, pp. 243–250, May 2017, doi: 10.1016/j.jgg.2017.03.007.

[40] Haibo He and E. A. Garcia, “Learning from Imbalanced Data,” *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[41] H. He and Y. Ma, Eds., *Imbalanced learning: foundations, algorithms, and applications*. Hoboken, New Jersey: John Wiley & Sons, Inc, 2013.