MACHINE LEARNING BASED TO PREDICT B-CELL EPITOPE REGION UTILIZING PROTEIN FEATURES

Fatema Nafa and Ryan Kanoff

Department of Computer Science, Salem State University, Salem, USA

ABSTRACT

Considering the current state of Covid-19 pandemic, vaccine research and production is more important than ever. Antibodies recognize epitopes, which are immunogenic regions of antigen, in a very specific manner, to trigger an immune response. It is extremely difficult to predict such locations, yet they have substantial implications for complex humoral immunogenicity pathways. This paper presents a machine learning epitope prediction model. The research creates several models to test the accuracy of B-cell epitope prediction based solely on protein features. The goal is to establish a quantitative comparison of the accuracy of three machine learning models, XGBoost, CatBoost, and LightGbM. Our results found similar accuracy between the XGBoost and LightGbM models with the CatBoost model having the highest accuracy of 82%. Though this accuracy is not high enough to be considered reliable it does warrant further research on the subject.

KEYWORDS

machine learning models; data exploratory techniques; B-cell epitope prediction.

1. INTRODUCTION

Given the current rate of globalization, we can expect with near certainty that we will see events similar to the Covid-19 pandemic in the coming years. This makes vaccine research and development, especially the speed and efficiency of these processes, more relevant than ever. An essential aspect of vaccine research and development is B-cell epitope prediction. B-cell epitope regions are areas of antigen proteins that when recognized by B-cells produce large amounts of antigen-specific antibodies. Historically, B-cell epitope predication has relied primarily on sequence data and not protein features. By utilizing these protein features we can potentially increase the speed of B-cell epitope prediction which may help expediate the research and development of new vaccines.

For the purpose of this paper, we used several different machine learning models to test the relationship between protein features and B-cell epitope regions in an attempt to find this highest accuracy possible. The input to our algorithm is a dataset consisting of protein/amino acid sequences and their associated protein features and target behavior (anti-body inducing behavior). Three different machine learning models applied to this dataset. All of these models utilize boosting algorithms, specifically gradient boosting. These algorithms function by first creating weak learners and converting them to strong learners [1]. XGBoost or extreme gradient boost, is a gradient boosting decision tree (GBDT) model that focuses on speed and performance [2]. In recent years it has gained significant popularity through its ability to yield highly accurate results from large datasets. LightBGM is another gradient boosting tree-based model. Alongside

David C. Wyld et al. (Eds): AI, AIMLNET, BIOS, BINLP, CSTY, MaVaS, SIGI - 2022
pp. 115-122, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121811

the expected speed and performance of a gradient boosting model, LightBGM offers reduced memory usage which can help facilitate large-scale data processing. The final model, CatBoost, is another GBDT model. It behaves much like similar models; however, it specializes in data processing speed, with the potential to be notably faster than the previously mentioned models [3]. Each of these models will output an accuracy for the prediction of the target behaviour based on the features.

2. RELATED WORK

Machine learning algorithms have become one of the foremost methodologies within the field of bioinformatics and more specifically, immunoinformatic, for the purpose of predicting B-cell epitope regions. The following section will detail a number of papers regarding B-cell epitope prediction to provide a basic survey of the topic. Three of the most methodologies used include Naïve Bayes Classification, Support Vector Machine and Artificial Neural Networks [2]. Many papers that explore B-cell epitope region prediction will include at least one of these as part of their methodology. J Chen et al used the amino acid pair (AAP) antigenicity scale to predict B-cell epitopes. They saw significantly improved performance with the AAP antigenicity scale by utilizing support vector machine rather than existing scales [4].

Tao Lui et al utilized deep learning, a type of neural network with many hidden nodes and hidden layers, to create a predictive model for linear B-cell epitopes. They obtained sample peptides from the IEDB database and used this data to build a feed forward deep neural network. This ensemble prediction model was named DLBEpitope and performed better than current major models [5].

A great deal of success has also been made less conventional models such as BERT-based epitope prediction.

While BERT models are typically used for natural language processing, Minjun Park et al were able to create a Bert-based model, EpiBERTope. They pre-trained it with the Swiss-Prot protein database so that it could predict linear and structural epitope while exclusively relying on protein sequences. This model outperformed all the classic benchmark models include random forest, gradient boosting, naïve bayes and support vector machine [6].

3. PROBLEM DEFINATION

The main motivation behind this work is, to highlight the work done in B-cell epitope prediction using machine learning models, the limitation and the strength of each work presented in this work. Also, introducing machine learning methods to predict the B-cell epitope, in additionally, to provide future directions in terms of the limitation of the proposed method.

4. METHODOLOGY

This section covers pre-processing steps, and the methodology used in this research.

4.1. Data Pre-Processing

In this work Python [7], [8] is used for performing the prediction. Python is a general-purpose programming language with most data analysis features provided by NumPy and pandas. It has efficient high-level data structures and an object-oriented programming technique that is simple but effective.

This dataset was obtained from IEDB and UniProt available for public [9]. This dataset consists of three csv files, input_bcell, input Sars and input covid. Input_bcell is by far the largest file with dimensions 14387 x 14. All three of these files have the same columns excluding input Sars which does not have the target column. The first few columns are identifying pieces for the protein sequences and are not utilized by this paper. These include the parent protein id, protein sequence, peptide start position, peptide end position and peptide sequence. All of these columns are dropped in pre-processing because they are not needed when exclusively examining the protein features.

4.2. ML Models

This work implements Boosting algorithms [2], [10]. Gradient boosting is an algorithm that stands out for its predictability and speed, especially when dealing with big and complicated datasets[11], [12]. Gradient Boosting algorithm has three main components, loss function, weak learner, and additive model. The main goals of Gradient Boosting algorithms are to improve the prediction power by converting a number of weak learners to strong learners. Boosting algorithms work on the idea of first building a model on the training dataset, then building a second model to correct the errors in the first model. This process is repeated until the errors are minimized and the dataset is accurately predicted. Boosting algorithms are divided into three main categories, AdaBoost algorithm, Gradient algorithm, and Extreme Gradient Boosting, or XGBoost [2]. Gradient boosting algorithms can be Regressors (for predicting continuous target variables) or Classifiers (predicting categorical target variables). In this paper, XGBoost is used for predicting categorical target variables.

XGBoost stands for eXtreme Gradient Boosting. XGBoost is a powerful machine learning algorithm. it is a supervised learning algorithm used optimized loss function and applied several regularization techniques[2]. XGBoost is a more regularized form of Gradient Boosting. XGBoost uses advanced regularization which improves model generalization capabilities. XGBoost can be used for a variety of applications, including regression and classification problems [f]. XGBoost is a faster algorithm because of its parallel and distributed computing. It has a deep consideration in terms of systems optimization in machine learning [1]. The adoption of XGBoost is mostly due to its execution speed and model performance. It uses ensemble learning methods, which means it combines several different algorithms to produce a single model. This method allows for parallel and distributed processing while maximizing memory utilization. The steps of XGBoost shown in Algorithm 1.

Algorithm 1: XGboost (XB) Algorithm

Input: The n-dimensional data, $X \in R^n$and target outcome,$Y \in \mathcal{R}$

Output: The posterior probability, P [0, 1] of unseen test data, x, where $C = 2$, C_1 (Represent Covid-19)*and* C_2(No Covid-19)

Data: Dataset

Initialize the model with constant value $F_0(X) = argmin\gamma \sum_{i=1} L(Y, \gamma)$ *where,* $\sum_{i=1} L(Y, \gamma)$ *is the differentiable lossfunction and N is the number of sample.*

for $t \leq T(nclassifier)$ **do**

  a) Train aweak learner using distribution D_t.

  b)Select a weak hypothesis,$h_t : R^n \rightarrow R$ *with low weight error,* $\_\xi_t = Pr_i \sim D_i[h_t(x_i) \neq Y]$

Choose $\alpha_t = \frac{1}{2} len \frac{1-\xi_t}{\xi_t}$

Update the Model

LightGBM is a tree-based learning optimization technique. This approach, as an optimization technique, minimizes both information loss and memory space usage. Furthermore, unlike traditional learning tree algorithms that develop at the level of the node with the largest share of information loss, Light GBM accelerates the learning process by

CatBoost, like the Light GBM method, is part of the GBDT model, which seeks to handle categorical features [3], [13], [14]. This approach is frequently used in search, system suggestions, personal help, self-driving automobiles, and weather forecasting. This optimization approach is notable for its data processing speed, which may be up to 60 times quicker than Light GBM in some situations [1].

The main motivation of choosing these models is as following:

- Faster training speed and higher efficiency.
- Lower memory usage.
- Better accuracy.
- Support of parallel and GPU learning.
- Capable of handling large-scale data.

implementations of the Gradient Boosted Trees algorithm, a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques. Gradient boosting is a machine learning technique that can be used for a variety of applications, including regression and classification. It returns a prediction model in the form of an ensemble of weak prediction models, most commonly decision trees. The resulting approach is called gradient-boosted trees when a decision tree is the weak learner; it usually outperforms random forest. A gradient-boosted trees model is constructed in the same stage-wise manner as other boosting approaches, but it differs in that it allows optimization of any differentiable loss function [15], [16].

5. RESULTS

In this section, we present experimental results conducted on the dataset.

5.1. Dataset Description

The experimental data in this paper are derived from dataset developed during a research process obtained from IEDB and UniProt [9].The dataset consists of 15 features, and the result in the data were taken from 14907 people. General statistical information about the dataset shown in Table1.

Table 1. Statistical Information about the Dataset.

Dataset statistics	
Number of variables	15
Number of observations	14907
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.7 MiB
Average record size in memory	120.0 B

Variable types	
NUM	11
CAT	3
BOOL	1

The first few features are identifying pieces for the protein sequences and are not utilized by this paper. These include the parent protein id, protein sequence, peptide start position, peptide end position and peptide sequence. All of these features are dropped in pre-processing because they

are not needed when exclusively examining the protein features. The features and its abbreviation are shown in Table 2.

Table 2. Dataset and Feature Abbreviation.

Features	Abbreviation of Features
parent_protein_id	Parent Protein
protein_seq	parent protein sequence
start_position	start position of peptide
end_position	end position of peptide
peptide_seq	peptide sequence
chou_fasman	peptide feature, β turn
emini	peptide feature, relative surface accessibility
kolaskar_tongaonkar	peptide feature, antigenicity
parker	peptide feature, hydrophobicity
isoelectric_point	protein feature

5.2. Data Correlation

After data pre-processing, we have to learn whether or not the data is processed properly and how much the correlation between the data is. The correlation coefficient can be used to reflect the close relationship between the features. The correlation coefficient is calculated by the difference method. It is also based on the dispersion of the two features and their respective averages. The two differences are multiplied to reflect the degree of correlation between the two features. The linear single correlation coefficient is studied as shown in Fig.1.

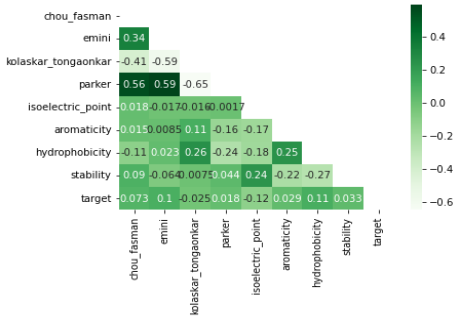

Figure 1. Correlation coefficient matrix

Diving more into the features. we have to learn whether or not the data is processed properly and how much the correlation between the data is. Finding the distribution of each feature is important. It is clear from Fig.2. some variables having skewed distribution. Most of machine learning models assume that the data is normally distributed those variables may need scaling.

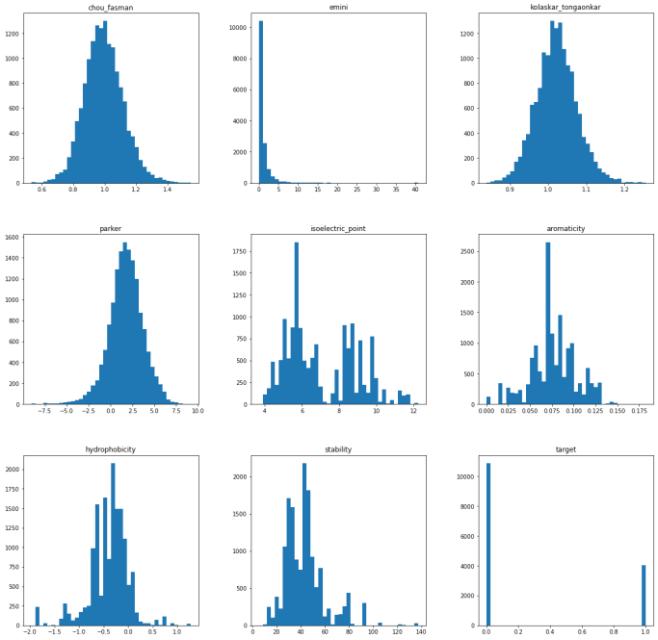

Figure 2. The population distribution of all attributes.

From correlation coefficient matrix, we can figure out whether there is a linear correlation between target and various parameters, such as the emini is 0.34.

5.3. Performance Metrics/Confusion Matrix

XGBoost, **LightGBM** and **CatBoost** Tree algorithms are used in this research work. Experiments are performed using internal cross-validation 10-folds [17], [18]. Accuracy, F-Measure, Recall, Precision and ROC (Receiver Operating Curve) measures are used for the prediction of this work. Table.3.3 defines accuracy measures below.

Table 3. List of Accuracy Measures

Measures	Definitions	Formula
1. Accuracy (A)	Accuracy determines the accuracy of the algorithm in predicting instances.	A=(TP+TN) / (Total no of samples)
2. Precision (P)	Accuracy is measured by Precision.	P = TP / (TP+ FP)
3. Recall (R)	To measure the classifier¢¢s completeness or sensitivity, Recall is used.	R =TP / (TP+FN)
4. F-Measure	F-Measure is the weighted average of precision and recall.	F=2*(P*R) / (P+R)
5. ROC	ROC (Receiver Operating Curve) curves are used to compare the usefulness of tests.	

5.4. Result Analysis

By the iterated training and adjustment parameters of the training set, the comparison results are showed between XGBoost, CatBoost, and LightGbM in Table.4.

Table 4. Performance comparison for dataset.

Prediction Models	Precision	Recall	Accuracy%
XGBoost	0.708	0.532	81.83
CatBoost	0.737	0.514	82.01
LightGBM	0.720	0.531	81.83

While carrying out B-cell epitope prediction, the model can also give the order of importance feature for improving the accuracy of the prediction model through the tree model mechanism. These important features can also be used as some clinical reference value for doctors.

Finally, the results of an attribute in all the models are weighted and summed, and then averaged to obtain the importance score.

As shown in Fig.3.[19]

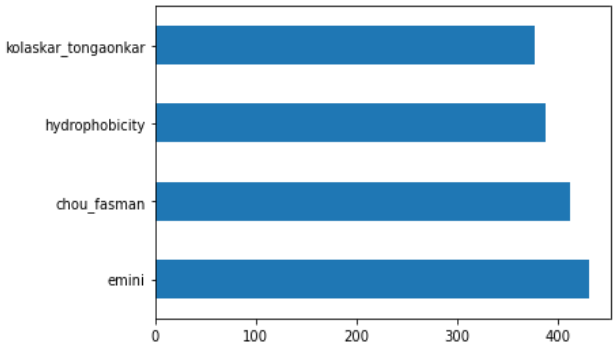

Figure 3. Feature importance of the Machine Learning Algorithm

As can be seen from Fig.3. we can clearly see that feature (peptide) is the most important feature which has contributed towards the prediction of the results. Followed by feature (kolaskar_tongaonkar) and feature (emini). The least important feature is feature (antigenicity). The model can also give the order of importance feature for improving the accuracy of the prediction model through the tree model mechanism. These important features can also be used as some clinical reference value for doctors.

6. CONCLUSION

Data preparation is a necessary step in ensuring model accuracy and speeding up the process. When the models are compared, it is evident that CatBoost outperforms several classic approaches. We suggested a better feature combination approach based on CatBoost after comparing it to the integrated algorithms. The experiment findings suggest that using the CatBoost model, it is possible to develop a B-cell epitope prediction model with high prediction accuracy, stability, and speed.However, there were no missing data in the datasets used in this experiment. As a result, the same conclusions cannot be predicted from a dataset with missing values and noisy data. As a result, we will test the accuracy and speed of the model's using datasets that include a variety of data types as well as missing data in the future. In addition, as we are entering the BigData era, we would attempt running the models in-memory distributed computing and design UGI for it to make it easy to use.

REFERENCES

[1] R. B. Sundaram, "Gradient Boosting Algorithm: A Complete Guide for Beginners," analyticsvidhya, 2021.

[2] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[3] G. Huang et al., "Evaluation of CatBoost method for prediction of reference evapotranspiration in humid regions," J. Hydrol., vol. 574, pp. 1029–1041, 2019.

[4] J. Chen, H. Liu, J. Yang, and K.-C. Chou, "Prediction of linear B-cell epitopes using amino acid pair antigenicity scale," Amino Acids, vol. 33, no. 3, pp. 423–428, 2007.

[5] T. Liu, K. Shi, and W. Li, "Deep learning methods improve linear B-cell epitope prediction," BioData Min., vol. 13, no. 1, pp. 1–13, 2020.

[6] M. Park, S. Seo, E. Park, and J. Kim, "EpiBERTope: a sequence-based pre-trained BERT model improves linear and structural epitope prediction by learning long-distance protein interactions effectively," bioRxiv, 2022.

[7] J. Faouzi and H. Janati, "pyts: A Python Package for Time Series Classification.," J Mach Learn Res, vol. 21, pp. 46–1, 2020.

[8] J. Hao and T. K. Ho, "Machine learning made easy: a review of scikit-learn package in python programming language," J. Educ. Behav. Stat., vol. 44, no. 3, pp. 348–361, 2019.

[9] R. Vita et al., "The immune epitope database (IEDB): 2018 update," Nucleic Acids Res., vol. 47, no. D1, pp. D339– D343, 2019.

[10] R. E. Schapire, "The boosting approach to machine learning: An overview," Nonlinear Estim. Classif., pp. 149–171, 2003.

[11] K. M. Ting, "A comparative study of cost-sensitive boosting algorithms," 2000.

[12] A. J. Ferreira and M. A. Figueiredo, "Boosting algorithms: A review of methods, theory, and applications," Ensemble Mach. Learn., pp. 35–85, 2012.

[13] A. V. Dorogush, V. Ershov, and A. Gulin, "CatBoost: gradient boosting with categorical features support," ArXiv Prepr. ArXiv181011363, 2018.

[14] J. T. Hancock and T. M. Khoshgoftaar, "CatBoost for big data: an interdisciplinary review," J. Big Data, vol. 7, no. 1, pp. 1–45, 2020.

[15] S. Neelakandan and D. Paulraj, "A gradient boosted decision tree-based sentiment classification of twitter data," Int. J. Wavelets Multiresolution Inf. Process., vol. 18, no. 04, p. 2050027, 2020.

[16] T. Pinto, I. Praça, Z. Vale, and J. Silva, "Ensemble learning for electricity consumption forecasting in office buildings," Neurocomputing, vol. 423, pp. 747–755, 2021.

[17] S. De Bruyne and F. Plastria, "2-class Internal Cross-validation Pruned Eigen Transformation Classification Trees," Optim. Online Httpwww Optim. OrgDB HTML2008051971 Html.

[18] H. Van Hasselt, "Estimating the maximum expected value: an analysis of (nested) cross validation and the maximum sample average," ArXiv Prepr. ArXiv13027175, 2013.

[19] F. Nafa, A. Babour, and A. Melton, "Prerequisite Relations among Knowledge Units: A Case Study of Computer Science Domain," Comput. Model. Eng. Sci., doi: 10.32604/cmes.2022.020084.