Machine Learning Models of B-cell and T-cell Epitopes Using Sequence and Structure Information

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

by

Kiran K. Sewsankar
Bachelor of Science
Worcester State University, 2015

Director: Iosif Vaisman, Professor
Department of Bioinformatics and Computational Biology

Spring Semester 2022
George Mason University
Fairfax, VA

DEDICATION

This is dedicated to all my teachers/professors, colleagues, friends, and family that have helped me accomplish this incredible feat.

ACKNOWLEDGEMENTS

I would like to thank Dr. Vaisman, Dr. Klimov, and Dr. Luchini for serving on my committee, and for their guidance throughout this journey, Dr. Majid Masso, Krista Smith, and Shengyuan Wang for their technical support and sharing of scripts, Chris Ryan and Ford Combs, for their web server assistance, and finally, the authors of DiscoTope, for their dataset and epitope annotations.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

MACHINE LEARNING MODELS OF B-CELL AND T-CELL EPITOPES USING SEQUENCE AND STRUCTURE INFORMATION

Kiran K. Sewsankar, Ph.D.

George Mason University, 2022

Dissertation Director: Dr. Iosif Vaisman

Epitopes, the regions of antigens that are detected by the immune system, have garnered considerable scientific interest in recent years due to their potential for influencing the development of novel medical countermeasures, an example of which, are epitope-based vaccines that can be both safer and more efficacious than those currently available. Innovative vaccines are profoundly needed to keep pace with the ever-changing landscape of global infectious disease. The drive towards creating new vaccines and treatments is critical to preserving world health and will be aided by studying epitopes. Epitopes of the antigen play an important role in immune response. For example, they are recognized by B-cells and T-cells and are the site of antibody binding. Therefore, identifying epitopes can help researchers better understand how foreign disease agents cause illness and how the host immune system reacts against it. Traditional

methods for the identification of epitopes centered around experimental structural studies including, X-ray crystallography and NMR techniques, which are time consuming and costly. Thus, bioinformatics and computational approaches have been explored to facilitate the epitope identification process. In this work, models of B-cell and T-cell epitopes were developed to assist in the prediction and classification of non-validated but potential epitope protein sequences. Specifically, machine learning algorithms were trained on a diverse set of epitope/non-epitope representative feature vectors, comprised of sequence derived features based on reduced amino acid alphabets and n-grams and structure derived features based on Delaunay tessellation and amino acid propensity scores to reliably predict epitope sequences and residues. Feature vectors were constructed based on the specific problem at hand, either linear or conformational epitope prediction. The epitope sequence and structure data were obtained from publicly available databases and several machine learning algorithms, including Random Forest, Gaussian Naïve Bayes, and Support Vector Machine were applied to the descriptor space. The best performing epitope prediction models trained here can be used to identify unknown epitope sequences or residues, consequently reducing the search space for candidate epitopes, epitopes that will be the basis for the development of new vaccines and other medical countermeasures. The models are incorporated into the TESSETOPE V1.0, which is a freely available web accessible API for epitope prediction available at http://omics.gmu.edu/tessetope/.

CHAPTER ONE

1.1 Epitopes: An Introduction

The primary job of the immune system is to respond to and fight disease. To accomplish this, the immune system of the host must be able to distinguish between different types of threats and determine the optimal response that will aid in its destruction. Particularly, the adaptive immune system is the division of the immune system that is tasked with recognizing specific threats and is a key driver of long-lasting immunity and rapid immune response [1]. Two types of immune response, humoral immunity and cell-mediated immunity play major roles in the success of the immune system [1]. For the framework of this dissertation research, it will be important to focus on both of these types of immunity. Specifically, humoral immunity, and its relation to B-cells and antibody production and, cell-mediated immunity, and its relation to T-cells and major histocompatibility complex (MHC) proteins.

Epitopes are critical to the discussion of humoral and cell-mediated immunity. Epitopes are the regions of antigens that are detected by the immune system and are able to elicit the aforementioned immune responses, either humoral or cell-mediated [2]. Consequently, making them a key topic of interest for researchers around the world [3]. Epitopes are important because they can be implicated in vaccine design and in the prevention, diagnosis, and treatment of disease [3].

To better understand the importance of epitopes and how they relate to humoral and cell-mediated immunity, it will be imperative to discuss two of the major epitope types, B-cell and T-cell. B-cell epitopes are the regions of antigens that are recognized by B-cells. Specifically, B-cells recognize antigens using B-cell receptors, which initiates the secretion of antibodies that bind and destroy the antigens, all part of humoral immunity [1]. Antibodies are fundamental components of the acquired immune response in vertebrates [4], [5]. Furthermore, antibodies are critical proteins that are used to detect and fight foreign pathogens that invade the host, such as bacteria, parasites, and viruses [4], [5]. The antigen binding site of the antibody is of great importance for studies examining antibody-antigen (Ab-Ag) interactions. This site contains three hypervariable regions, composed of mainly complementarity-determining region (CDR) loops in both the heavy and light chains, which allows the antibody to detect a plethora of antigens [4]–[6]. Thus, the Ab-Ag complex is extremely critical for the humoral immune response against pathogen invaders [7], [8]. B-cell epitopes, and their counterparts, paratopes, are quite important in the interactions of the antibody-antigen (Ab-Ag) complex [7]. Simply put, the Ab-Ag complex can be described by an interaction between the epitope and paratope, the antibody's paratope binds to the antigen's epitope [7]. The paratope owns a structural and chemical makeup that complements that of the B-cell epitope, allowing them to interact [9]. This knowledge itself gives extreme relevance to B-cell epitopes, as they are major players in the Ab-Ag complex, and it makes their identification significant for the development of vaccines and drugs and to the understanding of disease [7].

T-cell epitopes on the other hand, important for cell-mediated immunity, should be studied in the context of human leukocyte antigen (HLA) system genes and MHC proteins. T-cells are able to recognize antigens that are presented on HLA [1]. The HLA class I pathway can process intra-cellular antigens, while the HLA class II pathway can process extra-cellular antigens [1]. It has been well understood that T-cell epitopes are bound to MHCs linearly and that the epitopes, through R group side chain interactions, link together into the binding groove of both MHC Class I and Class II molecules [2]. Like antibodies, the structural and chemical makeup of the CDRs of T-cell receptors (TCRs) govern T-cell epitope binding [10]. T-cell epitopes are also quite important for the immunogenicity of certain foreign peptides that enter the host. T-cell epitopes enable the generation of MHC II-peptide-T cell receptor complexes, commencing a signaling cascade that stimulates helper T-cells, B-cells, and antibodies to eliminate foreign peptides [11]. Notably, the foreign peptides discussed here may be those that are a part of a biotherapeutic that is designed to treat a disease, but are unwantedly targeted by the immune system, producing undesired side effects and compromising drug efficacy [12]. Consequently, T-cell epitopes are the chief targets of protein deimmunization techniques, techniques which look to mutate the important residues in the epitopes of the biotherapeutics, in hopes of reducing immunogenicity [11]. An expanded discussion of this topic will be given in section 1.3.2.

1.2 Epitope Identification and Prediction

Because of the role epitopes play in immunogenicity and the fact that they are staple components of many disease-causing agents, their identification is critical to the

development of new and innovative ways to treat disease such as, epitope-based vaccines. Traditional methods for the identification of epitopes principally centered around experimental structural studies, such as X-ray crystallography and NMR [13]. These experimental approaches involve solving the 3D structure of antigen-antibody or antigen-T-cell complexes and determining amino acids in contact with each other allowing for the determination of the epitope [9]. Though accurate, these techniques are rather expensive and tedious in their protocols [13]. Consequently, researchers have been searching for ways to make the process of epitope prediction much faster and cheaper [13]. Bioinformatics, the scientific field that can be best described as a melting pot of biology, computing, mathematics, and information technology [2] has risen up to take on this task, using the computational epitope prediction approach. Bioinformatics and its plethora of computational tools can be applied to the *in silico* prediction of epitopes, which dramatically reduce the time and money spent on the task of epitope identification [13]. The main goal of these efforts into predicting epitopes is to reduce the search space of candidate epitopes, epitopes that may be useful for developing therapies, making the development process more efficient.

Epitope prediction via bioinformatics' computational tools have been primarily focused on certain types of epitopes. To date, a majority of the epitope predictors have been developed to predict either MHC-I or MHC-II binders, T-cell binders, or B-cell binders [2]. Moreover, two main classes dominate epitope prediction, linear (continuous) and conformational (discontinuous) [3]. Linear epitopes are epitopes whose residues are continuous in the protein sequence and are typically amphipathic in nature and are

peptide fragments that are 9-12 mers [3]. Conformational epitopes are epitopes whose residues are not continuous in the protein sequence but are moved into structural proximity due to protein folding and are typically 15-22 mers [3].

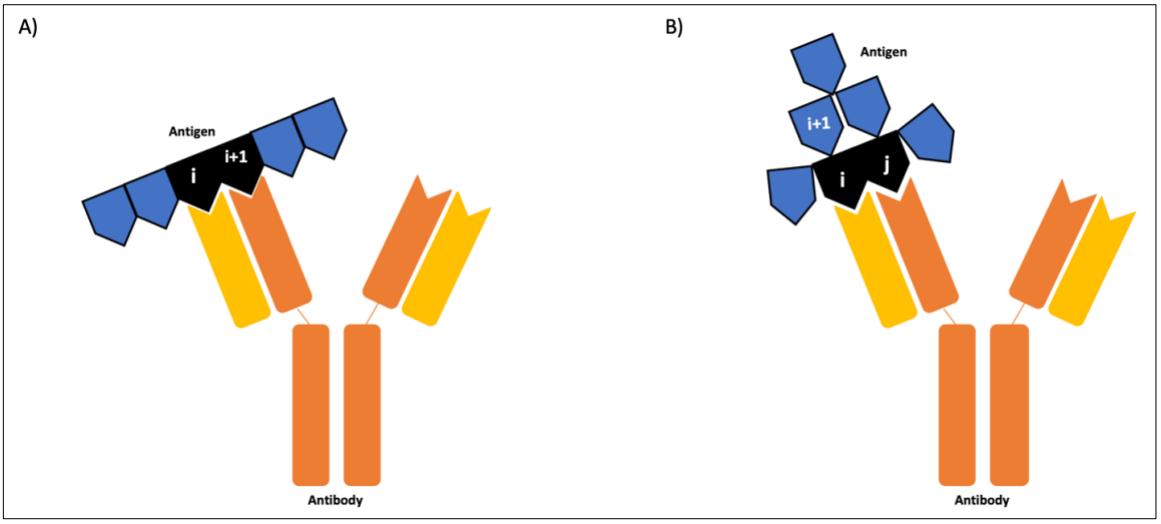


Figure 1 Linear vs. Conformational Epitopes A) Example of a Linear Epitope (LE) where the epitope (black) consists of a continuous residue segment (i and i + 1). B) Example of a Conformational Epitope (CE) where the epitope segment (black) is not continuous in the antigen sequence (j – i > 1).

It has been noted that approximately more than 90% of epitopes are conformational in nature (mainly B-cell epitopes), leaving less than 10% in the linear category (B-cell and T-cell epitopes) [14]. However, as described above both epitope categories are important players in immune response and are worthy of further exploration. Moreover, given the growing amount of antigen sequence and antigen three-dimensional structural data readily available today, there is increasing promise for the development of reliable *in silico* epitope prediction tools. Therefore, this dissertation will

explore state-of-the-art methods for epitope prediction and utilize novel approaches to generate well-performing, easy to use, and widely available prediction tools.

1.2.1 State-of-the-Art Computational Tools for Epitope Prediction

The computational experiments conducted in this dissertation will focus on two main types of epitopes, linear T-cell/B-cell epitopes and conformational B-cell epitopes. Tables 1 and 2 summarize the current state-of-the-art in bioinformatics-based epitope prediction tools.

Table 1 State-of-the-Art in T-cell Epitope Prediction. List of T-cell epitope prediction tools, along with a description of the tool, approximate accuracy, AUC, or Pearson's Correlation Coefficient, and reference.

Tool	Description	Method Quality	Reference
EpiJen	Multi-step algorithm	80% (accuracy)	[15]
BIMAS	Published coefficient tables	N/A	[16]
ProPred I	Quantitative matrix	38-80% (accuracy)	[17]
ProPred	Quantitative matrix	N/A	[18]
MHCPred	Additive method	N/A	[19]
NetMHC	ANN based method	0.761-0.912 (Pearson's Correlation Coefficient)	[20][21]
RANKPEP	PSSM	0.5-0.9 (AUC)	[22]
SVMHC	SVM-based method	N/A	[23]
NetCTL	ANN-regression	0.9 (AUC)	[24]
nHLAPred	Artificial Neural Networks	92.8% (accuracy)	[25]

Table 2 State-of-the-Art in B-cell Epitope Prediction. List of B-cell epitope prediction tools, along with a description of the tool, method quality (approximate accuracy, AUC, Pearson's Correlation Coefficient, etc.) and reference.

Tool	Description	Method Quality	Reference
ABCpred	Recurrent artificial neural networks	65.93% (accuracy)	[26]
BCPRED	Support vector machine with string kernels	0.758 (AUC)	[27]
FBCPred	Support vector machine with string kernels	73.37% (accuracy)	[28]
BepiPred2.0	Hydrophilicity scale combined with hidden markov model	0.62 (AUC)	[29]

COBEpro	SVM, prediction score	0.628-0.829 (AUC)	[30]
LBtope	Input FATSA	54-86% (accuracy)	[31]
Bcepred	Continuous, Combines hydrophilicity, flexibility, polarity, and exposed surface	58.7% (accuracy)	[32]
SVMTriP	Employs support vector machine to combine tripeptide similarity and propensity scores	0.702 (AUC)	[33]
DiscoTope	Amino acid stats, spatial context, surface accessibility of aa	0.711 (AUC)	[34]
BePro (PEPITO)	Weighted linear combination of amino acid propensity scores and half sphere exposure values	68.3-75.4 (AUC)	[35]
ElliPro	Approximates a protein surface patch by an ellipsoid, Protrusion index	0.732 (AUC)	[36]
SEPPA	Unit patch of residue triangle	0.742 (AUC)	[37]
EPITOPIA	Based on Naïve Bayes classifier with physiochemical and structural geometrical properties	70–90% (accuracy)	[38]
CBTOPE	SVM based predictor combines physiochemical profiles and sequence-derived inputs, Antigen primary structure	86.59% (accuracy)	[39]
EPCES	Consensus score by six functions	N/A	[40]
EPSVR	Based on SVR and meta-analysis	0.597 (AUC)	[41]
PEASE	Evaluates a pair score for all combinations of one residue from CDR of antibody and one form surface exposed region of antigen	N/A	[42]
EpiPred	Combines conformational matching of the Ab-Ag structures and knowledge based asymmetric Ab-Ag scoring	44% (Recall) 14% (Precision)	[5]
EpiSearch	Mimotope-based prediction, An automated sequence analysis based on sequence and 3D profiles	N/A	[43]
MIMOX	Mimotope-based prediction, First free web tool for it	N/A	[44]
PepSurf	Mimotope analysis: surface graph	N/A	[45]
CEP	Accessibility of residues and spatial distance cut-off (surface accessible)	75% (accuracy)	[46]
Rapberger	Based on antibody info	N/A	[47]
PEPOP	Accessible and sequence contiguous amino acids segments	N/A	[7]

Shinji Soga	Antibody-specific epitope propensity index	N/A	[48]
EPMeta	A meta server, which combines EPSVR, EPCES, EPITOPIA, SEPPA, PEPITO, and Discotope 1.2	0.638 (AUC)	[41]
Zhang	Based on random forests with a distance-based feature	65-70% (accuracy) 0.633 (AUC)	[49]

Methods for the prediction of epitopes produce significant tools that can be used by scientific researchers around the world to accelerate the production of novel medical countermeasures. Computational approaches have made the development process faster and cheaper and easy to use computational tools can be ran by experienced scientists and novelists alike. Though great strides have been made in this field, many improvements can be made to increase performance and reliability. This is evidenced by the method quality values in Tables 1 and 2. Not only will improved epitope prediction techniques lead to a better understanding of how epitopes are implicated in immune response, but they can also be the basis for continued medical advancement. A few examples of real-world applications of epitope prediction tools are discussed in section 1.3.

1.3 Real-world Applications of Epitope Prediction Tools

1.3.1 Vaccine Development

Vaccine development has been a tremendous success for the fight against disease over the years [2]. The most efficacious vaccines to date have been based on protein subunits or inactivated or attenuated whole microbes [50]. However, the process of vaccine development can take up to 15 years to complete and some vaccines can cause adverse effects to the patient such as, onset of the disease or death [2]. The need for new

vaccines is omnipresent, the need ascends from the rapidly evolving landscape of disease [51]. Bioinformatics has made it possible to generate new vaccines using computer *in silico* predictions that dramatically reduce the time spent on this task and produce safer and more efficacious vaccines [2]. One of the ways bioinformatics can achieve this type of success is through a deep understanding of epitopes, vital epitopes that induce immune responses [2]. Once again, epitopes are defined by their ability to bind to antibodies or other peptides sent in an immune response, therefore they are prime targets for vaccine development. In simple terms, the approach is to isolate epitopes that are known to elicit an immune response, such as antibody production, and replace the whole pathogen in a vaccine with these epitopes [2], the search for these epitopes can be simplified using computational methods. Therefore, epitope prediction and the identification of epitopes that induce immune responses can be used in the development of vaccines that can treat disease.

Recently, the coronavirus disease 2019 (COVID-19) pandemic caused by a single-stranded RNA virus, severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), has wreaked havoc across the globe causing severe illness and death [52], [53]. The spike protein of SARS-CoV-2 assists receptor binding and viral entry within the host making it a prime target for vaccine development [53]. Computational epitope prediction tools can be used in cases like this to identify potential immunogenic epitopes on the spike protein to develop multi-epitope vaccine constructs [53]. Epitope-based vaccines have many advantages over traditional vaccines, such as low-cost production, ease of modification, and safety [52]. These computational approaches can be used in the future

to continue to push the vaccine development process forward, allowing for faster development times and more efficacy.

1.3.2 Protein Biotherapeutic Deimmunization

One of the major drawbacks of current protein drugs or so called "biotherapeutics" is the immunogenicity that they often elicit [54]. The problem with this immunogenicity is that it can cause the body to produce anti-drug antibodies that may interfere with the efficacy of the administered drug and even cause harmful side effects for the patient [54]. Therefore, it is critical for the development of efficacious and safe protein-based drugs to understand why these drugs produce immunogenic responses that negatively impact their ability to treat disease and what can be done to counteract this immunogenicity. Methods developed to reduce immunogenicity but maintain the structure and function of the drug will be advantageous for researchers who are searching to create the most effective drugs that can treat common diseases impacting the world today. T-cell epitopes, which are major components of these biotherapeutics are chief determinants of their immune response [54]. Consequently, altering the sequences of these epitopes through sequence mutations will reduce their immunogenicity [54]. However, it is extremely vital that the sequence alterations are chosen, as such, to minimize any structural or functional impact on the protein biotherapeutic [54]. The scientific problem of protein biotherapeutic deimmunization is multi-phase in nature. First, the challenge is to deimmunize the biotherapeutic via alteration of T-cell epitopes through selected sequence point mutations. The second phase entails maintaining the structure and function of the protein drug after the sequence alterations. Computational

tools for the prediction of T-cell epitopes can help address this complex problem by finding these epitopes and helping to target the specific deimmunizing mutations.

1.3.3 Immune Checkpoint Therapy

Epitope prediction can be applied to another very important scientific/medical problem, that of cancer therapy. Specifically, epitope prediction can be implicated in Immune Checkpoint Therapy. In general, the immune system is a large determining factor for the fate of developing cancers [55]. The immune system can function as a tumor promoter, promoting tumor growth, and shape tumor cell immunogenicity [55]. Moreover, the immune system can function as a tumor suppressor, destroying tumor cells or slowing their growth [55]. This idea of the immune system functioning as a tumor suppressor can be probed to yield a very innovative form of cancer therapy. It is known that cancer induced immunosuppression can be mediated, in many individuals, by two immunomodulatory receptors that are expressed on T cells, Cytotoxic T-Lymphocyte Associated Antigen-4 (CTLA-4) and Programmed Death-1 (PD-1) [55]. Monoclonal antibody (mAb) therapies that target and block CTLA-4 and/or PD-1 have shown to produce many benefits in the clinical setting, because of their ability to allow the immune system to function without fallen victim to these checkpoints [55]. Because individual cancers have been shown to contain many mutant genes compared to normal tissues, it will be well served to study the cancer epitope landscape [56]. *In silico* epitope prediction can be applied to the identification of candidate tumor antigens, in simple terms, if we know the epitopes of the tumor antigens then we may understand how the immune system can attack it. The ultimate goal of the epitope prediction approaches applied

towards cancer immune checkpoint therapy is to design cancer specific vaccines based on the predicted epitopes for the tumor antigens [55],[56].

1.4 Motivation for this Dissertation Work

It is clear that the topic of epitope prediction is a significant one, for not only the scientific community, but the general public as well. It carries the potential to revolutionize the design and production of medical treatments and therapeutics. Also, with the current advancements in technology and growing databases storing antigen information, the door is open to improve upon the current state-of-the-art in the epitope prediction field. Taken together, this is exactly the motivation for exploring this topic in this dissertation. Here, epitope prediction has been approached using novel strategies, in hopes of producing interesting and significant results. The results can be applied to several applications throughout the world of immunology and will be important for the advancement of scientific research as a whole.

1.5 Summary of the Work Presented in this Dissertation

Chapter 2 describes methods for training machine learning models on strictly protein sequence-derived feature sets to predict B-cell and T-cell epitope sequences. The methods use linear B-cell and T-cell epitope sequences, non-epitope sequences, reduced amino acid alphabets (3-letter schemes), and n-grams (3-letter). Various machine learning algorithms are tested and model performance is evaluated to find the best performing model.

Chapter 3 describes methods for training machine learning models on strictly protein structure-derived feature sets to predict B-cell epitope residues. The methods use

conformational B-cell epitope sequences, non-epitope sequences, Delaunay tessellation, and amino acid propensity scores. Various machine learning algorithms are tested and model performance is evaluated to find the best performing model.

Chapter 4 describes methods for training machine learning models on protein sequence and structure-derived feature sets to predict B-cell epitope residues. The methods use conformational B-cell epitope sequences, non-epitope sequences, Delaunay tessellation, and amino acid propensity scores/identities. Various machine learning algorithms are tested and model performance is evaluated to find the best performing model.

Chapter 5 describes the TESSETOPE V1.0 web API. Directions for use along with examples are provided.

Chapter 6 describes conclusions and future directions.

CHAPTER TWO

2.1 Sequence-based Epitope Prediction: An Overview

This chapter describes machine learning experiments that were performed to generate models of linear B-cell and linear T-cell epitopes using strictly sequence derived feature sets. Models trained on these particular feature sets are desired due to the wide availability of antigen amino acid sequence data. The computational approaches described herein, were used to train accurate and efficient models that can predict linear epitope sequences in any unseen test datasets.

2.2 Introduction to the Sequence-based Approach for Linear Epitope Prediction (B-cell/T-cell)

A sequence-based method was employed to train machine learning models that have the ability to either predict linear B-cell or linear T-cell epitope sequences. These two types of epitopes were chosen as the basis of this chapter because tools falling into these prediction categories dominate the current literature on epitope prediction and are the focus of epitope-based vaccine design, protein deimmunization, and many other medical applications. The principal question investigated here was as follows, can a machine learning model trained solely on sequence derived features from experimentally determined epitopes and non-epitopes, using techniques such as reduced amino acid alphabets and n-grams, successfully predict linear B-cell or linear T-cell epitopes at

higher or comparable levels of performance than what is currently available today? Developing a totally sequence-based prediction tool is quite appropriate because epitope sequences are much more widely available than solved epitope three-dimensional structures. Consequently, more sequence data is accessible to researchers and more data will be produced in the future. This will allow for the continued rapid development of prediction tools which can be used in vaccine design and protein deimmunization, as mentioned previously. Additionally, these machine learning models are easier to apply to unseen data in practice, requiring only sequence information to run the model.

The sequence-based approach uses the techniques of alphabet reduction and n-grams to generate useful features needed by the machine learning algorithms to train the predictive models. Alphabet reduction is simply the clustering of the twenty amino acids into a smaller number of groups based on properties and/or similarities [57]. The goal of alphabet reduction is to reduce the compositional complexity of the sequence, without losing important biochemical information, for faster and more efficient machine learning [58]. An n-gram is a subsequence of "n" consecutive characters within a sequence, n-gram sliding windows of size "n" can be used to determine the attributes of each sequence based on n-gram frequencies/probabilities (see section 2.3.1) [59]. Altogether, this approach looks to represent a linear epitope sequence as a string of reduced alphabet characters, where the sequence characteristics can be discovered by examining its n-gram probabilities and these probabilities can be used by machine learning algorithms to train the predictive models.

2.3 Linear B-cell Epitope Prediction (Sequence-based)

2.3.1 Normalized n-gram Probability Approach

The first step in generating the predictive models is to construct a training dataset. The training set must contain both positive (linear B-cell epitope) and negative (non-epitope) instances to allow for proper training of machine learning models. The creation of the positive set (validated linear B-cell epitope sequences) was done by downloading the non-redundant B-cell epitope dataset from the Bcipep database [60], available from, http://crdd.osdd.net/raghava/bcipep/index.html. The Bcipep database contains information on experimentally verified linear B-cell epitopes having varying immunogenicity [60]. The database contains thousands of epitope entries, including immunodominant, immunogenic, and null-immunogenic epitopes and the epitopes originate from a variety of pathogens, such as viruses, bacteria, protozoa, and fungi [60]. Figure 2 and Table 3 examine the distribution of sequence lengths in the positive dataset.

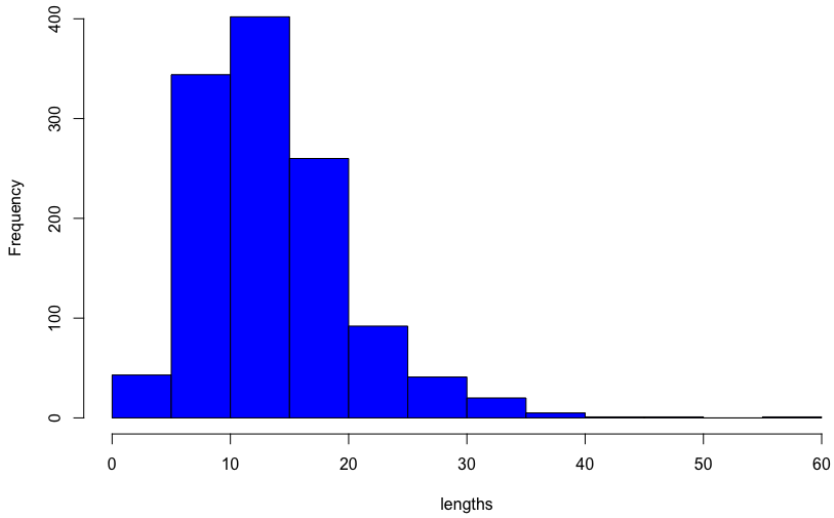


Figure 2 Length distribution of the experimentally validated linear B-cell epitopes from the Bcipep database in our positive dataset.

Table 3 Frequency distribution of the lengths of the linear B-cell epitopes in the positive dataset

Scores	Frequency
(0,5]	43
(5,10]	344
(10,15]	402
(15,20]	260
(20,25]	92
(25,30]	41
(30,35]	20
(35,40]	5
(40,45]	1
(45,50]	1
(50,55]	0
(55,60]	1

About 87% of the epitopes in the Bcipep set have a length $\leq$ 20 amino acid residues and about 65% of the epitopes have a length $\leq$ 15 amino acid residues. This data suggests linear B-cell epitope sequences tend to be shorter rather than longer peptides. A

more detailed look at epitope lengths and their impact on the predictive performance of trained machine learning models will follow in the next section.

To complete the creation of the training set, a Python script was written to extract the epitope sequences from the Bcipep data file and insert them into a MySQL database table. The negative set (non-epitopes) was created using Python scripts that take in all protein sequences from the Protein Data Bank (PDB) [61], available from, http://www.rcsb.org/, inserts them into a MySQL database table, randomizes the sequences, and cuts out sequence fragments, starting at random points in the protein sequence, correlating to the size of its corresponding entry in the positive set. Therefore, the positive and negative sets within the overall training set had equivalent numbers of sequences of equal lengths. For example, if the first sequence in the positive set had a length of 12 amino acid residues, the code creating the negative set would cut out a sequence fragment of length 12 starting at a random point in one of the sequences from the PDB, this fragment would then be inserted into the first slot of the negative set. Table 4 shows the size of both the positive and negative sets for four different training sets, with four different epitope length restrictions used for this approach.

Table 4 Size (number of sequences) of the positive and negative sets used for this sequence-based linear B-cell epitope prediction approach.

Training Set	Epitope Size Restriction (residues)	Number of Sequences in Positive Set	Number of Sequences in Negative Set
1	None	1210	1210
2	12+	774	774
3	14+	643	643
4	19+	328	328

Creating four separate training sets, each with differing sequence length cutoffs for the positive set, was done to find the optimal epitope length and training set size for this machine learning approach. For example, setting the length threshold to include shorter sequences, say of length less than 9, will dramatically increase the size of the positive set, but these short sequences may not provide enough information to the machine learning algorithms and simply complicate matters. On the other hand, setting the sequence length threshold too high will reduce the size of the set beyond usable levels. Therefore, the optimal balance was searched for to create the best possible positive set, yielding satisfying prediction results.

After the construction of the training sets, including both positive and negative instances, data cleaning was performed to ensure non-redundancy in the sets and unambiguous amino acid declarations. Then, alphabet reduction was performed on all sequences. The alphabet reduction script was previously developed in the Python programming language by members of Dr. Vaisman's group at George Mason University and was modified for use in this work. In total, five unique 3-letter (B, J, U) reduced amino acid alphabet schemes were implemented with this script [Table 5], and the sequences produced using the reduced alphabets were inserted into the tables denoted positive and negative set in the MySQL database.

Table 5 Reduced amino acid alphabet schemes used for feature extraction. Columns B, J, and U display the single letter amino acid codes for the amino acids grouped to that letter.

Reduced Alphabet	B	J	U	Reference
1	C, M, F, I, L, V, W, Y	A, T, H, G, P, R	D, E, S, N, Q, K	[62]
2	C, M, F, I, L, V, W, Y	G, P, A, T, S	E, K, R, D, N, Q, H	[57]
3	A, V, F, I, L, P, M, G	D, E, K, R	S, T, Y, C, N, Q, H, W	[63]
4	M, H, V, Y, N, D, I	Q, L, E, K, F	W, P, R, G, S, A, T, C	[64]
5	L, A, S, G, V, T, I, P, M, C	E, K, R, D, N, Q, H	F, Y, W	[65]

Subsequently, the n-gram procedure was ready to be applied to all reduced alphabet sequences to generate relevant sequence-based features to train machine learning algorithms. The n-gram Python script was also previously developed by members of Dr. Vaisman's group and modified for use in this dissertation research. The script calculates the normalized n-gram probabilities for each of the 27-possible n-grams (n-gram of size 3/ alphabet of size 3) for each sequence created using the reduced alphabets and inserts the probabilities into the MySQL database tables storing the training sets. Equation 1 describes the ngram probability and Equation 2 describes the normalized ngram probability.

$$P = \frac{f_n}{n} \tag{1}$$

$$NP = \frac{P}{(f_a * f_b * f_c)} \quad (2)$$

$P = ngram\ probability$

$f_n = frequency\ of\ ngram\ in\ sequence$

$n = total\ number\ of\ ngrams\ in\ sequence$

$NP = normalized\ ngram\ probability$

$f_a = frequency\ of\ 1st\ letter\ in\ ngram\ in\ the\ sequence$

$f_b = frequency\ of\ 2nd\ letter\ in\ ngram\ in\ the\ sequence$

$f_c = frequency\ of\ 3rd\ letter\ in\ ngram\ in\ the\ sequence$

For a traditional 20-letter amino acid alphabet, there are 20^n distinct n-grams of size n. For this approach, an n-gram size of 3 was chosen because it has been previously proven to be optimal for work with short peptide-length sequences. Thus, if a 20-letter alphabet was to be used here there would be 8,000 distinct n-grams of size 3. Alphabet reduction reduces this space dramatically by creating a 3-letter amino acid alphabet instead of a 20-letter alphabet, because $3^3 = 27$ distinct n-grams of size 3. This reduction allows for patterns to be seen more clearly and machine learning to work more effectively. The normalized n-gram probabilities were used as features to represent the epitope and non-epitope sequences and to train the machine learning algorithms. After the n-gram script was run and probabilities were determined, class labels, 1 = positive (epitope) and 0 =

negative (non-epitope), were added to the last ‘class’ column of the training sets for each sequence for binary classification purposes.

Finally, the training sets for each alphabet was used to train several machine learning algorithms. Supervised machine learning was performed using the scikit-learn Python module [66]. Stratified 10-fold cross-validation results and model performance metrics are summarized in section 2.3.2. Stratified cross-validation was used to preserve class percentage in each fold.

Figure 3 summarizes the entire sequence-based linear B-cell epitope prediction workflow.

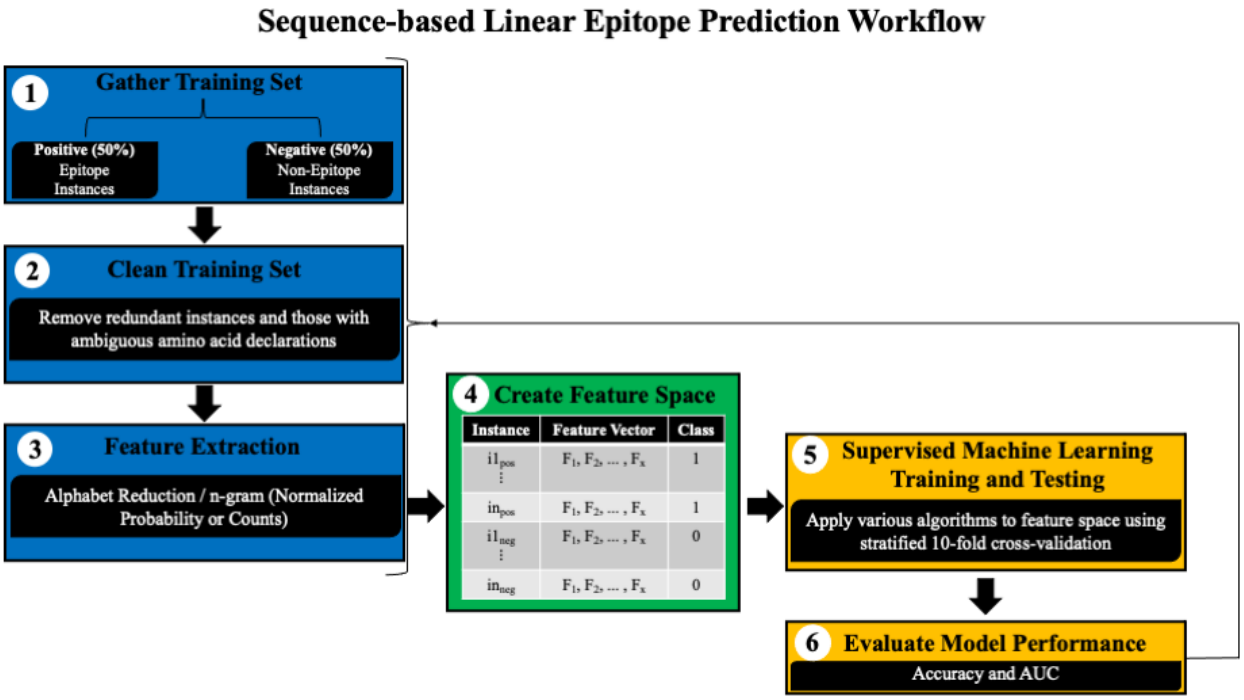


Figure 3 Sequence-based linear epitope prediction workflow. The iterative workflow follows six sequential steps, with a return to either steps 1, 2, or 3 after model performance evaluation. Notation for step 4, *i = instance, pos = positive, n = total number of instances for that particular subset, neg = negative, F = feature, x = total number of features in vector*.

2.3.2 Linear B-cell Epitope Prediction Results

The supervised machine learning approach to develop models trained on protein sequence-derived feature sets, using reduced amino acid alphabets and n-grams, was used to classify B-cell epitope and non-B-cell epitope sequences. Tables 6-9 display model performance evaluations for several machine learning algorithms trained on different linear B-cell epitope datasets.

Table 6 Model performance evaluation for several machine learning algorithms trained on training set 1 (described in Table 4) and five unique reduced amino acid alphabet schemes.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.65	0.58	0.62	0.55	0.55	0.63	0.61	0.56	0.59	0.55	0.53	0.59
2	0.62	0.58	0.60	0.55	0.56	0.60	0.58	0.56	0.57	0.55	0.55	0.57
3	0.60	0.53	0.57	0.53	0.50	0.58	0.57	0.53	0.54	0.54	0.50	0.57
4	0.61	0.56	0.58	0.54	0.55	0.61	0.57	0.54	0.56	0.55	0.55	0.58
5	0.60	0.55	0.57	0.54	0.53	0.57	0.57	0.54	0.56	0.54	0.52	0.55

Table 7 Model performance evaluation for several machine learning algorithms trained on training set 2 (described in Table 4) and five unique reduced amino acid alphabet schemes.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.66	0.59	0.60	0.56	0.57	0.64	0.61	0.56	0.57	0.56	0.55	0.59
2	0.68	0.61	0.64	0.55	0.56	0.65	0.63	0.56	0.60	0.55	0.54	0.61
3	0.62	0.56	0.59	0.55	0.52	0.60	0.58	0.53	0.54	0.55	0.51	0.57
4	0.62	0.57	0.60	0.54	0.54	0.61	0.58	0.56	0.57	0.54	0.53	0.57
5	0.63	0.56	0.60	0.56	0.52	0.60	0.58	0.54	0.57	0.56	0.53	0.57

Table 8 Model performance evaluation for several machine learning algorithms trained on training set 3 (described in Table 4) and five unique reduced amino acid alphabet schemes.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.65	0.57	0.61	0.54	0.57	0.62	0.61	0.55	0.58	0.56	0.54	0.60
2	0.66	0.57	0.61	0.54	0.57	0.62	0.61	0.56	0.58	0.56	0.53	0.59
3	0.63	0.57	0.59	0.55	0.53	0.60	0.60	0.56	0.55	0.55	0.52	0.55
4	0.66	0.60	0.61	0.57	0.54	0.65	0.60	0.57	0.57	0.57	0.52	0.61
5	0.60	0.55	0.56	0.54	0.53	0.53	0.56	0.52	0.52	0.54	0.52	0.53

Table 9 Model performance evaluation for several machine learning algorithms trained on training set 4 (described in Table 4) and five unique reduced amino acid alphabet schemes.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.64	0.53	0.59	0.51	0.56	0.62	0.60	0.53	0.56	0.53	0.55	0.58
2	0.65	0.59	0.60	0.54	0.58	0.63	0.61	0.56	0.58	0.53	0.56	0.59
3	0.63	0.59	0.60	0.50	0.57	0.62	0.58	0.56	0.57	0.51	0.55	0.58
4	0.60	0.58	0.60	0.52	0.55	0.62	0.55	0.56	0.57	0.51	0.54	0.60
5	0.65	0.61	0.65	0.56	0.57	0.65	0.60	0.57	0.61	0.56	0.54	0.62
Control 1*	0.49	0.48	0.50	0.49	0.52	0.50	0.50	0.53	0.53	0.51	0.49	0.51
Control 2**	0.48	0.50	0.51	0.54	0.50	0.51	0.51	0.54	0.53	0.51	0.50	0.55

*Control based on randomized class labels for alphabet 5
**Control based on randomized n-gram probabilities for each sequence for alphabet 5

A random forest model, trained on a feature set derived from normalized ngram probabilities for Bcipep sequences greater than or equal to 12 residues and corresponding non-epitope sequences, represented by reduced alphabet scheme 2, showed the best prediction performance [Table 7]. This model achieved an average AUC and accuracy value, for its 10 stratified-cross-validation folds, of 0.68 and 0.63, respectively. Moreover, machine learning experiments were performed on two control feature sets. The first, took feature set alphabet 5 from Table 9 and randomized its class labels. The second, took feature set alphabet 5 from Table 9 and randomized its ngram probabilities for each sequence. Controls were used to confirm the signal in our feature sets and specifically the signal in our best performing feature set. The controls produced AUC and accuracy values around the 0.50 random guessing mark. Control 1 gave AUC and accuracy values of 0.49 and 0.50, respectively for random forest. Control 2 gave AUC and accuracy values of 0.48 and 0.51, respectively for random forest. When directly compared to the random forest model trained on feature set alphabet 5 from Table 9, we see a significant difference in prediction performance. Our model does significantly better than random guessing, which gives credibility to the normalized ngram probability approach with reduced alphabet schemes for developing sequenced-based feature sets that can be used to train machine learning algorithms to predict epitope and non-epitope sequences.

Figure 3 displays AUC results for several machine learning algorithms trained on each of the four different feature sets described in this section [Table 4], represented by reduced alphabet scheme 1. We see that the AUC values remain fairly consistent for each subset.

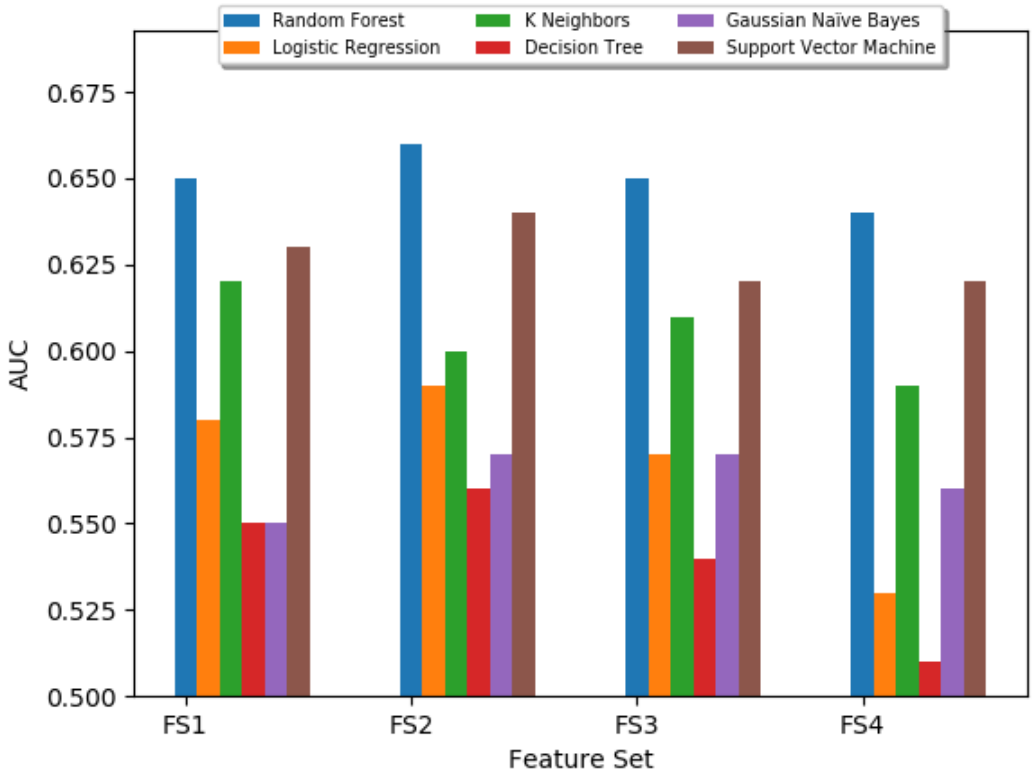


Figure 4 Comparison of AUC values for several machine learning algorithms trained on four different feature sets. Features sets correspond to training sets 1-4 from Table 4 represented by reduced alphabet scheme 1.

Figure 4 displays AUC values for several machine learning algorithms trained on the same subset, all sequences from the Bcipep database with length greater than or equal to 12, however the subset is represented by five different reduced alphabet schemes. We see that the AUC values remain fairly consistent for each alphabet.

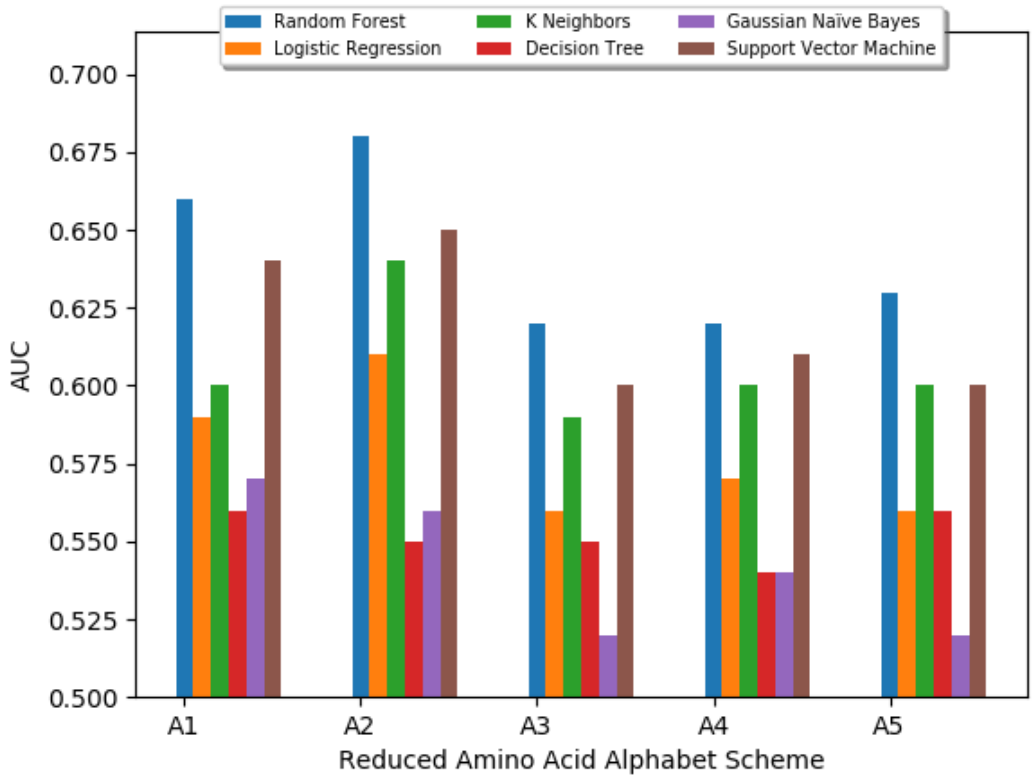


Figure 5 Comparison of AUC values for several machine learning algorithms trained on the same subset, all sequences from the Bcipep database with length greater than or equal to 12, with five different reduced alphabet schemes [Table 5].

Figure 6 is a confusion matrix that shows the breakdown of predictions for our best performing model.

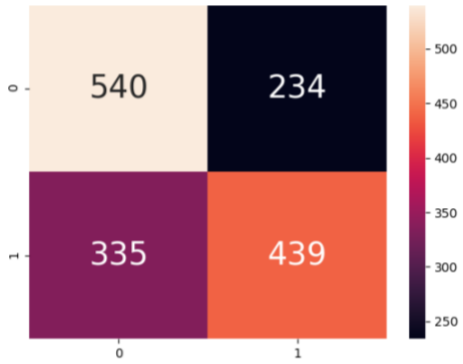


Figure 6 Confusion matrix for random forest model, trained on a feature set derived from normalized ngram probabilities for Bcipep sequences greater than or equal to 12 residues and corresponding non-epitope sequences, represented by reduced alphabet scheme 2. 0 is non-epitope (negative) and 1 is epitope (positive).

2.3.3 Tuning the Random Forest Model: Hyperparameter Optimization

Hyperparameters, parameters used to configure and control how certain machine learning algorithms learn, must be set before any training takes place [67]. The choice of hyperparameter values can have dramatic effects on the performance of the machine learning model [67]. Therefore, it is important to perform hyperparameter optimization to ensure proper selection of hyperparameter values to maximize model performance. Hyperparameter optimization methods help to alleviate many of the issues surrounding typical manual methods for setting hyperparameter values, such as reproducibility and computational burden [67], [68]. Here, we employ a hybrid approach for hyperparameter optimization that includes both random and grid search. Both methods were used to find a hyperparameter set that optimize the performance of the random forest model measured by AUC using stratified 10-fold cross-validation. First, random search, designed to search a wide parameter space by choosing random hyperparameter configurations, was

conducted to narrow value ranges for each of the chosen random forest hyperparameters. Hyperparameters included, number of estimators (trees), minimum samples to split an internal node, minimum samples required to be at a leaf node, number of features to consider when looking to split, maximum depth of the tree, and bootstrapping samples to build trees. Then, a grid search, which tests every parameter configuration combination, was performed on the narrowed hyperparameter value ranges. The grid search produced optimal hyperparameter values of 2400 trees, 2 samples to split an internal node, 2 samples to be at a leaf, 'sqrt' maximum features, 20 for maximum depth, and 'true' for bootstrap. The random forest model with these new hyperparameter values or "tuned random forest model" improved the performance of the best performing model from 0.68 to 0.69 AUC.

2.3.4 Linear B-cell Epitope Prediction Discussion

Our initial explorations into training supervised machine learning models of linear B-cell epitopes, based exclusively on sequence-derived features and a normalized n-gram probability approach, produced several candidate models that can be used to predict linear B-cell epitope sequences. These models have great promise; not only do they perform better than random guessing when evaluated using stratified 10-fold cross-validation but they own a very low computational burden. Training sets ranging from a low of 656 instances to a high of only 2420 instances coupled with fast training machine learning algorithms make this approach desirable.

Moreover, the performance of some of the models generated here is quite analogous to that of those found in the current literature. For comparison we will consider

BcePred, a linear B-cell epitope prediction tool published in 2004 [32]. The authors of BcePred developed a prediction method based on a combination of physiochemical properties, such as hydrophilicity, flexibility, polarity, and exposed surface [2], [32]. Interestingly, the performance of their models was tested and evaluated on a dataset consisting of 1029 non-redundant B-cell epitopes from Bcipep, the same source as our positive set, and an equal number of non-epitopes acquired from the Swiss-Prot database randomly [2], [32]. Comparing the performance of the models generated in this dissertation to that of those generated in the BcePred published work is rather reasonable due to the stark similarities in the training sets. The BcePred authors reported performance in terms of accuracy percentages. For models based on single residue properties, accuracy ranged between about 52.9% and 57.5%, and for models based on a combination of four residue properties, accuracy is about 58.7% [32]. Using our approach, the best performing model (random forest) achieved an average prediction accuracy, over the 10 stratified cross-validation folds, of about 64% (0.69 AUC).

For comparison to a published linear B-cell epitope prediction tool that is based on machine learning techniques, ABCpred will be considered. ABCpred uses both feed-forward neural network (FNN) and recurrent neural network (RNN) trained and tested on a dataset of 700 non-redundant B-cell epitopes from Bcipep and an equal number of random Swiss-Prot non-epitope sequences [26]. Their best prediction accuracy of about 65.9%, when tested using 5-fold cross-validation, was reached when using a RNN with a single hidden layer of 35 hidden units for a window length of 16 [26]. Our best performing model performs quite comparably to ABCpred's neural network-based model

when considering cross-validation prediction accuracy. This bodes well for the future of our approach given the often computationally intensive nature of neural network training. Future work may involve using a neural network/deep learning approach; however, these techniques typically perform better on much larger datasets, considerably larger than what has been used in this work. For a comprehensive look at other methods for B-cell epitope prediction refer to Table 2.

Additionally, it must be noted that the results achieved here give relevance to the reduced amino acid alphabet, normalized n-gram probability, machine learning approach for generating models for the prediction of linear B-cell epitope sequences and warrants further investigation and optimization to yield possible better model performance. We believe that adopting such an approach will be beneficial for the future improvement of epitope prediction.

2.4 Linear T-cell Epitope Prediction (Sequence-based)

2.4.1 Normalized n-gram Probability Approach

The approach for linear T-cell epitope prediction was performed according to the methods described in section 2.3.1. Refer to the flowchart for a detailed overview of the approach [Figure 3]. However, the experimentally validated linear T-cell epitope data used for the positive training set was obtained from the AntiJen Database *v2.0*, available from: http://www.ddg-pharmfac.net/antijen/AntiJen/antijenhomepage.htm. This database is an advancement of the previously established JenPep [69], [70]. In total, the positive set consisted of 1731 experimentally determined linear T-cell epitope sequences (MHC class I-binders).

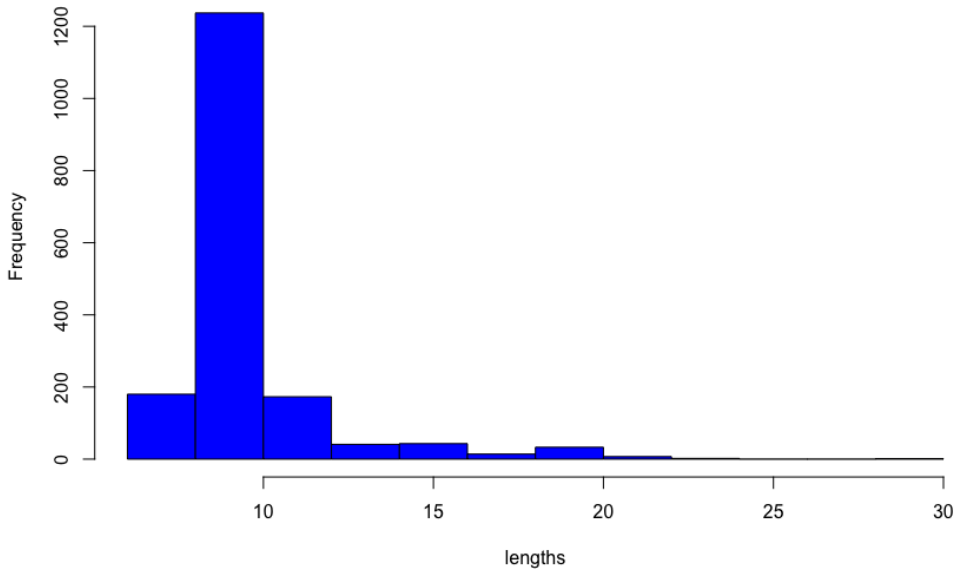


Figure 7 Lengths (number of residues) of linear T-cell epitope sequences from the AntiJen *v2.0* database.

Table 10 Frequency distribution of the lengths of the linear T-cell epitope sequences used in the positive set.

Scores	Frequency
(0,5]	0
(5,10]	1417
(10,15]	242
(15,20]	62
(20,25]	9
(25,30]	1

Most of the sequences in this dataset have lengths between 5 and 15 residues. Therefore, we chose to simply include all 1731 sequences in the positive set for these experiments.

2.4.2 Linear T-cell Epitope Prediction Results

Supervised machine learning models of linear T-cell epitopes were trained using several algorithms and model performance was evaluated. Table 11 displays the results for linear T-cell epitope prediction.

Table 11 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear T-cell epitope sequences from the AntiJen Database *v2.0* and corresponding random non-epitope sequences. The five unique reduced amino acid alphabet schemes are described in Table 5.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.75	0.62	0.71	0.64	0.60	0.73	0.68	0.58	0.65	0.63	0.57	0.66
2	0.74	0.63	0.68	0.62	0.61	0.72	0.68	0.59	0.64	0.62	0.58	0.65
3	0.69	0.60	0.65	0.60	0.60	0.66	0.63	0.58	0.60	0.61	0.56	0.63
4	0.67	0.56	0.62	0.58	0.55	0.64	0.62	0.54	0.59	0.58	0.53	0.59
5	0.69	0.61	0.65	0.61	0.60	0.66	0.64	0.58	0.61	0.61	0.55	0.63
Control 1*	0.49	0.51	0.49	0.50	0.51	0.48	0.48	0.51	0.49	0.49	0.51	0.50
Control 2**	0.56	0.58	0.51	0.50	0.56	0.56	0.55	0.55	0.52	0.52	0.53	0.54

*Control based on randomized class labels for alphabet 1
**Control based on randomized n-gram probabilities for each sequence for alphabet 1

Figure 8 presents the model evaluation AUC scores for each of the machine learning models from table 11.

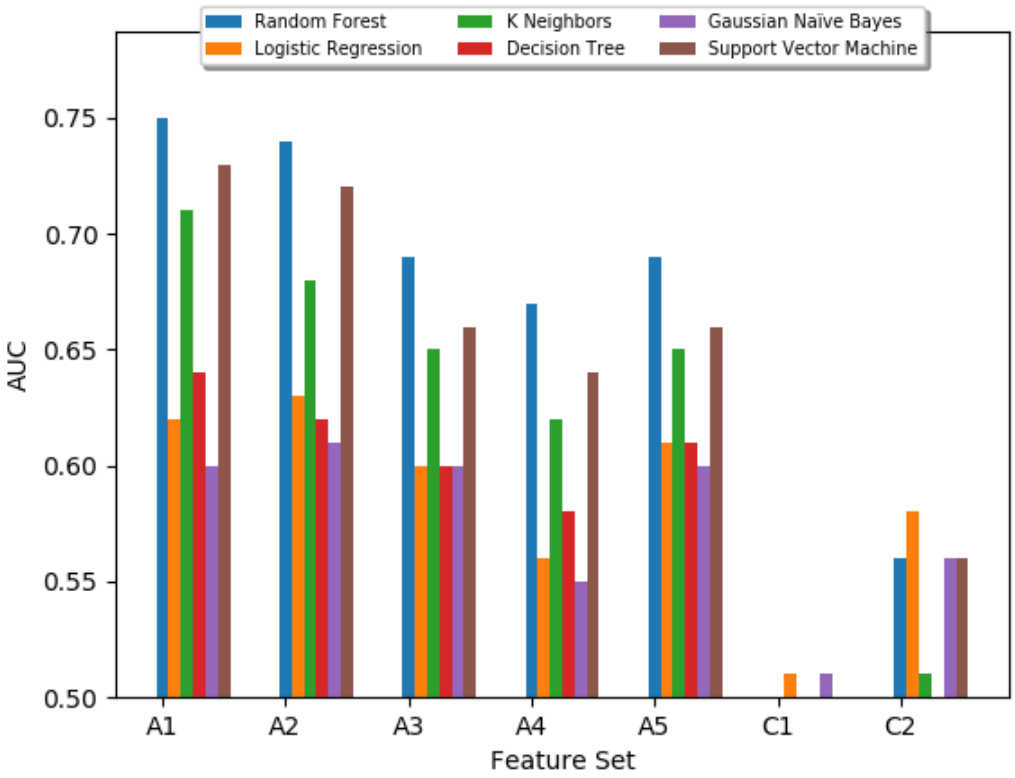


Figure 8 Comparison of AUC values for several machine learning algorithms trained on the sequences of the AntiJen Database *v2.0*, with five different reduced alphabet schemes [Table 5] and two random guessing controls.

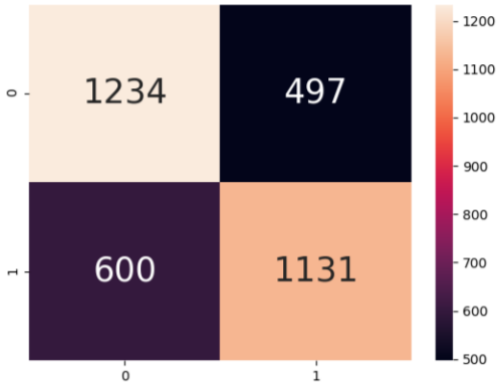


Figure 9 Confusion matrix for random forest model, trained on a feature set derived from normalized n-gram probabilities for AntiJen *v2.0* sequences and corresponding random non-epitope sequences, represented by reduced alphabet scheme 1. 0 is non-epitope (negative) class and 1 is epitope (positive) class.

The best performing model, evaluated by average AUC and accuracy over 10 stratified cross-validation folds, for linear T-cell epitope prediction was a random forest model trained on a feature vector derived from the sequences of the AntiJen Database *v2.0* and corresponding random non-epitope sequences using reduced amino acid alphabet scheme 1 and the normalized n-gram probability approach. The model achieved an average AUC and accuracy of approximately 0.75 and 0.68, respectively. The control feature sets tested here helped to confirm the results, as they produced AUC and accuracies around the 0.5 random guessing mark. This indicates the presence of a signal within our created feature vector that when coupled with machine learning can be used to distinguish between linear T-cell epitope and non-linear T-cell epitope sequences.

2.4.3 Tuning the Random Forest Model: Hyperparameter Optimization

Hyperparameter optimization was performed according to the methods of section 2.3.3. The grid search produced optimal hyperparameter values of 1700 trees, 5 samples to split an internal node, 2 samples to be at a leaf, ‘sqrt’ maximum features, 50 maximum depth, and ‘true’ bootstrap. The tuned model with optimized random forest hyperparameter values improved the highest achieved performance results from 0.75 to 0.76 AUC.

2.4.4 Linear T-cell Epitope Prediction Discussion

The explorations into training supervised machine learning models of linear T-cell epitopes, based solely on sequence-derived feature sets and the alphabet reduction/normalized n-gram probability approach, produced several candidate models for use in linear T-cell epitope prediction. Just like some of the models produced for

linear B-cell epitope prediction, using the same methodology, the best performing models here perform better than random guessing when evaluated using stratified 10-fold cross-validation and possess a very low computational burden; the training set contained only 3462 examples.

The focus of these experiments was the analysis of MHC class I binding peptides, a specific type of T-cell epitope. The justification for this stems from the knowledge that activated T-cells only recognize antigenic peptides that are bound to MHC molecules on cell surfaces [23]. MHC class I is one of two major classes of MHC molecules that display antigenic peptides on the cell surface to allow elimination by cytotoxic T-cells ($CD8^+$) [23]. Computational methods for the prediction of MHC class I-binding T-cell epitopes have shown wide ranging levels of performance with various tools having prediction accuracies between 60-99% [3]. In particular, we will highlight the work of Adams and Koziol (1995), which developed a neural network based approach for predicting MHC class I binders [71]. In this work, the neural networks, trained on sequence data from 552 nonamers and 486 decamers, achieved a predictive hit rate of 0.78. These results compare favorably to our best performing random forest model that achieved an average AUC of 0.76 when evaluated using stratified 10-fold cross-validation.

2.4.5 Discussion of Random Forests

After our supervised machine learning experiments designed to train models of both linear B-cell and T-cell epitopes, using only sequence-derived feature sets, we observe that the best performing models were based on the random forest algorithm. The

random forest (RF) algorithm was first introduced in 2001 by Breiman [72]. RF is an ensemble learning method, meaning multiple models/classifiers are used to solve the problem at hand, used for both classification and regression tasks [73] and is a combination of Breiman's bagging sampling approach [74] and the random selection of features [75]–[77].

Breiman defines a RF as "a classifier consisting of a collection of tree-structured classifiers $\{h(\boldsymbol{x}, \Theta_k), k = 1, ...\}$ where the $\{\Theta_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input **x**" [72]. In other words, a RF is a collection of a number of single decision tree classifiers or "forest", each tree is trained on a random sample with replacement of instances within the full training set, each node of the tree is split according to the best split of a random sample of features, the trees are maximally grown, and each tree determines its own classification or vote, the class with the highest vote total is chosen as the final classification.

Figure 10 illustrates a simple example of a decision tree classifier. In this example, our dataset contains seven instances and two classes, square and triangle. The decision tree is used to separate the two classes based on two features, color and number of sides. At the first node, we ask the question: Is it blue? This node splits the set into three blues and four reds. The non-blue split gives all triangles; therefore, no further splits are necessary. The blue split can be further split by asking the question: Does it have 4 sides? This node splits the set into two squares and one triangle. After the second node it seems that both classes are perfectly separated, and the tree is completed. This may be a

rather simple decision tree example used on a completely separatable dataset, but nevertheless random forests are constructed using the same guiding principles but with a collection of these individual decision trees. As mentioned before, a sample of instances from the full training set is used for each decision tree and a sample of features is used to determine the best feature to use to split each node. Each tree gives its own classification vote and the class with the highest number of votes is taken as the final classification decision. Advantages of the RF include robustness to noise and overfitting [73].

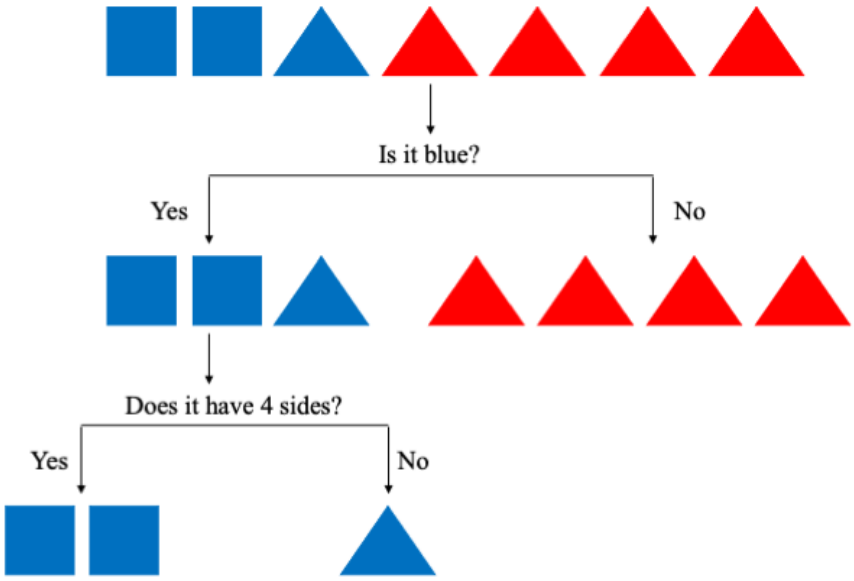


Figure 10 A simple example of a decision tree classifier used to separate two classes (square vs. triangle) using two features (color and number of sides) and two tree nodes.

Figure 11 shows a visual representation of one decision tree from our best performing RF model for linear B-cell epitope prediction. The number of features and depth of the tree makes it uninterpretable. However, it is provided for illustrative

purposes. Figure 12 shows a visual depiction of the same model as figure 11, however the maximum depth of the tree is set to 2 for better interpretation.

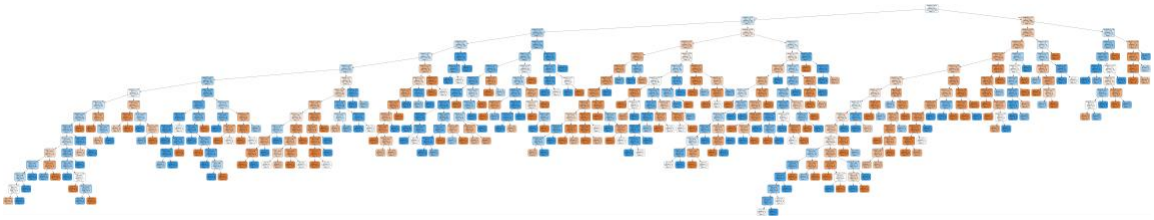


Figure 11 Visual representation of one decision tree from RF model of linear B-cell epitope.

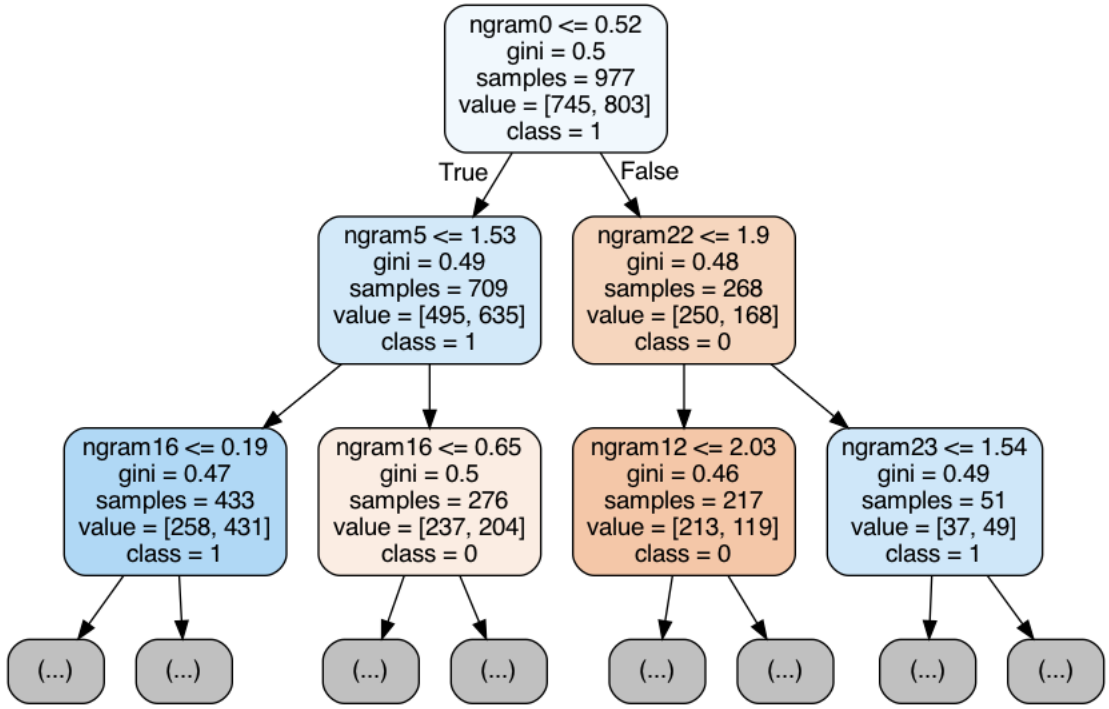


Figure 12 Visual representation of one decision tree from the same RF model of a linear B-cell epitope from the previous figure. However, the maximum depth of tree was restricted to 2 for better interpretation.

2.5 Exploring Additional 3-Letter Reduced Amino Acid Alphabet Schemes

Supplementary machine learning experiments were performed for linear epitope prediction according to the methods described in sections 2.3 and 2.4 for linear B-cell and T-cell epitope prediction, respectively. Though, to assess the value of the previously chosen reduced amino acid alphabet schemes used in the feature extraction process, new reduced amino acid alphabet schemes were created and used to represent the data set.

2.5.1 Novel 3-Letter Reduced Amino Acid Alphabet Scheme

A novel 3-letter reduced amino acid alphabet scheme was created based on amino acid indices and several different amino acid scales. To do so, a review of the literature was first conducted and the amino acid properties that have been found to be successful for epitope prediction in the past were chosen and their amino acid scales were used to rank all the amino acids from 1 to 20. Then, the ranks for all properties on each amino acid were summed and the 20 amino acids were clustered into three distinct groups based on these ranks. The three groups were designated the letters B, J, and U, respectively. This novel alphabet scheme was then implemented in the feature extraction procedure for both linear B- and T-cell epitope prediction. It was hypothesized that tailoring a reduced amino acid alphabet scheme to linear epitope prediction would yield improved machine learning model performance over models trained using the previous alphabet schemes that were not epitope prediction specific. This hypothesis is tested in this section. Table 12 displays the 8 amino acid properties chosen to create the alphabet scheme.

Table 12 List of properties and corresponding reference for the creation of the reduced 3-letter amino acid alphabet scheme.

Property	Reference
Amino acid indices (*Rk, Rc, Ro, Rb, Ph, Li*)	[78]
Average flexibility index*	[79]
Polarity*	[80]
Hydrophilicity*	[81]
Antigenicity*	[82]
Solvent Accessibility	[83]
Charge (PI)**	N/A
Secondary structure (beta-turns)*	[84]

*Obtained from https://web.expasy.org/protscale/
**Obtained from
https://www.sigmaaldrich.com/life-science/metabolomics/learning-center/amino-acid-reference-chart.html

The created amino acid alphabet scheme is presented below (B, J, U):

B = [R, K, D, G, S, H, Q]

J = [N, P, Y, E, T, A, C]

U = [V, L, I, W, F, M]

2.5.2 Results for Machine Learning Experiments using New Alphabet Scheme

Table 13 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear B-cell epitope sequences with length greater than or equal to 12 from the Bcipep database and corresponding random non-epitope sequences. The novel reduced amino acid alphabet described in section 2.5.1 was also used here.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.65	0.56	0.60	0.55	0.58	0.62	0.62	0.54	0.58	0.54	0.56	0.59
Control 1*	0.49	0.52	0.50	0.50	0.50	0.49	0.50	0.48	0.49	0.50	0.49	0.51
Control 2**	0.55	0.51	0.56	0.51	0.51	0.54	0.54	0.55	0.51	0.51	0.52	0.56

*Control based on randomized class labels for alphabet 1
**Control based on randomized n-gram probabilities for each sequence for alphabet 1

Table 14 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear T-cell epitope sequences from the AntiJen Database *v2.0* and corresponding random non-epitope sequences. The novel reduced amino acid alphabet described in section 2.5.1 was also used here.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.73	0.62	0.67	0.62	0.61	0.72	0.67	0.58	0.62	0.62	0.57	0.66
Control 1*	0.52	0.51	0.51	0.51	0.50	0.50	0.51	0.51	0.51	0.50	0.51	0.50
Control 2**	0.51	0.51	0.52	0.49	0.48	0.52	0.49	0.51	0.50	0.49	0.50	0.52

*Control based on randomized class labels for alphabet 1
**Control based on randomized n-gram probabilities for each sequence for alphabet 1

Table 13 and Table 14 display model performance evaluations for linear B- and T-cell epitope prediction, respectively. The evaluations revealed similar results to the previous machine learning experiments for linear B- and T-cell epitope prediction. The new alphabet scheme showed mixed performance for each of the machine learning algorithms used. While it outperformed some of the previous alphabet schemes, it did not perform as well as others. Each of the two controls ran for B-cell and T-cell epitope prediction, using the new created reduced amino acid alphabet, produced results around the 0.50 random guessing mark, confirming the signal in the approach. The best performing models were again random forest models, for linear B-cell epitope prediction this model achieved 0.65 AUC and 0.62 accuracy. For linear T-cell epitope prediction this model achieved 0.73 AUC and 0.67 accuracy. This result shows that the new alphabet scheme does improve prediction over random guessing. The new alphabet scheme did not improve performance dramatically as hypothesized. This may be due to the fact that the clustering or grouping of amino acids means more than the reduced alphabet itself. This will be further examined in the following section.

2.5.3 Random Reduced Amino Acid Alphabet Schemes

Because each of the six reduced amino acid alphabet schemes used in the feature extraction process for linear epitope prediction produced similar model performance, further experimentation is warranted to understand the value of the specific reduced amino acid alphabet scheme used. Towards this goal, five random reduced amino acid alphabet schemes were used as controls to determine the impact of the reduced alphabet on the performance of the machine learning models. First, the 20 amino acids were randomized and grouped into three groups, the first seven were grouped to the letter B, the next seven were grouped to the letter J, and the final six were grouped to the letter U. This procedure was performed five times to produce the five schemes. These new random reduced amino acid alphabet schemes were implemented according to the methods described in sections 2.3 and 2.4 for linear B- and T-cell epitope prediction, respectively. Results are displayed in the section 2.5.4.

2.5.4 Results

Table 15 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear B-cell epitope sequences with length greater than or equal to 12 from the Bcipep database and corresponding random non-epitope sequences. The five random reduced amino acid alphabets described in section 2.5.3 was also used here.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.61	0.55	0.59	0.53	0.51	0.60	0.57	0.54	0.57	0.54	0.50	0.56
2	0.58	0.54	0.57	0.53	0.52	0.58	0.56	0.53	0.55	0.53	0.52	0.55
3	0.59	0.52	0.57	0.53	0.48	0.59	0.57	0.52	0.55	0.52	0.50	0.56
4	0.66	0.60	0.61	0.59	0.58	0.64	0.61	0.56	0.58	0.58	0.54	0.61
5	0.59	0.52	0.56	0.53	0.51	0.54	0.55	0.51	0.54	0.53	0.50	0.51
Average	0.61	0.55	0.58	0.54	0.52	0.59	0.57	0.53	0.56	0.54	0.51	0.56

Table 16 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear T-cell epitope sequences from the AntiJen Database *v2.0* and corresponding random non-epitope sequences. The five random reduced amino acid alphabets described in section 2.5.3 was also used here.

Alphabet	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.66	0.61	0.63	0.59	0.58	0.66	0.61	0.58	0.60	0.58	0.55	0.61
2	0.68	0.59	0.64	0.58	0.57	0.65	0.63	0.56	0.60	0.58	0.56	0.61
3	0.62	0.56	0.59	0.56	0.53	0.61	0.58	0.55	0.57	0.56	0.52	0.57
4	0.69	0.61	0.65	0.60	0.60	0.67	0.63	0.57	0.61	0.59	0.57	0.62
5	0.64	0.58	0.61	0.56	0.57	0.63	0.59	0.56	0.58	0.56	0.54	0.59
Average	0.66	0.59	0.62	0.58	0.57	0.64	0.61	0.56	0.59	0.57	0.55	0.60

The performance of the machine learning models trained on both linear B-cell and T-cell sequence-derived feature vectors are displayed in Table 15 and Table 16, respectively. The model performance using the random reduced amino acid alphabet schemes with the normalized ngram approach declined compared to the previous six alphabet schemes used. Though the performance declined, decreases were only slight. The best performing algorithm was again random forest. Random forest models achieved an average AUC and accuracy for the five random schemes of 0.61 and 0.57, respectively for linear B-cell epitope prediction. For linear T-cell epitope prediction random forest models achieved an average AUC and accuracy for the five random schemes of 0.66 and 0.61, respectively. Once again, these models outperform the method of random guessing. This shows that there is still some signal in the approach regardless of the reduced alphabet scheme chosen. This leads to the idea that the amino acid alphabet scheme itself does not matter as much to the overall feature extraction process compared to the actual 3-letter ngram groupings themselves.

2.6 n-gram "Counts" Method for Linear Epitope Prediction

To further explore the development of linear epitope predictive models, using sequence-derived feature sets, the n-gram "counts" method was implemented. For this method, each sequence in the dataset is represented as a 3^n feature vector. Here, "3" represents the reduced three-letter amino acid alphabet that replaces the traditional twenty-letter amino acid alphabet and "n" represents the n-gram size, 3 in this case. Each component of the feature vector represents an individual n-gram type and the value placed in the vector represents the count or absolute frequency of that n-gram in the particular sequence. The counts method is previously described in [85].

The feature sets used to train the best performing linear B- and T-cell epitope prediction models, using the normalized n-gram probability method, were used to train the models using the counts method, Table 17 and Table 18 show results for these machine learning experiments.

Table 17 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear B-cell epitope sequences with length greater than or equal to 12 from the Bcipep database and corresponding random non-epitope sequences. (Alphabet 2)

Feature Set	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.69	0.61	0.63	0.57	0.62	0.66	0.65	0.57	0.58	0.57	0.58	0.62

Table 18 Model performance evaluation for several machine learning algorithms trained on sequence-derived feature vectors from experimentally validated linear T-cell epitope sequences from the AntiJen Database *v2.0* and corresponding random non-epitope sequences. (Alphabet 1)

Feature Set	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1	0.75	0.68	0.69	0.64	0.64	0.72	0.68	0.63	0.62	0.64	0.60	0.65

The counts method used in the feature extraction process yielded machine learning models that performed similar to the normalized n-gram probability models for both B- and T-cell epitope prediction. The best performing model for B-cell epitope prediction was once again a random forest model. This model achieved an AUC and accuracy of 0.69 and 0.65, respectively. The best performing model for T-cell epitope prediction was also a random forest model that achieved and AUC and accuracy of 0.75 and 0.68, respectively. The counts method shows promise for deriving sequence-based features for the prediction of linear epitopes due to its simplicity and performance compared to other methods. This method can be key to developing even better predictive models in the future.

2.7 Linear B-cell Epitope Model Prediction on Independent Test Set

The best performing linear B-cell epitope model from section 2.3.3 was used to make predictions on an independent test set. This was done to further evaluate the performance and utility of the model. The independent test set was based on experimentally validated B-cell epitopes collected by the authors of Lbtope [31] and random PDB sequence fragments. The experimentally validated epitope dataset (Lbtope_Variable) is made up of 14876 B-cell epitopes of varying lengths [31]. To ensure no overlap between the set used to train the predictive model, a global sequence alignment was performed pairwise between all sequences in the Lbtope set and all sequences in the original training set. Any sequence in the test set that shared greater than or equal to 95% sequence identity with any sequence in the training set was removed. This ensured no duplicate epitope sequences in the training and test sets. The scoring

system used for the global sequence alignments was match: +1, mismatch: -1, Gap open: -0.5, Gap extend: -0.1. The sequence identity was calculated according to the following: (alignment score / query sequence length) * 100. The final test set consisted of a subset of 100 B-cell epitope sequences from the cleaned Lbtope set and 100 non epitope sequences (non-epitope sequences were generated using the methods described in section 2.3.1).

The prediction results for our model on the independent test set are displayed in a confusion matrix [Figure 13] and ROC curve [Figure 14].

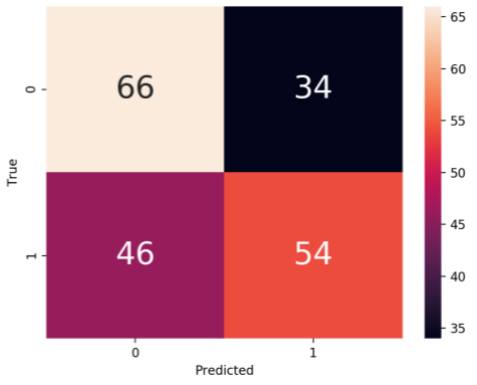


Figure 13 Confusion matrix for random forest model predictions on an independent test set. 0 is non-epitope (negative) and 1 is epitope (positive).

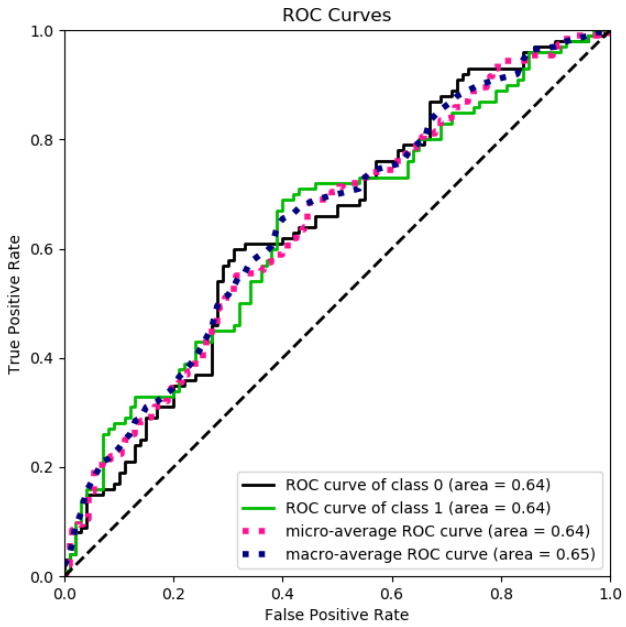


Figure 14 ROC curve for random forest model predictions on an independent test set. AUC = 0.64.

Our random forest model achieved an AUC of about 0.64 on the independent test set. This model performs better than random guessing which is a noteworthy result. However, future work can be done to improve the prediction performance of our linear B-cell epitope models. Improvements can be made to size and variability of the training set.

2.8 Linear T-cell Epitope Model Prediction on Independent Test Set

The best performing linear T-cell epitope model from section 2.4.3 was used to make predictions on an independent test set. As in the previous section, this was done to further evaluate the performance and utility of the model. The independent test set was based on experimentally validated T-cell epitopes collected from The Immune Epitope Database (IEDB) (https://www.iedb.org/) and random PDB sequence fragments. To

collect the T-cell epitope sequences, a search was performed on the IEDB. The search was conducted for linear epitopes, T-cell assays, and MHC class 1 binders. A total of 1000 sequences were extracted for possible use in the independent test set. Once again, to ensure no overlap between the set used to train the predictive model, a global sequence alignment was performed pairwise between all sequences in the IEDB set and all sequences in the original training set. Any sequence in the test set that shared greater than or equal to 95% sequence identity with any sequence in the training set was removed. This ensured no duplicate epitope sequences in the training and test sets. The scoring system used for the global sequence alignments was match: +1, mismatch: -1, Gap open: -0.5, Gap extend: -0.1. The sequence identity was calculated according to the following: (alignment score / query sequence length) * 100. The final test set consisted of a subset of 250 T-cell epitope sequences from the cleaned IEDB set and 250 non epitope sequences (non-epitope sequences were generated using the methods described in section 2.3.1).

The prediction results for our model on the test set are displayed in a confusion matrix [Figure 15] and ROC curve [Figure 16].

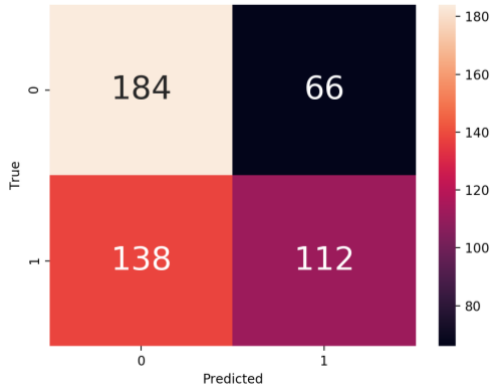


Figure 15 Confusion matrix for random forest model predictions on an independent test set. 0 is non-epitope (negative) and 1 is epitope (positive).

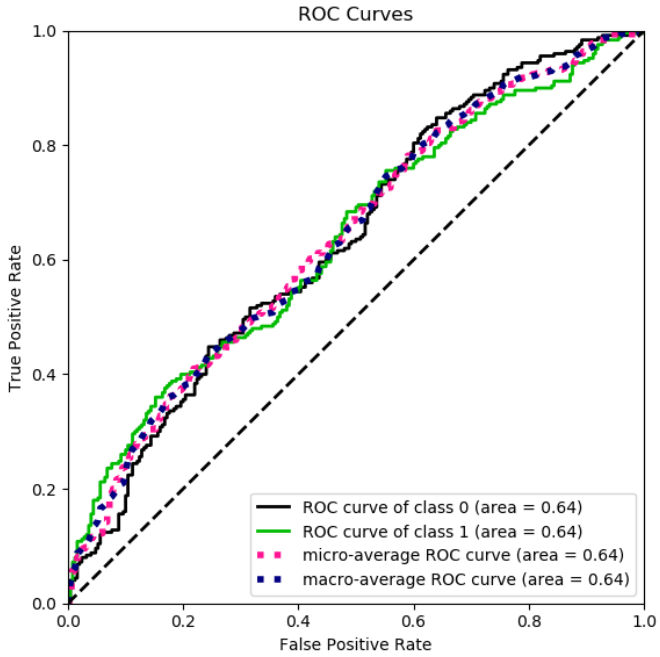


Figure 16 ROC curve for random forest model predictions on an independent test set. AUC = 0.64.

Our pre-trained random forest model achieved an AUC of about 0.64 on the independent test set. Just like for linear B-cell epitope prediction, this model outperforms

the method of random guessing. However, future work can be done to improve the prediction performance of our linear T-cell epitope models. Improvements can be made to size and variability of the training set.

CHAPTER THREE

3.1 Structure-based Epitope Prediction: A Brief Overview

The vast nature of readily available epitope sequence data has made sequence-based methods for epitope prediction prominent in the scientific literature. However, as mentioned previously, most epitopes are conformational (discontinuous) in composition. Therefore, using solely epitope sequence-derived features to represent these epitopes may be inadequate for training reliable predictive models because conformational epitopes are not continuous in the protein sequence but are brought into proximity through protein folding. This aspect of conformational epitopes is the motivation for deriving machine learning features from the three-dimensional protein structure of the epitope. As before, the features can be the components of a feature vector that can be used in machine learning experiments to train models that predict epitopic residues. This chapter will introduce our approach for structure-based epitope prediction and the specific method of Delaunay tessellation which was used to probe bound antigen-antibody structures.

3.2 Introduction to Delaunay tessellation

Early protein structure analysis methods relied on definitions of nearest neighbor residues that were based on arbitrary distance criteria [86]. Examples of nearest neighbor definitions include C_α atoms separated by no more than 5.5 Å [87] and at most 2.8 Å separating any pair of atoms belonging to different residues [88]. These arbitrary

definitions of residue contacts can dramatically bias the results of the protein structural analyses in which they are used. Consequently, unbiased and robust protein residue nearest neighbor definitions are necessary to address these issues in protein structure analysis [86]. The method of Delaunay tessellation has been introduced to provide these unbiased and robust definitions.

Delaunay tessellation is used to explore the graph or network theoretic properties of protein structures, while they are being represented as a residue contact map [89]. As mentioned before, residue contact maps were typically studied using a definition of the contacts between the residues based on a pairwise residue separation in the 3D-space, relying on arbitrary distance cut-off values [89]. Delaunay tessellation counters this concept by defining the residue contact map in a more exhaustive manner [89]. Simply put, Delaunay tessellation is a method of describing the space between a set of points, points that represent the protein structure [89]. The points can either represent an atom, a group of atoms, or a residue, for this work the single point per residue method was chosen, where the points are set at the α-carbon atom locations [89]. The points are connected by edges that form nonoverlapping tetrahedra and residues that are connected by an edge are considered nearest neighbors [89]. The Delaunay tessellation of a set of 3-dimensional points can be found by lifting the points to a paraboloid and computing their 4-dimensional convex hull [86].

The goal of using Delaunay tessellation in these initial explorations into structure-based epitope prediction was to determine the nearest neighbors of the amino acid residues in the protein chain of an antigen interacting with an antibody. This would

replace the neighborhood definitions used by DiscoTope [34], [90], in hopes of improving epitope residue prediction. We were also interested in examining each simplex formed by each residue in the protein structure. Moreover, a four-body statistical contact pseudo-potential derived from Delaunay tessellation could be determined for each simplex and the sum of all potentials for each residue could be used as a component in the feature vector for the residue in machine learning experiments.

3.2.1 Four-body statistical contact pseudo-potential

Every amino acid residue in tessellated protein structures participate in a number of Delaunay simplices and each simplex is a quadruplet of amino acids. Singh, Tropsha, and Vaisman (1996) analyzed Delaunay simplices in their dataset to determine the statistical likelihood of four nearest neighbor amino acid residues occurring [86]. The log-likelihood factor, q, for each of the quadruplet defined simplices was computed according to Equation 3.

$$\boldsymbol{q_{ijkl} = log \frac{f_{ijkl}}{p_{ijkl}}} \quad (\mathbf{3})$$

i, j, k, and l represent one of the 20 natural amino acid residues, f_{ijkl} is the normalized observed quadruplet frequency, and p_{ijkl} is the randomly expected quadruplet frequency. The likelihood of observing four specific residues in one simplex, q_{ijkl}, can be further described by Equation 4 and Equation 5.

$$f_{ijkl} = \frac{total\ number\ of\ occurrence\ of\ each\ quadruplet\ type}{total\ number\ of\ observed\ quadruplets\ of\ all\ types} \quad (4)$$

$$p_{ijkl} = C a_i a_j a_k a_l, a \quad (5)$$

a_i, a_j, a_k, and a_l are the observed amino acid frequencies for a given amino acid and *C* is the permutation factor.

$$C = \frac{4!}{\prod_i^n (t_i!)} \quad (6)$$

n denotes the quantity of distinct residue types in a quadruplet and t_i denotes the total number of amino acids of type *i*. We assume that there is order independence among residues that make up Delaunay simplices. Therefore, the maximum number of all possible combinations of quadruplets that form simplices is 8855. The values of q (four-body potential energy function) found by these authors were used in this work to generate protein structure-based feature sets to describe epitope and non-epitope residues.

3.3 Conformational B-cell Epitope Prediction (Structure-based) Methods

3.3.1 Dataset

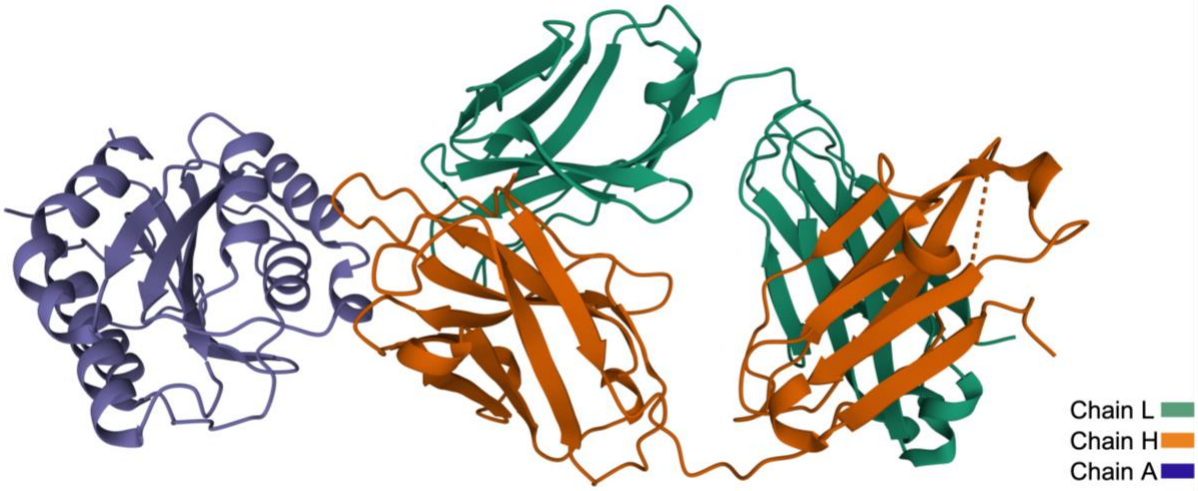
Our final dataset was based on the DiscoTope dataset [34], [90]. The dataset consisted of 75 X-ray crystallography determined antigen-antibody complex structures. The three-dimensional protein structures were obtained from the PDB. After manual

inspection of the structures and the epitope annotations provided by the authors of DiscoTope, a final set of 68 antigen-antibody complex structures was used to ensure integrity between the epitope annotations and the structures pulled from the PDB. Python scripts were written to extract the α-carbon atoms for all amino acid residues in the dataset, obtain the x, y, z, coordinates for the atoms, and perform Delaunay tessellation on the structures via the pyhull module. Pyhull is a python wrapper to qhull [91]. It must be noted that only the protein chain of the antigen that interacts with the antibody was used in the Delaunay tessellation. DiscoTope used the 4 Å rule to determine the epitope residues, where epitope residues were described as antigen amino acids having atoms within 4 Å of antibody atoms [90]. Delaunay tessellation was used to obtain the nearest neighbor definitions for each amino acid residue, instead of using arbitrary distance cut-offs or functions, like those used in DiscoTope.

Figure 17 shows the three-dimensional crystal structure of one protein complex obtained from the final dataset along with a visualization of the Delaunay tessellation of one of its protein chains. The crystal structure is of the von Willebrand factor (VWF) A1 domain in complex with the function blocking NMC-4 antigen-binding fragment (Fab) (PDB ID: 1OAK). VWF is a large, multimeric glycoprotein that is predominantly found in blood plasma and performs two key functions in hemostasis, mediation of the adhesion of platelets to subendothelial connective tissue and binding to blood clotting factor VIII [92]. Patients with VWF deficiencies often suffer from severe bleeding disorders due to blood clot and platelet plug defects [92]. The A1 domain of VWF binds the platelet glycoprotein (GP) Ibα to initiate hemostatic plugs in rapid blood flow wounds [93].

Alternatively, pathological conditions, such as thrombotic occlusions can arise due to VWF and GP Ibα interaction [93]. The Fab fragment of NMC-4, a mouse monoclonal antibody that binds to the A1 domain of VWF has been used to understand the interactions between VWF and GP Ibα that cause certain pathologic conditions, like thrombus [94]. Delaunay tessellation was performed on the antigen-antibody interacting chain. This is an example of a "self" antigen, one derived from the host.

(A)



(B)

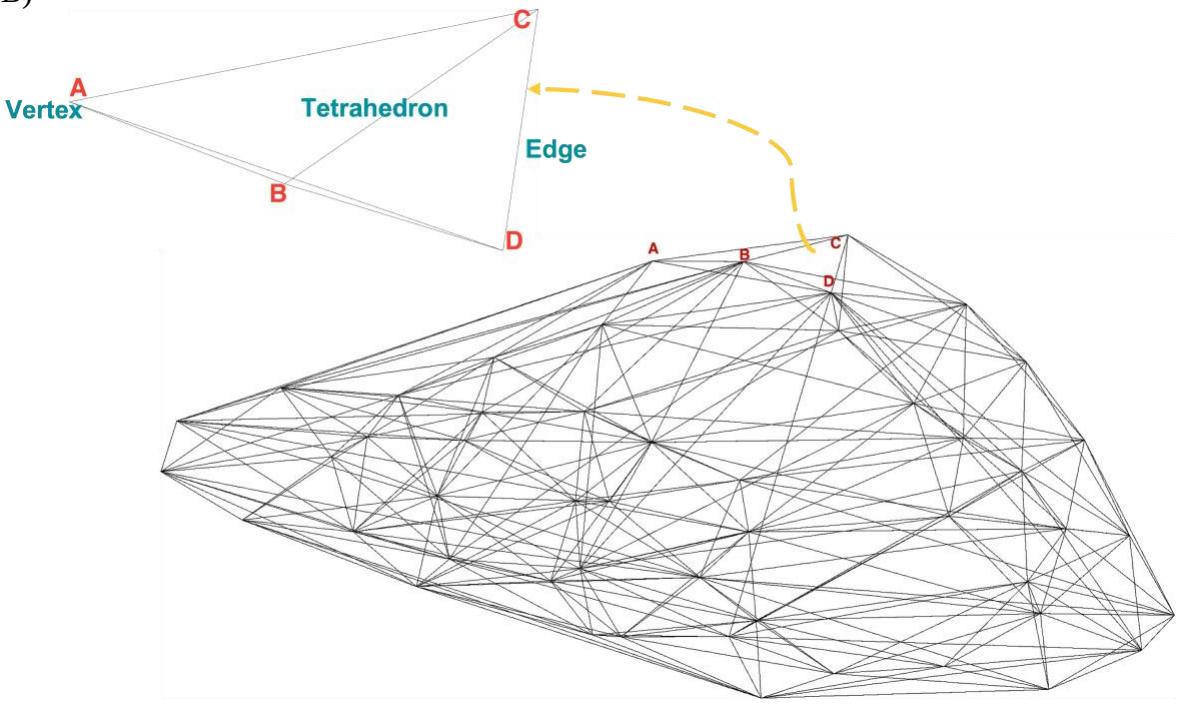


Figure 17 (A) Crystal structure of the VON WILLEBRAND FACTOR (VWF) A1 domain in complex with the function blocking NMC-4 Fab. 3D structure created using PDB's 3D View tool (PDB ID: 1OAK) [61], [93], [95]. (B) Delaunay tessellation of a 50-residue segment of protein chain A of PDB ID: 1OAK with an example of one tetrahedral simplex consisting of four α-carbon atom vertices [A, B, C, D]. Vertex, tetrahedron, and edge are labelled. B) created using MATLAB script provided by Dr. Majid Masso.

Figure 18 and Figure 19 provide visualizations of the residues making up the discontinuous epitope formed due to the interaction of the VWF's A1 domain (chain A) and the function blocking NMC-4 Fab (chain H). The epitope consists of 13 residues.

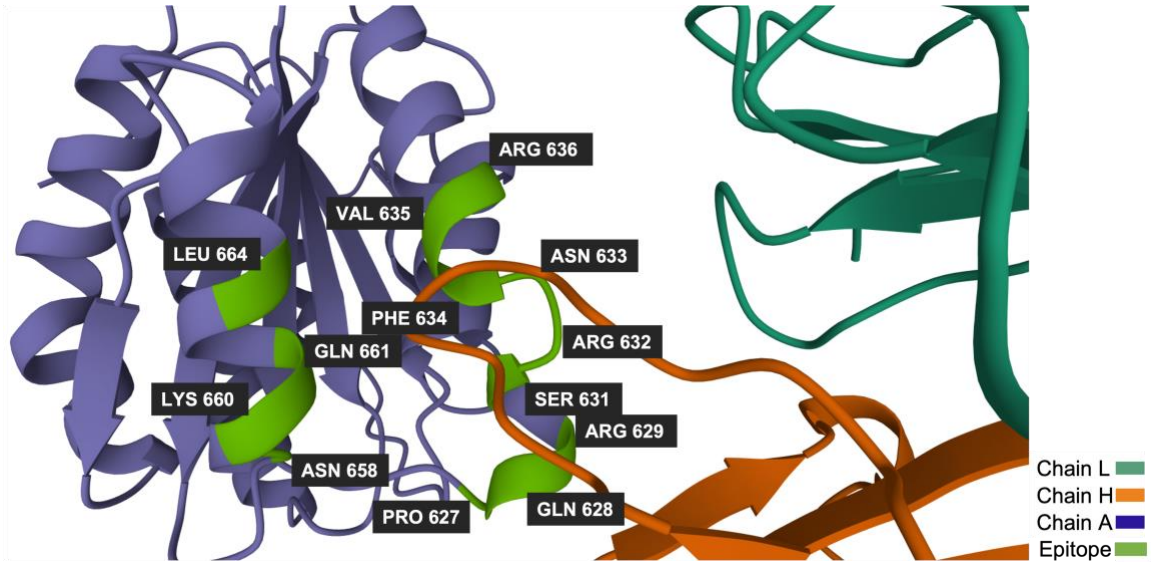


Figure 18 Crystal structure of the VWF A1 domain in complex with the function blocking NMC-4 Fab. 3D structure created using PDB's 3D View tool (PDB ID: 1OAK) [61], [93], [95]. The residues making up the discontinuous epitope are highlighted in light green on protein chain A and interact with chain H.

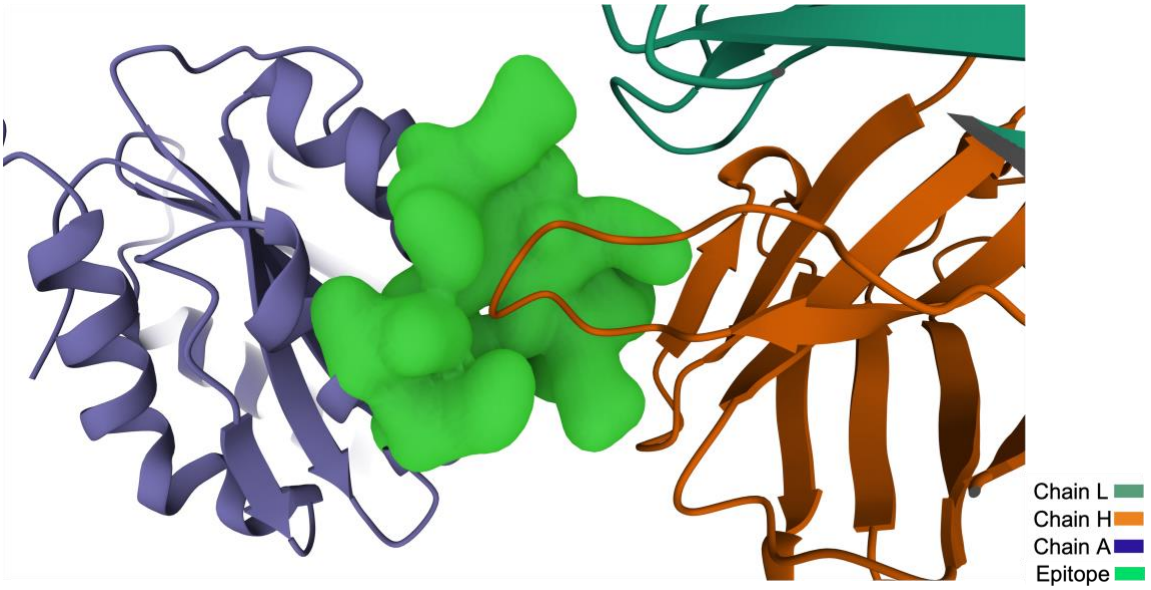


Figure 19 Crystal structure of the VWF A1 domain in complex with the function blocking NMC-4 Fab. 3D structure created using PDB's 3D View tool (PDB ID: 1OAK) [61], [93], [95]. The residues making up the discontinuous epitope and surrounding region in 3D space are highlighted in light green on protein chain A and interact with chain H.

3.3.2 Feature Vector

A feature vector based on summed scores for all amino acid residues in the dataset was constructed to train machine learning algorithms to predict epitope and non-epitope residues that appear in discontinuous B-cell epitopes. First, a list of nearest neighbors, determined by the Delaunay tessellation tetrahedra, for each amino acid residue in the antigen protein chain was obtained for all structures in the dataset. For each neighborhood (simplex), the log-odds ratios (values placed on the representation of a certain amino acid type in an epitope) obtained from, [90], for each amino acid were summed and used as a component in the machine learning feature vector. Second, the summed four-body statistical contact pseudo-potential for each of the simplices that the residue of interest participates was calculated and used as a component in the machine learning feature vector.

3.4 Conformational B-cell Epitope Prediction (Structure-based) Results

The results for machine learning experiments designed to train models to predict epitope and non-epitope residues in the antigen-antibody interacting chain of bound protein structures using the 2-component feature vector are displayed in Table 21.

Table 19 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC). Models were trained using a 2-component feature vector with default parameters and tested using 5-fold stratified cross-validation.

PDB ID	Epitope Size (residues)	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1OAK.A	13	0.75	0.85	0.69	0.56	0.79	0.79
1TQC.A	19	0.50	0.38	0.50	0.51	0.44	0.46
1H0D.C	17	0.54	0.46	0.67	0.54	0.46	0.63
1GC1.G	11	0.54	0.71	0.48	0.48	0.55	0.56
1RZK.G	12	0.47	0.58	0.41	0.48	0.52	0.53

1K4D.C	13	0.75	0.75	0.72	0.72	0.86	0.80
1IC5.Y	16	0.56	0.59	0.41	0.50	0.61	0.51
1YQV.Y	14	0.80	0.87	0.71	0.55	0.86	0.58
1G7H.C	16	0.48	0.63	0.48	0.51	0.62	0.47
1G9M.G	12	0.55	0.64	0.56	0.48	0.55	0.73
1RZJ.G	11	0.51	0.71	0.64	0.49	0.62	0.73
1QFU.A	20	0.44	0.52	0.44	0.47	0.55	0.33
1IC4.Y	18	0.54	0.61	0.48	0.49	0.60	0.33
1EGJ.A	12	0.54	0.80	0.64	0.56	0.79	0.69
1A2Y.C	15	0.50	0.57	0.47	0.53	0.54	0.55
1G7I.C	15	0.46	0.64	0.45	0.47	0.64	0.56
1XIW.A	17	0.67	0.76	0.71	0.65	0.72	0.62
1KYO.E	15	0.67	0.78	0.64	0.46	0.79	0.75
1TQB.A	18	0.52	0.32	0.41	0.46	0.48	0.34
1NDM.C	18	0.56	0.60	0.58	0.54	0.60	0.62
1FE8.A	19	0.53	0.71	0.62	0.58	0.68	0.69
1NBZ.C	19	0.50	0.67	0.59	0.56	0.68	0.64
2JEL.P	15	0.69	0.64	0.72	0.57	0.67	0.55
1BJ1.W	16	0.65	0.74	0.66	0.62	0.73	0.72
1IQD.C	16	0.55	0.62	0.57	0.49	0.63	0.59
1NBY.C	19	0.63	0.64	0.58	0.57	0.65	0.63
1TZH.V	13	0.81	0.88	0.77	0.59	0.88	0.84
1BQL.Y	13	0.73	0.85	0.72	0.59	0.81	0.65
1G9N.G	12	0.53	0.64	0.53	0.48	0.57	0.61
1G7J.C	15	0.55	0.69	0.57	0.46	0.69	0.47
1IC7.Y	17	0.50	0.61	0.43	0.47	0.57	0.54
1NDG.C	21	0.51	0.64	0.52	0.47	0.62	0.47
1TPX.A	18	0.52	0.35	0.46	0.62	0.47	0.34
1K4C.C	14	0.75	0.77	0.76	0.59	0.77	0.76
1BVK.C	16	0.56	0.59	0.52	0.58	0.54	0.45
1KIP.C	15	0.53	0.66	0.45	0.55	0.66	0.60
1J1X.Y	19	0.55	0.65	0.51	0.54	0.62	0.51
1ORS.C	10	0.54	0.55	0.64	0.50	0.70	0.66
2HMI.B	9	0.73	0.83	0.65	0.49	0.80	0.79
1TY6.A (2VDN.A)	19	0.61	0.65	0.60	0.50	0.75	0.65
1MHP.A	16	0.87	0.92	0.86	0.68	0.92	0.89
1FNS.A	12	0.67	0.81	0.74	0.60	0.74	0.73
1EZV.E	17	0.74	0.78	0.68	0.54	0.79	0.66
1J1P.Y	20	0.62	0.63	0.50	0.52	0.63	0.48
1KIQ.C	15	0.54	0.68	0.48	0.49	0.68	0.61
1DZB.X	18	0.65	0.62	0.63	0.51	0.65	0.55
1MEL.L	22	0.65	0.58	0.44	0.56	0.57	0.61
1OSP.O	20	0.60	0.76	0.64	0.51	0.74	0.55
1JPS.T	21	0.45	0.73	0.45	0.50	0.63	0.50
1OAZ.A	16	0.65	0.67	0.66	0.56	0.57	0.61
1C08.C	17	0.46	0.60	0.41	0.49	0.60	0.52
1G7L.C	15	0.48	0.63	0.37	0.52	0.63	0.43
1JHL.A	11	0.76	0.72	0.64	0.50	0.81	0.72
1AR1.B	15	0.56	0.77	0.57	0.50	0.80	0.68
1MLC.E	16	0.74	0.81	0.70	0.62	0.84	0.75

1CZ8.W	16	0.62	0.74	0.62	0.48	0.72	0.67
1G7M.C	15	0.35	0.64	0.46	0.44	0.64	0.49
1EO8.A	17	0.58	0.68	0.57	0.47	0.62	0.55
1KIR.C	14	0.49	0.67	0.58	0.48	0.62	0.59
1FJ1.F	17	0.53	0.63	0.60	0.47	0.71	0.80
1OTS.A	9	0.67	0.72	0.66	0.59	0.76	0.66
1DQJ.C	21	0.55	0.66	0.54	0.50	0.63	0.49
1N8Z.C	17	0.58	0.80	0.61	0.52	0.69	0.55
1J1O.Y	19	0.54	0.67	0.48	0.50	0.66	0.40
1LK3.A	18	0.79	0.80	0.72	0.59	0.76	0.71
1FDL.Y	14	0.58	0.69	0.41	0.51	0.65	0.53
1JRH.I	15	0.63	0.77	0.69	0.50	0.68	0.61
1FSK.A	17	0.71	0.81	0.59	0.54	0.74	0.47

The machine learning experiments involved training models using feature sets derived from only one protein complex. This approach severely limits the amount of training data available for the models. Therefore, to study the impact of increasing the size of the training set - the number of residues trained on, we trained models that combined information from multiple protein complexes. The same 2-component feature vector from above was used here but models were trained with 15, 30, 45, 60, and 68 protein structure complexes, respectively. Results for these machine learning experiments are displayed in Table 22.

Table 20 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 2-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.65	0.61	0.59	0.57	0.60	0.54	0.92	0.92	0.91	0.87	0.92	0.92
30	0.73	0.64	0.67	0.61	0.64	0.52	0.91	0.91	0.89	0.87	0.91	0.91
45	0.77	0.64	0.70	0.66	0.64	0.54	0.92	0.91	0.90	0.89	0.91	0.91
60	0.77	0.65	0.70	0.67	0.64	0.54	0.92	0.91	0.90	0.88	0.91	0.91
68	0.77	0.67	0.69	0.66	0.65	0.58	0.92	0.91	0.91	0.88	0.91	0.91

The performance results for these machine learning models were further examined to understand their utility. Therefore, a confusion matrix was generated for one of the models. This is shown in Figure 20 below.

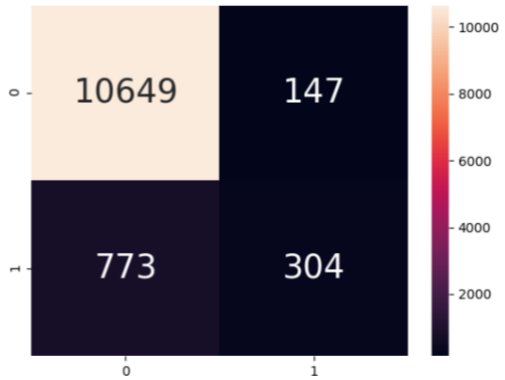


Figure 20 Confusion matrix for random forest model, trained on a 2-component feature set with structure information from 68 protein complexes. 0 is non-epitope (negative) and 1 is epitope (positive).

Figure 20 shows that the random forest model produces 10649 true negatives, 773 false negatives, 304 true positives, and 147 false positives. Therefore, in this full dataset there are 1077 epitope residues and 10796 non-epitope residues. This dataset suffers from class imbalance, to address this issue we will train models on balanced training sets with an equal number of epitope and non-epitope residues.

3.4.1 Conformational B-cell Epitope Prediction (Structure-based) Results for Balanced Training Sets

The five subsets listed in Table 22 were transformed from unbalanced to balanced training sets. This was done by randomizing all of the non-epitope residues (the dominant class) in the full dataset and taking samples with sizes equivalent to the number of epitope residues in each of the five subsets. Thus, the five subsets contained 438, 918, 1394, 1894, and 2154 residues respectively, each containing an equal number of epitope and non-epitope residues. Results for these machine learning experiments are displayed in Table 23.

Table 21 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 2-component feature vector on balanced sets with default parameters and tested using 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.64	0.65	0.62	0.56	0.64	0.64	0.60	0.59	0.61	0.55	0.59	0.58
918	0.69	0.64	0.62	0.60	0.64	0.64	0.61	0.61	0.59	0.61	0.59	0.58
1394	0.73	0.66	0.69	0.63	0.66	0.65	0.66	0.62	0.64	0.63	0.59	0.61
1894	0.75	0.67	0.69	0.66	0.66	0.67	0.67	0.63	0.63	0.65	0.61	0.62
2154	0.75	0.68	0.70	0.65	0.66	0.67	0.68	0.63	0.64	0.66	0.61	0.63

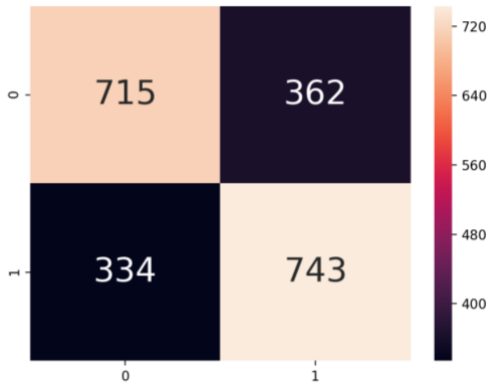


Figure 21 Confusion matrix for random forest model, trained on a 2-component feature set with structure information from 2154 amino acid residues. 0 is non-epitope (negative) and 1 is epitope (positive).

The results obtained for the models trained on the balanced sets are more reasonable than that for the unbalanced sets and they show the true performance of the models. The best performing model for these experiments was a random forest model that was trained on 2154 residues from the interacting chain of various bound antigen-antibody protein structures. The 2154 residues were balanced with an equal number of epitope and non-epitope residues. The model achieved 0.75 and 0.68 AUC and accuracy, respectively.

3.4.2 Conformational B-cell Epitope Prediction (Structure-based) using Strictly Delaunay tessellation-based Feature Sets

The summed four-body statistical contact pseudo-potential derived from Delaunay tessellation described in section 3.2.1 was used to derive new structure-based feature sets to train machine learning models to predict epitope and non-epitope residues in the antigen-antibody interacting chain of bound protein structures. A seven-component feature vector was created for each of the residues in the dataset. This feature set consisted of the potential score for the central residue (residue of interest) and its six-

nearest neighbors. Nearest neighbor lists were determined by Delaunay tessellation and the closest six in terms of Euclidean distance were chosen for these experiments. Three protein complexes, 1EZV, 1K4C, and 1KYO, were removed from the dataset because they had at least one residue that did not have the minimum six nearest neighbors to generate this feature set. Therefore, these experiments were performed on a dataset with a total of 65 antigen-antibody protein structures. Results for these machine learning experiments are given in section 3.4.3.

3.4.3 Prediction Results for Delaunay-derived Feature Vectors

Table 22 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC). Models were trained using a 7-component feature vector with default parameters and tested using 5-fold stratified cross-validation.

PDB ID	Epitope Size (residues)	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
1OAK.A	13	0.77	0.77	0.59	0.45	0.73	0.59
1TQC.A	19	0.62	0.51	0.72	0.40	0.63	0.63
1H0D.C	17	0.69	0.37	0.46	0.59	0.48	0.64
1GC1.G	11	0.70	0.57	0.57	0.53	0.55	0.52
1RZK.G	12	0.44	0.45	0.66	0.56	0.57	0.49
1K4D.C	13	0.77	0.77	0.70	0.63	0.86	0.73
1IC5.Y	16	0.46	0.49	0.60	0.46	0.40	0.47
1YQV.Y	14	0.67	0.44	0.79	0.50	0.77	0.78
1G7H.C	16	0.76	0.72	0.70	0.63	0.78	0.75
1G9M.G	12	0.59	0.62	0.48	0.52	0.66	0.43
1RZJ.G	11	0.46	0.52	0.67	0.53	0.60	0.49
1QFU.A	20	0.58	0.38	0.56	0.52	0.40	0.51
1IC4.Y	18	0.58	0.54	0.55	0.54	0.44	0.51
1EGJ.A	12	0.64	0.63	0.55	0.58	0.65	0.59
1A2Y.C	15	0.77	0.68	0.55	0.62	0.76	0.63
1G7I.C	15	0.67	0.79	0.69	0.54	0.81	0.66
1XIW.A	17	0.60	0.52	0.63	0.61	0.69	0.60
1TQB.A	18	0.70	0.43	0.69	0.70	0.59	0.66
1NDM.C	18	0.65	0.59	0.61	0.54	0.46	0.60

1FE8.A	19	0.67	0.63	0.69	0.57	0.69	0.72
1NBZ.C	19	0.62	0.63	0.61	0.58	0.60	0.48
2JEL.P	15	0.60	0.55	0.63	0.57	0.59	0.65
1BJ1.W	16	0.72	0.71	0.60	0.59	0.77	0.59
1IQD.C	16	0.82	0.74	0.69	0.56	0.84	0.70
1NBY.C	19	0.68	0.56	0.48	0.50	0.52	0.51
1TZH.V	13	0.73	0.71	0.75	0.67	0.88	0.81
1BQL.Y	13	0.53	0.59	0.68	0.50	0.74	0.72
1G9N.G	12	0.63	0.51	0.71	0.51	0.68	0.65
1G7J.C	15	0.64	0.74	0.57	0.56	0.75	0.55
1IC7.Y	17	0.59	0.38	0.50	0.55	0.31	0.48
1NDG.C	21	0.61	0.63	0.66	0.46	0.57	0.66
1TPX.A	18	0.65	0.43	0.67	0.58	0.53	0.51
1BVK.C	16	0.63	0.72	0.67	0.57	0.73	0.66
1KIP.C	15	0.58	0.72	0.58	0.53	0.77	0.62
1J1X.Y	19	0.60	0.57	0.56	0.46	0.41	0.48
1ORS.C	10	0.81	0.75	0.69	0.62	0.71	0.62
2HMI.B	9	0.91	0.73	0.61	0.64	0.56	0.62
1TY6.A (2VDN.A)	19	0.73	0.60	0.64	0.53	0.77	0.61
1MHP.A	16	0.75	0.53	0.72	0.54	0.83	0.80
1FNS.A	12	0.71	0.70	0.43	0.55	0.69	0.64
1J1P.Y	20	0.55	0.44	0.48	0.51	0.35	0.35
1KIQ.C	15	0.73	0.79	0.72	0.56	0.84	0.75
1DZB.X	18	0.69	0.45	0.49	0.53	0.39	0.47
1MEL.L	22	0.53	0.47	0.51	0.47	0.41	0.35
1OSP.O	20	0.58	0.56	0.60	0.54	0.65	0.54
1JPS.T	21	0.63	0.48	0.52	0.52	0.51	0.70
1OAZ.A	16	0.73	0.81	0.59	0.58	0.76	0.72
1C08.C	17	0.65	0.50	0.43	0.50	0.40	0.43
1G7L.C	15	0.77	0.80	0.69	0.58	0.84	0.69
1JHL.A	11	0.74	0.82	0.73	0.57	0.87	0.65
1AR1.B	15	0.62	0.66	0.59	0.45	0.75	0.72
1MLC.E	16	0.74	0.50	0.72	0.54	0.75	0.69
1CZ8.W	16	0.55	0.57	0.59	0.64	0.70	0.62
1G7M.C	15	0.61	0.72	0.66	0.60	0.80	0.73
1EO8.A	17	0.73	0.59	0.63	0.64	0.57	0.65
1KIR.C	14	0.69	0.77	0.68	0.59	0.81	0.63
1FJ1.F	17	0.69	0.62	0.47	0.45	0.76	0.60
1OTS.A	9	0.86	0.57	0.44	0.54	0.68	0.46
1DQJ.C	21	0.52	0.55	0.51	0.48	0.43	0.56
1N8Z.C	17	0.80	0.80	0.61	0.54	0.71	0.70
1J1O.Y	19	0.55	0.59	0.60	0.48	0.42	0.50
1LK3.A	18	0.53	0.44	0.68	0.44	0.67	0.58
1FDL.Y	14	0.75	0.77	0.67	0.67	0.83	0.63
1JRH.I	15	0.76	0.55	0.60	0.57	0.61	0.67
1FSK.A	17	0.47	0.52	0.46	0.52	0.50	0.41

Once again, in this section we have shown the prediction performance of machine learning models trained and tested on feature sets derived from single protein structures. As mentioned previously, this methodology limits the amount of training data available for the models. Thus, to study the impact of increasing the size of the training set - the number of residues trained on, we trained models that combined information from multiple protein complexes. The same 7-component feature vector from above was used here but models were trained with 15, 30, 45, 60, and 65 protein structure complexes, respectively. Results for these machine learning experiments are displayed in Table 25.

Table 23 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 7-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.74	0.60	0.64	0.55	0.64	0.64	0.92	0.92	0.91	0.86	0.85	0.92
30	0.80	0.60	0.71	0.59	0.65	0.67	0.91	0.90	0.90	0.85	0.75	0.90
45	0.80	0.59	0.73	0.60	0.62	0.66	0.92	0.91	0.90	0.86	0.86	0.91
60	0.80	0.62	0.72	0.60	0.63	0.67	0.92	0.91	0.91	0.86	0.78	0.91
65	0.80	0.62	0.73	0.61	0.63	0.67	0.92	0.91	0.91	0.86	0.79	0.91

The model evaluation for these machine learning experiments were further examined to understand the performance of these models. Therefore, a confusion matrix was generated for one of the models. This is shown in Figure 22 below.

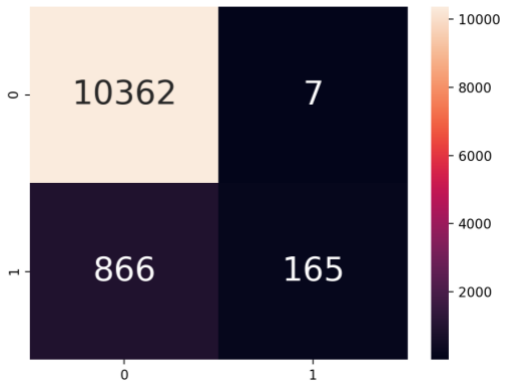


Figure 22 Confusion matrix for random forest model, trained on a 7-component feature set with structure information from 65 protein complexes. 0 is non-epitope (negative) and 1 is epitope (positive).

Figure 22 shows that the random forest model produces 10362 true negatives, 866 false negatives, 165 true positives, and 7 false positives. Therefore, in this full dataset there are 1031 epitope residues and 10369 non-epitope residues. This dataset also suffers from class imbalance, to address this issue we will train models on balanced training sets with an equal number of epitope and non-epitope residues.

3.4.4 Prediction Results for Balanced Delaunay-derived Feature Vectors

Feature sets described in section 3.4.2 were balanced according to the methods of section 3.4.1. This yielded five feature sets with 438, 922, 1422, 1896, and 2062 residues, respectively each with an equal number of epitope and non-epitope residues. The results for these machine learning experiments are shown in Table 26.

Table 24 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 7-component feature vector on balanced sets with default parameters and tested using 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.73	0.61	0.67	0.60	0.64	0.65	0.68	0.57	0.63	0.60	0.59	0.61
922	0.76	0.61	0.64	0.63	0.66	0.70	0.69	0.60	0.62	0.64	0.60	0.63
1422	0.77	0.60	0.70	0.63	0.63	0.67	0.69	0.58	0.64	0.63	0.59	0.63
1896	0.77	0.62	0.68	0.62	0.63	0.67	0.68	0.59	0.63	0.62	0.58	0.63
2062	0.78	0.62	0.70	0.62	0.63	0.69	0.70	0.60	0.64	0.62	0.59	0.64

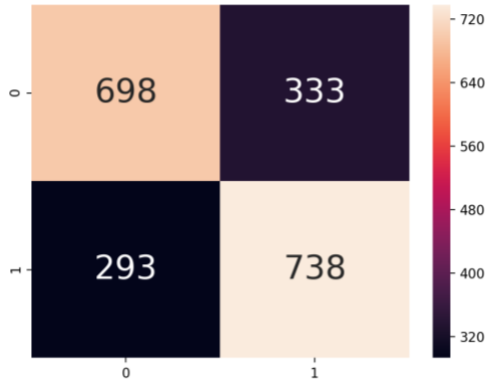


Figure 23 Confusion matrix for random forest model, trained on a 7-component feature set with structure information from 2062 amino acid residues. 0 is non-epitope (negative) and 1 is epitope (positive).

The results obtained for the models trained on the balanced sets are more reasonable than that for the unbalanced sets and they show the true performance of the models. The best performing model for these experiments was a random forest model that was trained on 2062 residues from the interacting chain of various bound antigen-antibody protein structures. The 2062 residues were balanced with an equal number of epitope and non-epitope residues. The model achieved 0.78 and 0.70 AUC and accuracy, respectively. This model produced 698 true negatives, 293 false negatives, 738 true positives, and 333 false positives.

3.5 Conformational B-cell Epitope Prediction (Structure-based) Discussion

The initial findings of this chapter provide some promise for the future of this structure-based approach for training machine learning models of conformational B-cell epitopes because some of the models achieved AUC values that compare quite favorably with the results from the original DiscoTope paper. For example, the AUC value for complex 1MHP.A for DiscoTope is about 0.76 and for our approach is about 0.79. DiscoTope produces an average AUC, for prediction on the evaluation sets of about 0.6, when only predicting based on the raw epitope log-odds. Our approach produces an average AUC of about 0.6 under these same conditions. However, DiscoTope2.0 produces an average AUC of about 0.7, when only predicting based on the raw epitope log-odds scores. These results show that our approach to conformational epitope prediction is comparable to the original DiscoTope but is lacking in performance compared to DiscoTope2.0. However, we do define residue neighbors/contacts in a more robust manner than using an arbitrary distance cut-off or function, which is favorable over DiscoTope.

CHAPTER FOUR

4.1 Combined Sequence and Structure Approach for Epitope Prediction

Thus far, we have focused on deriving feature sets to train epitope machine learning models using either protein sequence or structure information. The next logical step will be to assess the performance of machine learning models trained on feature sets composed of a combination of sequence and structure-derived features. For the initial explorations into a combined sequence and structure approach for epitope prediction, the use of the identity of the central residue was added to the feature set. In addition to the seven Delaunay tessellation derived structural features (summed four-body statistical contact pseudo-potentials) used to represent each residue in the dataset, the identity of the central residue using the traditional twenty letter amino acid alphabet was added as a component. Machine learning experiments were performed according to the methods described in chapters 2 and 3 and results are presented in section 4.1.1.

4.1.1 Results for the 8-component Feature Vector (unbalanced training sets)

Models containing residue feature set information from 15, 30, 45, 60, and 65 proteins respectively were used in these machine learning experiments. The results for these machine learning experiments are displayed in table 27.

Table 25 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using an 8-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.78	0.64	0.61	0.54	0.66	0.63	0.92	0.92	0.91	0.86	0.61	0.92
30	0.82	0.66	0.73	0.60	0.67	0.71	0.91	0.90	0.90	0.86	0.60	0.90
45	0.83	0.66	0.74	0.62	0.65	0.72	0.92	0.91	0.91	0.86	0.56	0.91
60	0.83	0.68	0.74	0.62	0.67	0.72	0.93	0.91	0.91	0.87	0.58	0.91
65	0.83	0.68	0.75	0.63	0.67	0.73	0.93	0.91	0.91	0.87	0.57	0.91

This set of machine learning experiments, utilizing an 8-component feature vector, including both sequence and structure information produced models that achieved consistently high AUC values. The models consisting of 45, 60, and 65 protein structures achieved AUC values of about 0.83 with the random forest algorithm.

For the next set of machine learning experiments the central residue was represented, not by the twenty-letter amino acid alphabet, but by a 3-letter reduced alphabet scheme. The reduced alphabet scheme used here is reduced alphabet 5 from the sequence-based epitope prediction approach from chapter 2. Reduced alphabet 5 was arbitrarily chosen as a representative reduced amino acid alphabet scheme. The three-letter (B, J, and U) reduced alphabet scheme is shown below.

B = [L, A, S, G, V, T, I, P, M, C]

J = [E, K, R, D, N, Q, H]

U = [F, Y, W]

Once again, machine learning models of 15, 30, 45, 60, and 65 protein structures were trained. The results for these machine learning experiments are displayed in table 28.

Table 26 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using an 8-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.74	0.60	0.64	0.55	0.63	0.63	0.92	0.92	0.91	0.86	0.84	0.92
30	0.80	0.62	0.72	0.60	0.65	0.68	0.92	0.90	0.89	0.86	0.76	0.90
45	0.81	0.62	0.73	0.62	0.63	0.68	0.92	0.91	0.90	0.86	0.81	0.91
60	0.81	0.64	0.73	0.62	0.65	0.69	0.92	0.91	0.91	0.87	0.79	0.91
65	0.81	0.64	0.74	0.62	0.65	0.69	0.93	0.91	0.92	0.87	0.79	0.91

The model performance of machine learning models trained on the feature set containing the central residue represented by a twenty-letter amino acid alphabet versus the reduced 3-letter alphabet are overall quite similar. However, models trained on the feature set containing the central residue represented by a twenty-letter amino acid alphabet performed either better or equal to the second feature set for all algorithms tested, in terms of an average AUC for all 5 subsets. Subsequent analyses will test the viability of including a reduced amino acid alphabet representation of residue identities in our feature vector.

4.1.2 Results for the 14-component feature vector (unbalanced training sets)

Additional experiments were performed to evaluate the performance of machine learning models trained on a 14-component feature vector. The feature vector components consisted of the summed four-body statistical contact pseudo-potential for

the central residue and its six nearest neighbors as well as the identity of the central residue and its six nearest neighbors, using a twenty-letter alphabet. The results for these machine learning experiments are displayed in table 29.

Table 27 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 14-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.81	0.71	0.68	0.60	0.69	0.71	0.92	0.92	0.91	0.88	0.70	0.92
30	0.86	0.72	0.77	0.63	0.70	0.78	0.92	0.90	0.91	0.87	0.69	0.90
45	0.86	0.72	0.78	0.66	0.70	0.77	0.93	0.90	0.91	0.88	0.67	0.91
60	0.85	0.73	0.77	0.66	0.71	0.77	0.93	0.91	0.92	0.88	0.67	0.91
65	0.85	0.74	0.78	0.67	0.71	0.78	0.93	0.91	0.91	0.89	0.66	0.91

The 14-component feature vector has trained our best performing discontinuous B-cell epitope prediction models thus far. The models trained on 30 and 45 structures, using the random forest algorithm, both achieved AUC values of about 0.86. Additional experiments and analyses will be performed to further improve this feature vector to train even better performing models.

Next, models trained on 15, 30, 45, 60, and 65 protein structures were trained once again. The feature set consisted of the summed four-body statistical contact pseudo-potential for the central residue and its six nearest neighbors as well as the identity of the central residue and its six nearest neighbors. However, for this feature set a reduced three-letter amino acid alphabet was used to represent the residue identities. The reduced amino acid alphabet scheme used is described in section 4.1.1. Results for these machine learning experiments are displayed in table 30.

Table 28 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 14-component feature vector with default parameters and tested using 10-fold stratified cross-validation.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.72	0.57	0.67	0.54	0.61	0.62	0.92	0.92	0.91	0.86	0.79	0.92
30	0.82	0.60	0.76	0.59	0.64	0.71	0.91	0.90	0.90	0.85	0.74	0.90
45	0.83	0.61	0.75	0.63	0.63	0.73	0.92	0.91	0.90	0.86	0.78	0.91
60	0.83	0.63	0.76	0.62	0.64	0.74	0.93	0.91	0.91	0.87	0.76	0.91
65	0.83	0.64	0.76	0.63	0.64	0.74	0.93	0.91	0.92	0.87	0.76	0.91

The prediction performance of these machine learning models compared to the previous set that used the full twenty-letter amino acid alphabet to represent the residue identities instead of the reduced three-letter amino acid alphabet, decreased, in terms of average AUC for all five subsets for all algorithms tested. This points to the conclusion that representing the residue identities using the traditional twenty-letter amino acid alphabet is better suited for this feature vector and this prediction task.

4.2 Adding a Surface Measure Component to the Feature Vector

For several years it has been known that the antigenicity of certain polypeptide-chain segments is directly related to their surface exposure, these segments that display tremendous surface exposure or protrusion are readily available for contact with antigen-combining sites [90][96][97]. Therefore, it was hypothesized that adding a definition of surface exposure to the feature vector for each residue in the dataset would improve the performance of the predictive models.

The next set of machine learning experiments involved the addition of a surface measure to the feature set. The relative solvent accessibility (RSA) was calculated for each of the residues in the antibody-antigen interacting chain for all of the protein

complexes in our dataset. To do this, the DSSP files for each of the PDB structures was downloaded using the DSSP website tool available from, http://www.cmbi.ru.nl/xssp/. Next, a Python script was written to parse the DSSP output and extract the ACC value for each of the residues. DSSP defines the ACC value as the number of water molecules in contact with the residue *10 or as the residue water exposed surface in Angstrom **2.

Next, the ACC value was converted to RSA by dividing the ACC value by the total surface area of each amino acid residue (TSA). The TSA values were taken from [98].

Table 29 Total surface area (TSA) for each amino acid calculated for the residue X in the tripeptide G-X-G. The value is calculated in $Å^2$. The values were obtained from the paper, *The Nature of the Accessible and Buried Surfaces in Proteins* by: C. Chotia [98].

Amino Acid	Total Surface Area (TSA)
A	115
R	225
D	150
N	160
C	135
E	190
Q	180
G	75
H	195
I	175
L	170
K	200
M	185
F	210
P	145
S	115
T	140
W	225
Y	230
V	155

The calculated RSA values for each of the residues in the dataset were added as a component of the feature vector containing the summed four-body statistical contact

pseudo-potential for the central residue and its six nearest neighbors and the identity of the central residue and its six nearest neighbors represented by either a 20-letter or 3-letter alphabet (B, J, U). Note that two of the residues in the full dataset had an RSA value of NAN. For these two residues the mean RSA value for all of the residues in the subset was used to replace the NAN value for use in the machine learning algorithms. Models were trained on subsets containing 15, 30, 45, 60, and 65 proteins and feature vectors containing 15 components. The results for these machine learning experiments are displayed in tables 32 and 33.

Table 30 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 15-component feature vector with default parameters and tested using 10-fold stratified cross-validation. Residue identities were represented with a 20-letter amino acid alphabet.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.84	0.77	0.69	0.62	0.70	0.78	0.93	0.92	0.92	0.89	0.71	0.92
30	0.90	0.78	0.77	0.67	0.72	0.85	0.93	0.90	0.91	0.89	0.71	0.90
45	0.91	0.78	0.79	0.69	0.72	0.84	0.93	0.91	0.91	0.90	0.68	0.91
60	0.90	0.79	0.78	0.71	0.72	0.83	0.94	0.91	0.92	0.90	0.68	0.91
65	0.90	0.80	0.78	0.72	0.73	0.84	0.94	0.91	0.92	0.90	0.68	0.91

The addition of the RSA value surface measure component to the feature vector improved the results. We achieved our highest AUC value, 0.91, thus far with the random forest model trained on the 45-protein structure subset.

Table 31 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 15-component feature vector with default parameters and tested using 10-fold stratified cross-validation. Residue identities were represented with a 3-letter amino acid alphabet.

Number of Structures	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
15	0.79	0.68	0.66	0.59	0.68	0.66	0.92	0.92	0.91	0.87	0.81	0.92
30	0.87	0.72	0.76	0.65	0.72	0.79	0.92	0.90	0.90	0.87	0.78	0.90
45	0.88	0.72	0.76	0.67	0.71	0.80	0.93	0.91	0.91	0.88	0.81	0.91
60	0.88	0.73	0.77	0.67	0.71	0.80	0.94	0.91	0.92	0.89	0.80	0.91
65	0.89	0.74	0.77	0.66	0.72	0.82	0.94	0.91	0.92	0.88	0.79	0.91

Overall, the model performance has improved with the addition of the surface measure component. A random forest model trained on the 45-protein structure subset with the 20-letter alphabet representing the residue identities achieved our highest AUC value, 0.91, thus far.

4.3 Implications of several cut-off values for RSA to define surface/buried residues

The determination of buried/exposed (surface) residues can come down to an arbitrary threshold definition. RSA values are compared to the chosen threshold and the classification of buried or exposed can be made for the given residue in the protein structure. If the RSA value for the residue is below the threshold then it is classified as buried, otherwise it is classified as exposed. Chen and Zhou 2005 [99] defined their threshold as 20%. Wu et al. 2017 [100] defined their threshold as 25%. Zhang et al. 2009 [101] avoided the arbitrary threshold definition problem by using a range of thresholds from 0-95% with a 5% step. Miller et al. 1987 [102] used a threshold of just 5%. For this work, several thresholds were chosen and used to determine a binary classification of buried or exposed residues. We will use thresholds from 5-50% with a step of 5%. The

binary classification was used to replace the calculated RSA value in the feature vectors from section 4.2. We chose to test this approach using the 20-letter amino acid alphabet to represent the residue identities and the random forest algorithm, a configuration that has been proven to produce optimal prediction performance in previous experiments. Note the NANs were filled with the mode for the column. Results for this approach are displayed in table 34.

Table 32 AUC values for machine learning models trained on a 15-component feature vector (with a 20-letter alphabet representing the residue identities). Results are shown for the subsets that contain 15, 30, 45, 60, and 65 proteins, respectively. The new feature added here is a binary classification of either buried or exposed residue determined by a step-wise threshold definition. Random forest with default parameters and 10-fold stratified cross-validation was used to perform the machine learning experiments.

	RSA Threshold									
Number of Structures	**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**
	AUC									
15	0.82	0.83	0.83	0.82	0.84	0.85	0.84	0.84	0.85	0.83
30	0.88	0.88	0.89	0.89	0.90	0.89	0.90	0.90	0.90	0.89
45	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.89	0.88
60	0.87	0.87	0.88	0.88	0.88	0.89	0.89	0.89	0.89	0.88
65	0.87	0.87	0.88	0.89	0.88	0.89	0.89	0.89	0.88	0.88

After these experiments, it seems as though the particular threshold chosen to define buried/surface residues does not influence the predictive performance of the machine learning models dramatically. Each threshold from 5% to 50% produced similar prediction results for each subset, evaluated in terms of AUC. Using the threshold with binary classification of buried/exposed residues did not outperform the machine learning models trained with the raw RSA values, though they were quite similar.

4.4 Discontinuous B-cell epitope prediction (balanced training sets)

Our discontinuous B-cell epitope dataset consists of annotated epitope and non-epitope residues. In the full dataset (65 antigen-antibody bound structures) there are a total of 1031 epitope residues and 10369 non-epitope residues. Thus far, all of the machine learning experiments for discontinuous B-cell epitope prediction with a combined sequence and structure feature set have been conducted on unbalanced datasets of epitope/non-epitope residue feature sets. This poses a class imbalance problem that may bias the performance evaluation of our models. Consequently, it will be important to study models that have been trained on balanced datasets, which is the focus of this section.

For each of the five subsets of our dataset, the number of epitope residues was calculated. Then, all of the non-epitope residues in the full set were randomized and samples were taken to create balanced sets of equal numbers of epitope and non-epitope sequences. Therefore, each of the negative sets were unique and randomized. The resulting subsets consisted of 438, 922, 1422, 1896, and 2062 residues, respectively. Section 4.4.1 presents the results of the balanced set experiments.

4.4.1 Discontinuous B-cell epitope prediction (balanced training sets) Results

Table 33 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using an 8-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors and the identity of the central residue using a 20-letter alphabet) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.76	0.65	0.66	0.60	0.66	0.66	0.68	0.62	0.61	0.60	0.61	0.61
922	0.78	0.65	0.65	0.62	0.67	0.71	0.71	0.61	0.62	0.63	0.62	0.64
1422	0.79	0.64	0.72	0.63	0.64	0.70	0.71	0.62	0.65	0.64	0.60	0.65
1896	0.80	0.67	0.70	0.63	0.66	0.71	0.72	0.62	0.64	0.64	0.63	0.65
2062	0.80	0.68	0.72	0.64	0.67	0.73	0.72	0.64	0.65	0.64	0.63	0.67

Table 34 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using an 8-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors and the identity of the central residue using a 3-letter alphabet) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.73	0.61	0.65	0.59	0.64	0.65	0.65	0.59	0.62	0.60	0.60	0.60
922	0.77	0.61	0.66	0.63	0.66	0.70	0.70	0.58	0.62	0.64	0.59	0.63
1422	0.78	0.62	0.70	0.63	0.64	0.68	0.71	0.59	0.64	0.62	0.59	0.64
1896	0.78	0.64	0.69	0.63	0.64	0.68	0.69	0.60	0.62	0.63	0.60	0.63
2062	0.79	0.65	0.71	0.64	0.65	0.70	0.71	0.60	0.65	0.64	0.60	0.65

The best performing models, in terms of AUC and accuracy, for the 8-component feature vector experiments were random forest models trained on 1896 and 2062 residues. The feature vector with the 20-letter amino acid alphabet residue identity slightly outperformed the one with the 3-letter alphabet. The best AUC value for the 20-letter set was 0.80 and 0.79 for the 3-letter set.

Table 35 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 14-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors and the identity of the central residue and its 6 nearest neighbors using a 20-letter alphabet) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.80	0.69	0.70	0.66	0.66	0.71	0.72	0.63	0.65	0.66	0.60	0.63
922	0.81	0.70	0.68	0.64	0.67	0.76	0.73	0.66	0.64	0.66	0.64	0.66
1422	0.82	0.71	0.73	0.63	0.69	0.77	0.74	0.66	0.67	0.64	0.65	0.68
1896	0.82	0.72	0.73	0.65	0.71	0.77	0.73	0.67	0.66	0.65	0.65	0.69
2062	0.83	0.73	0.75	0.67	0.71	0.79	0.74	0.68	0.68	0.67	0.66	0.71

Table 36 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 14-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors and the identity of the central residue and its 6 nearest neighbors using a 3-letter alphabet) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.76	0.58	0.65	0.63	0.61	0.66	0.70	0.56	0.62	0.65	0.58	0.61
922	0.78	0.60	0.68	0.63	0.64	0.70	0.68	0.57	0.64	0.64	0.59	0.64
1422	0.77	0.61	0.73	0.63	0.62	0.70	0.71	0.58	0.67	0.63	0.60	0.65
1896	0.79	0.64	0.72	0.62	0.64	0.70	0.71	0.60	0.66	0.63	0.60	0.65
2062	0.81	0.64	0.74	0.65	0.64	0.74	0.73	0.61	0.67	0.66	0.60	0.67

The best performing model, in terms of AUC and accuracy, for the 14-component feature vector experiments was a random forest model trained on 2062 residues. The feature vector with the 20-letter amino acid alphabet residue identities outperformed the one with the 3-letter alphabet. The best AUC value for the 20-letter set was 0.83 and 0.81 for the 3-letter set. Moreover, the addition of the new components did improve the performance of the models and we improved the performance of our best performing model by 0.03 AUC.

Table 37 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 15-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors, identity of the central residue and its 6 nearest neighbors using a 20-letter alphabet, and raw RSA values) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.84	0.74	0.70	0.63	0.67	0.75	0.75	0.68	0.65	0.67	0.61	0.65
922	0.86	0.77	0.68	0.67	0.68	0.80	0.77	0.70	0.64	0.67	0.65	0.70
1422	0.88	0.78	0.74	0.70	0.70	0.82	0.79	0.72	0.67	0.70	0.66	0.73
1896	0.87	0.78	0.74	0.68	0.72	0.81	0.78	0.71	0.66	0.68	0.66	0.74
2062	0.88	0.80	0.75	0.71	0.72	0.84	0.79	0.73	0.68	0.71	0.68	0.75

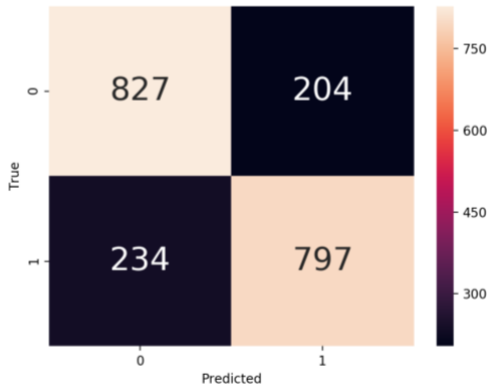


Figure 24 Confusion matrix heatmap for Random Forest predictions on balanced dataset consisting of 2062 residues. 0 represents the negative class (non-epitope residues) and 1 represents the positive class (epitope residues).

Table 38 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 15-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors, identity of the central residue and its 6 nearest neighbors using a 3-letter alphabet, and raw RSA values) with default parameters and 10-fold stratified cross-validation.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.81	0.69	0.64	0.62	0.70	0.69	0.72	0.64	0.61	0.61	0.64	0.63
922	0.85	0.72	0.69	0.66	0.71	0.76	0.76	0.66	0.64	0.68	0.65	0.67
1422	0.86	0.73	0.73	0.69	0.71	0.77	0.76	0.67	0.67	0.69	0.65	0.69
1896	0.85	0.73	0.73	0.68	0.71	0.77	0.76	0.66	0.67	0.67	0.64	0.69
2062	0.88	0.74	0.75	0.72	0.73	0.80	0.78	0.68	0.68	0.71	0.66	0.72

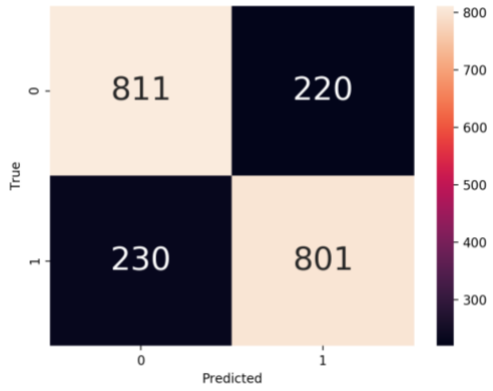


Figure 25 Confusion matrix heatmap for Random Forest predictions on balanced dataset consisting of 2062 residues. 0 represents the negative class (non-epitope residues) and 1 represents the positive class (epitope residues).

The best performing models, in terms of AUC and accuracy, for the machine learning experiments conducted on the 15-component feature vector that consisted of the summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors, the identity of the central residue and its 6 nearest neighbors using a 20-letter alphabet, and raw RSA values, were random forest models trained on the 1422 and 2062 residue subsets. These models both achieved an AUC value and accuracy of

0.88 and 0.79, respectively. These models also achieved the best performance of all models to date. These results point to the value of our 15-component feature vector for predicting residues in a discontinuous B-cell epitope. Also, the balanced sets provide a fair estimate of the model performance, seen in both the AUC and accuracy scores.

4.4.2 15-component Feature Vector (Balanced) Control Set

To help confirm the results observed for discontinuous B-cell epitope prediction, a control feature set was created. This control was generated by randomizing the class labels of the best performing feature set (15-component with 20-letter alphabet). The goal of this experiment is to test the performance of the machine learning models on a feature set the randomly assigns the vectors to a class label. The hypothesis is that this would yield performance equal to random guessing.

Table 39 Discontinuous B-cell epitope prediction model evaluation quantified as area under the ROC curve (AUC) and accuracy. Models were trained using a 15-component feature vector (summed four-body statistical contact pseudo-potential for the central residue and its 6 nearest neighbors, identity of the central residue and its 6 nearest neighbors using a 20-letter alphabet, and raw RSA values) with default parameters and 10-fold stratified cross-validation. The original class labels assigned in this feature set were randomized to produce the control set.

Number of Residues	AUC						Accuracy					
	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine	Random Forest	Logistic Regression	K Neighbors	Decision Tree	Gaussian Naïve Bayes	Support Vector Machine
438	0.51	0.46	0.53	0.50	0.48	0.45	0.45	0.50	0.48	0.45	0.49	0.36
922	0.47	0.52	0.50	0.47	0.52	0.47	0.48	0.51	0.51	0.49	0.49	0.49
1422	0.50	0.50	0.50	0.48	0.48	0.46	0.51	0.52	0.51	0.48	0.52	0.51
1896	0.51	0.49	0.51	0.51	0.49	0.50	0.50	0.51	0.50	0.51	0.52	0.51
2062	0.52	0.54	0.51	0.52	0.53	0.51	0.49	0.50	0.50	0.50	0.50	0.49

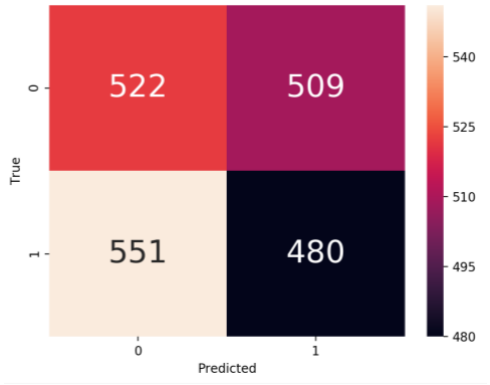


Figure 26 Confusion matrix heatmap for Random Forest predictions on balanced dataset control consisting of 2062 residues. 0 represents the negative class (non-epitope residues) and 1 represents the positive class (epitope residues).

Randomized class labels are used as a control to determine if the signal in the feature set is in fact due to the nature of the feature set, not by chance. Randomized class labels associate feature vectors with class labels that are not true, this will hopefully interfere with the machine learning algorithm. Therefore, about 50% accuracy is expected, equivalent to random guessing. The results of the control experiments are around the 0.50 AUC and accuracy random guessing mark. This confirms that machine learning models trained on a randomized version of our best performing feature vector cannot perform better than random guessing when trying to predict epitopic resides. However, our feature vector when not randomized can produce results far better than random guessing. Giving validity to our models.

4.4.3 Balanced sets with binary surface/buried classification

Table 40 AUC values for machine learning models trained on a 15-component feature vector (with a 20-letter alphabet representing the residue identities). Results are shown for the subsets that contain 15, 30, 45, 60, and 65 proteins, respectively. The new feature added here is a binary classification of either buried or exposed residue determined by a step-wise threshold definition. Random forest with default parameters and 10-fold stratified cross-validation was used to perform the machine learning experiments.

	RSA Threshold									
Number of Residues	**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**
	AUC									
438	0.82	0.82	0.79	0.81	0.82	0.82	0.82	0.83	0.84	0.83
922	0.84	0.83	0.85	0.83	0.85	0.85	0.84	0.85	0.85	0.85
1422	0.84	0.84	0.85	0.85	0.86	0.86	0.87	0.87	0.86	0.86
1896	0.84	0.84	0.85	0.85	0.85	0.85	0.85	0.86	0.86	0.85
2062	0.84	0.86	0.85	0.86	0.86	0.87	0.87	0.87	0.87	0.86

Overall, the RSA threshold value did not seem to dramatically affect the performance of the machine learning models. Thresholds from 5 to 50% all appeared to be associated with models that achieved similar AUC values using the random forest algorithm. The best performing models using this feature vector achieved AUC values of 0.87, this was achieved using a 30, 35, 40, and 45% threshold value.

4.4.4 Discontinuous B-cell Epitope Prediction (balanced training sets) Discussion

The results of our machine learning experiments, conducted on several feature sets and several subsets of the final dataset, are displayed in this section. The components of the eight feature vectors are listed below.

Feature Set 1: Seven components, summed four-body statistical contact pseudo-potential for every simplex in which the central residue participates and the summed four-body

statistical contact pseudo-potential for every simplex in which the central residue's six nearest-neighbors participate.

Feature Set 2: Feature Set 1 + identity of the central amino acid residue (20-letter alphabet).

Feature Set 3: Feature Set 1 + identity of the central amino acid residue (3-letter alphabet).

Feature Set 4: Feature Set 2 + identities of the central amino acids 6-nearest neighbors' residues (20-letter alphabet).

Feature Set 5: Feature Set 3 + identities of the central amino acids 6-nearest neighbors' residues (3-letter alphabet).

Feature Set 6: Feature Set 4 + RSA value of the residue

Feature Set 7: Feature Set 5 + RSA value of the residue

Feature Set 8: Control, Feature Set 6 with randomized class labels.

Subsets: S1 = 438, S2 = 922, S3 = 1422, S4 = 1896, S5 = 2062 residues

Our feature sets leverage both protein structural and sequence information and amino acid surface measures. This information is some of the most readily available for protein residues and make them a practical choice for training our machine learning models.

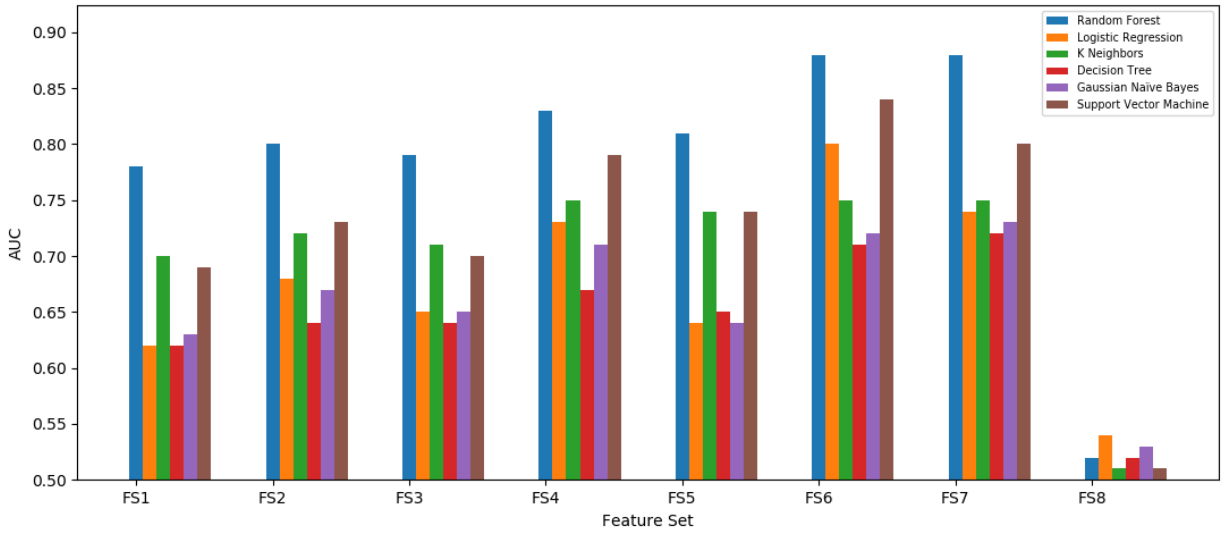


Figure 27 Comparison of AUC for several machine learning algorithms trained on different feature sets (FS1-FS8) and the entire final dataset of 2062 residues (subset 5).

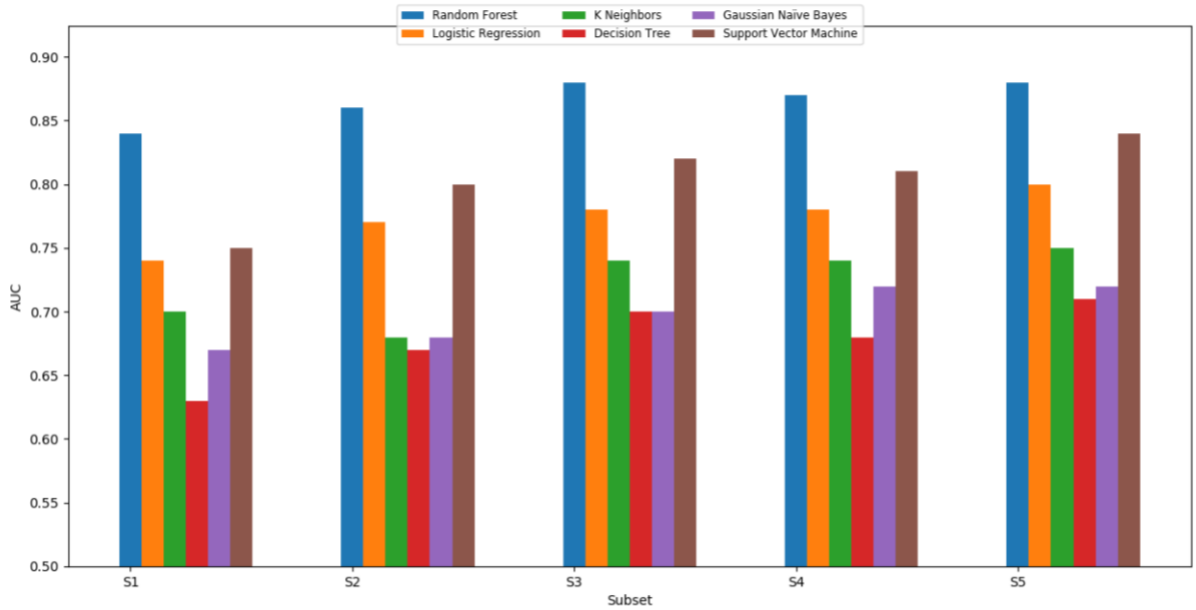


Figure 28 Comparison of AUC values for several machine learning algorithms trained on different subsets of the final dataset and feature set 6. Subsets S1, S2, S3, S4, and S5 are composed of 438, 922, 1422, 1896, and 2062 residues, respectively.

4.5 Tuning the Random Forest Model: Hyperparameter Optimization

We looked to boost the performance of our best performing model by conducting hyperparameter optimization. A default Random Forest classifier achieved an AUC of about 0.88 when trained on feature set 6, subset 5. A grid of value ranges was established for several Random Forest hyperparameters, random samples were taken for each hyperparameter and combined to define new Random Forest classifiers, stratified 10-fold cross-validation was used to determine the performance of each model based on AUC, and the best model, with its corresponding hyperparameter values, was determined. The optimal hyperparameters found by the random search was used to narrow the ranges of hyperparameter values for a more exhaustive grid search, which tries every combination of hyperparameters within the search space. When trained on the same training data, the new optimized model achieved an AUC of about 0.89, a minimal increase over the previous model, but may be significant.

4.6 Discontinuous B-cell Epitope Prediction on Independent Test Set

An independent test set was manually compiled from entries of the Immune Epitope Database (IEDB), available at https://www.iedb.org/. This dataset consisted of 10 experimentally determined discontinuous epitopes with available 3D structures. The tuned model from above was used to predict epitope and non-epitope residues in the test set. The test set was balanced to better evaluate performance. The model achieved an AUC of about 0.73.

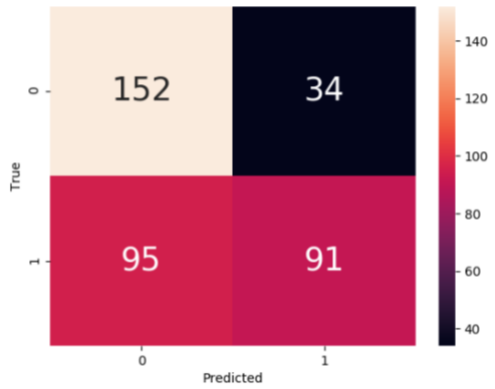


Figure 29 Confusion matrix heatmap for Random Forest predictions on balanced independent test set. 0 represents the negative class (non-epitope residues) and 1 represents the positive class (epitope residues).

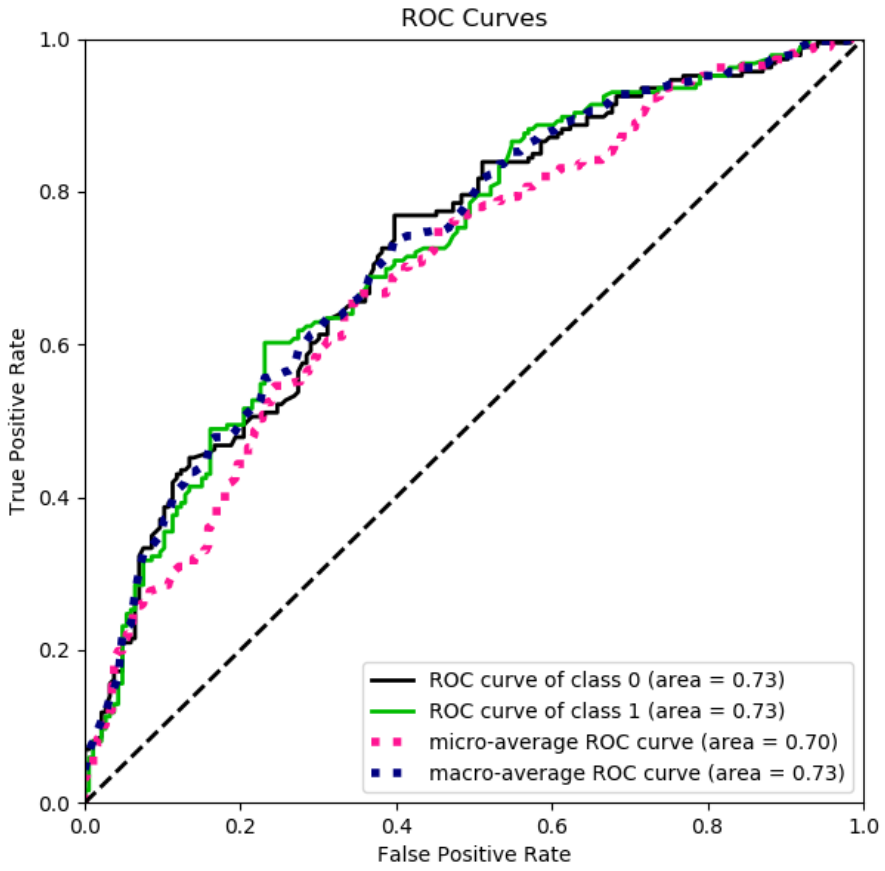


Figure 30 ROC curve for our tuned best performing Random Forest model on the independent test set.

The performance of our model on the independent test set is promising. Future testing will be needed on a larger test set to evaluate the performance of our model.

4.6.1 SARS-CoV-2 Independent Test Set

As another form of validation for the conformational B-cell epitope model discussed in this section, classification performance of the model on unseen SARS-CoV-2 conformational B-cell epitope data was evaluated. First, the testing set was manually compiled from entries of the IEDB. A search was performed for positive discontinuous B-cell epitopes found on the spike proteins of SARS-CoV-2. A total of 24 antigen-antibody bound structures with epitope annotations were collected to form the testing set. After class balancing, the dataset contained 428 epitope-participating and 428 non-epitope-participating residues (856 total). The model achieved an AUC of about 0.71.

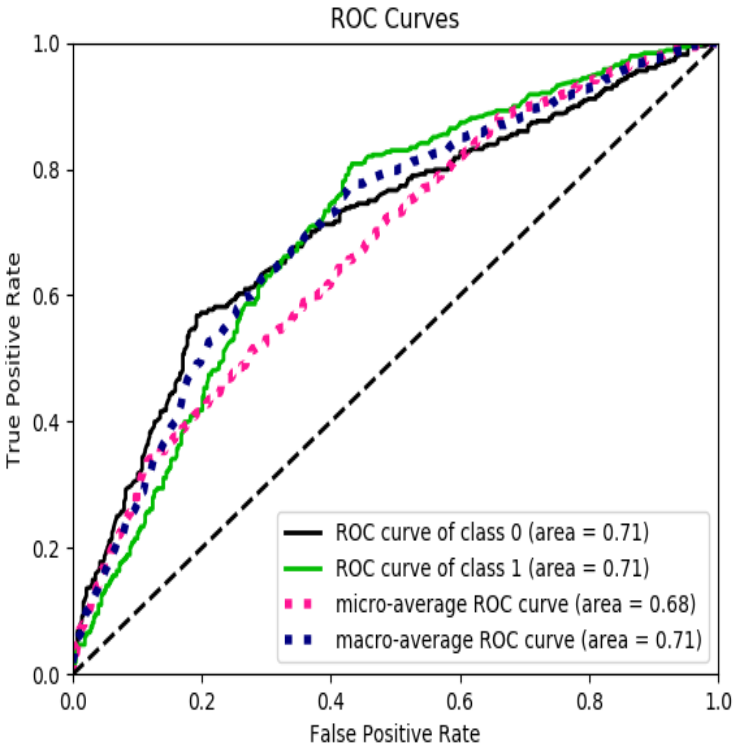


Figure 31 ROC curve for model predictions on the independent SARS-CoV-2 test set.

CHAPTER FIVE

5.1 TESSETOPE V1.0: A Web API for Linear and Conformational Epitope Prediction

5.1.1 Brief Introduction

Delaunay tessellation-based epitope prediction or TESSETOPE is a freely available web accessible application programming interface (API) that allows the client to upload protein data and returns a response in the form of an epitope prediction. For example, TESSETOPE's conformational B-cell epitope prediction tool accepts requests comprised of PDB, DSSP, and protein chain data, processes the input data, applies a trained Random Forest model to classify the residues in the protein chain as either epitopic or non-epitopic, and displays the prediction to the client as a hypertext transfer protocol (HTTP) response. TESSETOPE also includes tools for linear B-cell and linear T-cell epitope prediction which simply accept requests in the form of text files that hold protein primary structure (sequence) information.

The relatively simple user-interface and public availability of TESSETOPE makes it a convenient choice for researchers around the world. It allows for fast and efficient epitope prediction that can ultimately complement computational and experimental studies for vaccine and biotherapeutic design. The machine learning models within TESSETOPE, trained for prediction of linear B-cell epitope sequences, linear T-

cell epitope sequences, and residues in conformational B-cell epitopes, are those described in sections 2.3.3, 2.4.3, and 4.5, respectively.

5.1.2 Implementation

TESSETOPE was developed using Django (https://www.djangoproject.com/) and the Django REST framework (https://www.django-rest-framework.org/) in the Python programming language. It is freely available at http://omics.gmu.edu/tessetope/.

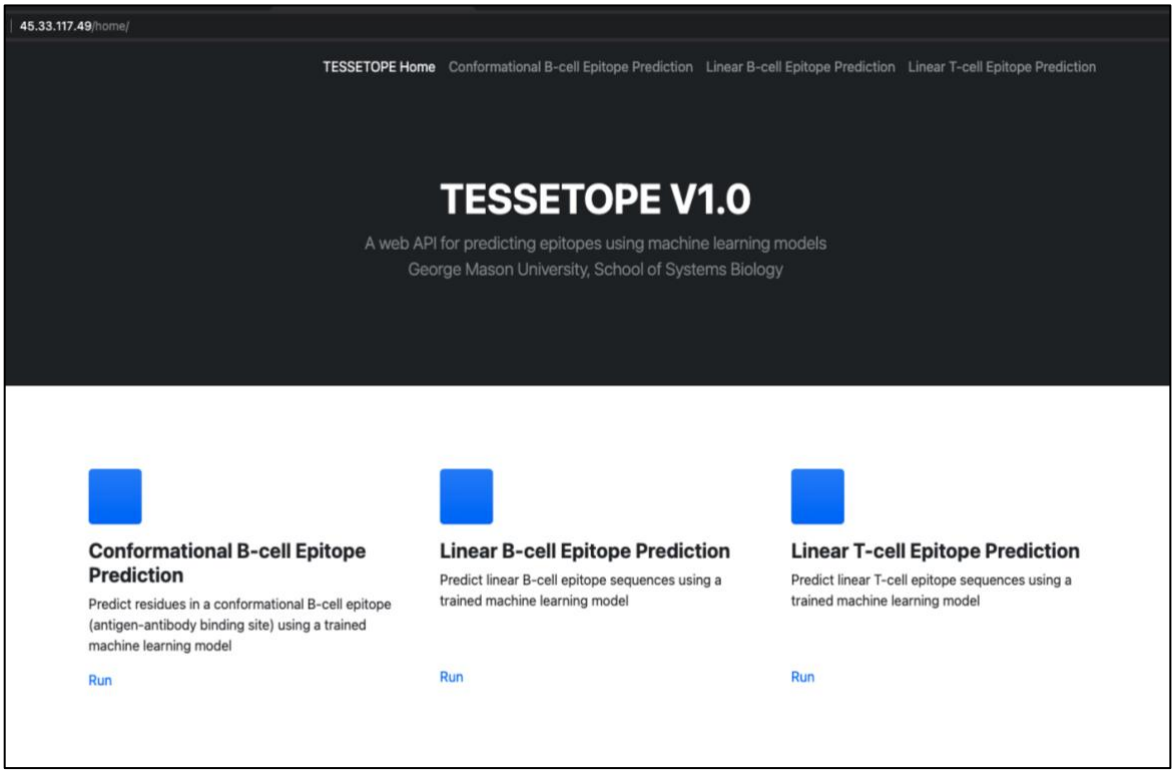


Figure 32 Screenshot of the TESSETOPE V1.0 home page. The home page is equipped with links to each of the three epitope prediction tools at the top of the page and through the blue "Run" buttons.

5.1.3 Conformational B-cell Epitope Prediction Example

To access the conformational B-cell epitope prediction tool, simply click on the "Conformational B-cell Epitope Prediction" links on the home page or go to the following URL: http://45.33.117.49/conf_ep_pred/run/conf_ep_pred/. To run the tool, the client must obtain the PDB and DSSP files for the antigen of interest, henceforth referred to as the query (TESSETOPE's Random Forest model for predicting conformational B-cell epitopes was trained on bound antigen-antibody structures). Copy and paste the data from both the query's PDB and DSSP files into one blank text file with the information from the PDB file preceding the DSSP information (note: all data from both files should be copied over), the client must be sure to save the created file with the *.txt* extension. Next, scroll to the bottom of the web page and upload the created file using the "Choose File" button of the "Pdb dssp" field. In the "Chain" field, type the query's protein chain of interest (note: only one protein chain can be sent as a request at this time). Finally, the client can click "POST" to send the request to the server to run the tool and output predictions.

To illustrate this implementation, the PDB and DSSP files for a SARS-CoV-2 spike protein in complex with the S2E12 neutralizing antibody Fab fragment (PDB ID 7K4N) were copied into a single text file according to the specifications described above, saved as 7k4n.txt, uploaded, and chain "A" was chosen as the query's protein chain of interest.

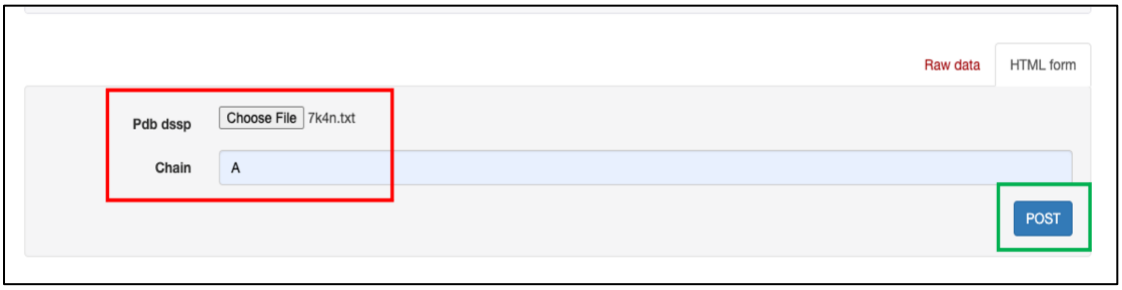


Figure 33 Screenshot of the request to TESSETOPE's conformational B-cell epitope prediction tool, 7k4n.txt and chain A (red box). The "POST" button is used to send the request to the server, run the tool, and obtain the predictions (green box).

Figure 34 shows the HTTP response sent to the client after the predictions are made. Each residue in the query's chain is listed, identified by the residue number and three-letter amino acid abbreviation, along with its predicted classification, either "non-epitope" or "epitope".

Figure 34 Screenshot of TESSETOPE's conformational B-cell epitope prediction tool's HTTP response to the client; the prediction output. Each residue in the query's protein chain of interest has a prediction of "epitope residue" or "non-epitope residue".

5.1.4 Linear B-cell Epitope Prediction Example

For both implementations of linear epitope prediction, TESSETOPE accepts requests in the form of text files that must only contain the complete primary structure (sequence of amino acids) of the query. Once the input text file is uploaded to the API using the "Choose File" button of the "File" field, click "POST" to submit the request, run the linear epitope prediction tools, and display the results.

To illustrate this implementation, a known linear B-cell epitope sequence was obtained from the IEDB, epitope ID 103097, and saved to the text file protein_sequence.txt. The 40 amino acid residue sequence (`DAEFRHDSGYEVHHQKLVFFAEDVGSNKGAIIGLMVGGVV`) is studied as part of

Amyloid-beta precursor protein from humans. Figure 35 shows the request made to the server and Figure 36 shows the HTTP response sent back to the client.

Figure 35 Screenshot of the request to TESSETOPE's linear B-cell epitope prediction tool, protein_sequence.txt (red box). The "POST" button is used to send the request to the server, run the tool, and obtain the predictions (green box).

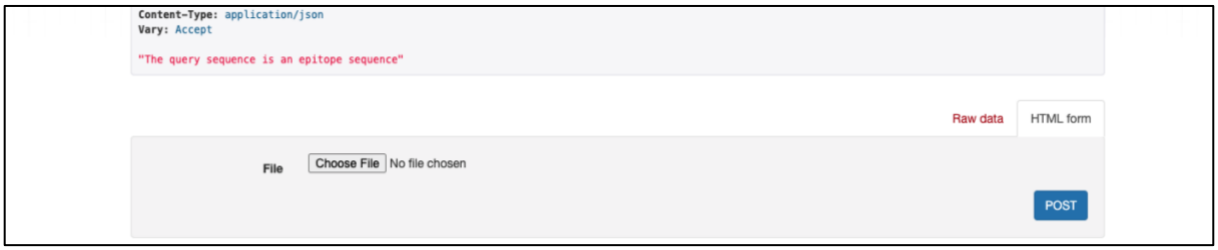


Figure 36 Screenshot of TESSETOPE's linear B-cell epitope prediction tool's HTTP response to the client; the prediction output. The query sequence is correctly classified as an "epitope sequence".

5.1.4 Conclusion and Future Directions

TESSETOPE was created with the goals of scientific researchers in mind. Ultimately, TESSETOPE provides easy to use, free, web accessible tools for epitope prediction. The tools can be leveraged to complement not only experimental studies but also additional computational studies for vaccine, biotherapeutic, and immunotherapy development. For example, in epitope-based vaccine design the candidate epitope search

space that forms the potential immune response targets is often enormous, therefore the epitope predictions made by TESSETOPE can narrow this search space, facilitating the completion of the development process.

Future versions of TESSETOPE will look to improve the web user interface, making it even easier to navigate and improving input and output displays, as well as, updating the machine learning models under the hood for better prediction performance.

CHAPTER SIX

6.1 Conclusion and Future Directions

The goal of this dissertation research was to develop models of B-cell and T-cell epitopes to assist in the classification of non-validated but potential epitope protein sequences. The models generated were rooted in machine learning algorithms and trained using protein sequence data, protein three-dimensional structure data, or a combination of both. Several experiments were conducted to determine appropriate feature vectors to represent conformational B-cell epitopes, linear B-cell epitopes, and linear T-cell epitopes and to choose and tune the best machine learning model based on stratified cross-validation performance. Models were then validated based on prediction performance on unseen test data. The best performing models were deployed to a server and used as the models under the hood of the TESSETOPE web API.

The tools of the TESSETOPE V1.0 web API can be leveraged to complement experimental and computational studies for vaccine, biotherapeutic, and immunotherapy development as the epitope predictions made by TESSETOPE can narrow the search space of candidate epitopes, expediting the development process.

Future directions, include larger training datasets to incorporate more data into the machine learning model training process, obtaining datasets where more than one epitope in a conformational epitope is identified to account for the reality of multiple epitopes of

the same antigen, and training other machine learning algorithms, such as the deep learning neural network. Future steps can be taken to continue to improve the models to account for updating epitope knowledge.

REFERENCES

[1] L. Backert and O. Kohlbacher, “Immunoinformatics and epitope prediction in the age of genomic medicine,” *Genome Med.*, vol. 7, no. 1, pp. 1–12, 2015.

[2] R. E. Soria-Guerra, R. Nieto-Gomez, D. O. Govea-Alonso, and S. Rosales-Mendoza, “An overview of bioinformatics tools for epitope prediction: Implications on vaccine development,” *J. Biomed. Inform.*, vol. 53, pp. 405–414, 2015.

[3] X. Yang, Xingdong; Yu, “An introduction to epitope prediction methods and software,” *Rev. Med. Virol.*, vol. 19, no. 1, pp. 77–96, 2009.

[4] A. C. Moser, S. Trenhaile, and K. Frankenberg, “Studies of antibody-antigen interactions by capillary electrophoresis: A review,” *Methods*, 2018.

[5] K. Krawczyk, X. Liu, T. Baker, J. Shi, and C. M. Deane, “Improving B-cell epitope prediction and its application to global antibody-antigen docking,” *Bioinformatics*, vol. 30, no. 16, pp. 2288–2294, 2014.

[6] D. Inbar, J. Hochman, and D. Givol, “Localization of Antibody-Combining Sites within the Variable Portions of Heavy and Light Chains,” *Proc. Nat. Acad. Sci.*, vol. 69, no. 9, pp. 2659–2662, 1972.

[7] V. Moreau *et al.*, “PEPOP: Computational design of immunogenic peptides,” *BMC Bioinformatics*, vol. 9, pp. 1–15, 2008.

[8] L. Potocnakova, M. Bhide, and L. B. Pulzova, “An Introduction to B-Cell Epitope Mapping and in Silico Epitope Prediction,” *J. Immunol. Res.*, vol. 2016, 2016.

[9] M. H. Van Regenmortel, “Specificity, polyspecificity, and heterospecificity of antibody-antigen recognition,” *J. Mol. Recognit.*, vol. 27, no. May, pp. 627–639, 2014.

[10] E. Jokinen, A. Huuhtanen, S. Mustjoki, M. Heinonen, and H. Lahdesmaki, “Predicting recognition between T cell receptors and epitopes with TCRGP,” *PLoS Comput. Biol.*, pp. 1–27, 2021.

[11] R. S. Salvat, A. S. Parker, Y. Choi, C. Bailey-Kellogg, and K. E. Griswold, “Mapping the Pareto Optimal Design Space for a Functionally Deimmunized Biotherapeutic Candidate,” *PLoS Comput. Biol.*, vol. 11, no. 1, 2015.

[12] S. K. Dhanda *et al.*, “Development of a strategy and computational application to select candidate protein analogues with reduced HLA binding and immunogenicity,” *Immunology*, vol. 153, no. 1, pp. 118–132, 2018.

[13] P. Sun *et al.*, “Bioinformatics resources and tools for conformational B-cell epitope prediction,” *Comput. Math. Methods Med.*, vol. 2013, 2013.

[14] H. W. Wang, Y. C. Lin, T. W. Pai, P. W. Tsai, and H. T. Chang, “A hybrid method of propensity scales and support vector machine in a linear epitope prediction,” *Proc. Int. Conf. Complex, Intell. Softw. Intensive Syst. CISIS 2011*, pp. 541–546, 2011.

[15] I. A. Doytchinova, P. Guan, and D. R. Flower, “EpiJen: A server for multistep T cell epitope prediction,” *BMC Bioinformatics*, vol. 7, pp. 1–11, 2006.

[16] K. C. Parker, M. A. Bednarek, and J. E. Coligan, “Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains.,” *J. Immunol.*, vol. 152, no. 1, pp. 163–75, 1994.

[17] H. Singh and G. P. S. Raghava, “ProPred1: Prediction of promiscuous MHC class-I binding sites,” *Bioinformatics*, vol. 19, no. 8, pp. 1009–1014, 2003.

[18] H. Singh and G. P. S. Raghava, “ProPred: Prediction of HLA-DR binding sites,” *Bioinformatics*, vol. 17, no. 12, pp. 1236–1237, 2001.

[19] P. Guan, I. A. Doytchinova, C. Zygouri, and D. R. Flower, “MHCPred: A server for quantitative prediction of peptide-MHC binding,” *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3621–3624, 2003.

[20] M. Nielsen *et al.*, “Reliable prediction of T-cell epitopes using neural networks with novel sequence representations,” *Protein Sci.*, vol. 12, no. 5, pp. 1007–1017, 2003.

[21] M. Andreatta and M. Nielsen, “Gapped sequence alignment using artificial neural networks: Application to the MHC class i system,” *Bioinformatics*, vol. 32, no. 4, pp. 511–517, 2015.

[22] P. A. Reche, J. P. Glutting, H. Zhang, and E. L. Reinherz, “Enhancement to the RANKPEP resource for the prediction of peptide binding to MHC molecules using profiles,” *Immunogenetics*, vol. 56, no. 6, pp. 405–419, 2004.

[23] P. Dönnes and O. Kohlbacher, “SVMHC: A server for prediction of MHC-binding peptides,” *Nucleic Acids Res.*, vol. 34, no. WEB. SERV. ISS., pp. 194–197, 2006.

[24] M. V. Larsen, C. Lundegaard, K. Lamberth, S. Buus, O. Lund, and M. Nielsen,

"Large-scale validation of methods for cytotoxic T-lymphocyte epitope prediction," *BMC Bioinformatics*, vol. 8, pp. 1–12, 2007.

[25] M. Bhasin and G. Raghava, "A hybrid approach for predicting promiscuous MHC class I restricted T cell epitopes," *J. Biosci.*, vol. 32, no. 1, pp. 31–42, 2007.

[26] S. Saha and G. P. S. Raghava, "Prediction of Continuous B-Cell Epitopes in an Antigen Using Recurrent Neural Network," *Proteins*, vol. 65, pp. 40–48, 2006.

[27] Y. El-Manzalawy, D. Dobbs, and V. Honavar, "Predicting linear B-cell epitopes using string kernels," *J. Mol. Recognit.*, vol. 21, no. 4, pp. 243–255, 2008.

[28] Y. El-Manzalawy, D. Dobbs, and V. Honavar, "Predicting flexible length linear B-cell epitopes.," *Comput. Syst. Bioinformatics Conf.*, vol. 7, pp. 121–32, 2008.

[29] M. C. Jespersen, B. Peters, M. Nielsen, and P. Marcatili, "BepiPred-2.0: Improving sequence-based B-cell epitope prediction using conformational epitopes," *Nucleic Acids Res.*, vol. 45, no. W1, pp. W24–W29, 2017.

[30] M. J. Sweredoski and P. Baldi, "COBEpro: A novel system for predicting continuous B-cell epitopes," *Protein Eng. Des. Sel.*, vol. 22, no. 3, pp. 113–120, 2009.

[31] H. Singh, H. R. Ansari, and G. P. S. Raghava, "Improved Method for Linear B-Cell Epitope Prediction Using Antigen's Primary Sequence," *PLoS One*, vol. 8, no. 5, pp. 1–8, 2013.

[32] S. Saha and G. P. S. Raghava, "BcePred: Prediction of continuous B-cell epitopes in antigenic sequences using physico-chemical properties," *Artif. Immune Syst. Third Int. Conf. ICARIS 2004, Catania Sicily, Italy, Sept. 13-16, 2004 Proc.*, pp.

197–204, 2004.

[33] B. Yao, L. Zhang, S. Liang, and C. Zhang, “SVMTriP: A Method to Predict Antigenic Epitopes Using Support Vector Machine to Integrate Tri-Peptide Similarity and Propensity,” *PLoS One*, vol. 7, no. 9, 2012.

[34] J. V. Kringelum, C. Lundegaard, O. Lund, and M. Nielsen, “Reliable B Cell Epitope Predictions: Impacts of Method Development and Improved Benchmarking,” *PLoS Comput. Biol.*, vol. 8, no. 12, 2012.

[35] M. J. Sweredoski and P. Baldi, “PEPITO: Improved discontinuous B-cell epitope prediction using multiple distance thresholds and half sphere exposure,” *Bioinformatics*, vol. 24, no. 12, pp. 1459–1460, 2008.

[36] J. Ponomarenko *et al.*, “ElliPro: A new structure-based tool for the prediction of antibody epitopes,” *BMC Bioinformatics*, vol. 9, 2008.

[37] J. Sun *et al.*, “SEPPA: A computational server for spatial epitope prediction of protein antigens,” *Nucleic Acids Res.*, vol. 37, no. SUPPL. 2, pp. 612–616, 2009.

[38] N. D. Rubinstein, I. Mayrose, and T. Pupko, “A machine-learning approach for predicting B-cell epitopes,” *Mol. Immunol.*, vol. 46, no. 5, pp. 840–847, 2009.

[39] H. R. Ansari and G. P. Raghava, “Identification of conformational B-cell Epitopes in an antigen from its primary sequence,” *Immunome Res.*, vol. 6, no. 1, 2010.

[40] S. Liang, S. Liu, C. Zhang, and Y. Zhou, “A simple reference state makes a significant improvement in near-native selections from structurally refined docking decoys,” *Proteins*, vol. 69, pp. 244–253, 2007.

[41] S. Liang, D. Zheng, D. M. Standley, B. Yao, M. Zacharias, and C. Zhang,

"EPSVR and EPMeta : prediction of antigenic epitopes using support vector regression and multiple server results," *BMC Bioinformatics*, vol. 11, p. 381, 2010.

[42] I. Sela-Culang *et al.*, "Using a combined computational-experimental approach to predict antibody-specific B cell epitopes," *Structure*, vol. 22, no. 4, pp. 646–657, 2014.

[43] S. S. Negi and W. Braun, "Automated detection of conformational epitopes using phage display peptide sequences," *Bioinform. Biol. Insights*, no. 3, pp. 71–81, 2009.

[44] J. Huang, A. Gutteridge, W. Honda, and M. Kanehisa, "MIMOX: A web tool for phage display based epitope mapping," *BMC Bioinformatics*, vol. 7, 2006.

[45] I. Mayrose *et al.*, "Epitope mapping using combinatorial phage-display libraries: A graph-based algorithm," *Nucleic Acids Res.*, vol. 35, no. 1, pp. 69–78, 2007.

[46] U. Kulkarni-Kale, S. Bhosle, and A. S. Kolaskar, "CEP: A conformational epitope prediction server," *Nucleic Acids Res.*, vol. 33, no. SUPPL. 2, pp. 168–171, 2005.

[47] R. Rapberger, A. Lukas, and B. Mayer, "Identification of discontinuous antigenic determinants on proteins based on shape complementarities," *J. Mol. Recognit.*, vol. 20, pp. 113–121, 2007.

[48] S. Soga, D. Kuroda, H. Shirai, M. Kobori, and N. Hirayama, "Use of amino acid composition to predict epitope residues of individual antibodies," *Protein Eng. Des. Sel.*, vol. 23, no. 6, pp. 441–448, 2010.

[49] W. Zhang, Y. Xiong, M. Zhao, H. Zou, X. Ye, and J. Liu, "Prediction of conformational B-cell epitopes from 3D structures by random forests with a

distance-based feature," *BMC Bioinformatics*, vol. 12, 2011.

[50] Q. M. Sheikh, D. Gatherer, P. A. Reche, and D. R. Flower, "Towards the knowledge-based design of universal influenza epitope ensemble vaccines," *Bioinformatics*, vol. 32, no. 21, pp. 3233–3239, 2016.

[51] S. E. C. Caoili, "Benchmarking B-cell epitope prediction for the design of peptide-based vaccines: Problems and prospects," *J. Biomed. Biotechnol.*, vol. 2010, 2010.

[52] M. Topuzoğullari *et al.*, "An insight into the epitope-based peptide vaccine design strategy and studies against COVID-19," *Turkish J. Biol.*, vol. 44, pp. 215–227, 2020.

[53] A. Naz, F. Shahid, T. T. Butt, F. M. Awan, and A. Ali, "Designing Multi-Epitope Vaccines to Combat Emerging Coronavirus Disease 2019 (COVID-19) by Employing Immuno-Informatics Approach," *Front. Immunol.*, vol. 11, no. July, pp. 1–13, 2020.

[54] B. Schubert, C. Schärfe, P. Dönnes, T. Hopf, D. Marks, and O. Kohlbacher, "Population-specific design of de-immunized protein biotherapeutics," *arXiv*, vol. 1706.09083, pp. 1–19, 2018.

[55] M. Gubin *et al.*, "Checkpoint Blockade Cancer Immunotherapy Targets Tumour-Specific Mutant Antigens," *Nature*, vol. 515, no. 7528, pp. 577–581, 2015.

[56] N. Segal *et al.*, "Epitope Landscape in Breast and Colorectal Cancer," *Cancer Res.*, vol. 68, no. 3, pp. 889–892, 2008.

[57] T. Li, K. Fan, J. Wang, and W. Wang, "Reduction of protein sequence complexity by residue grouping," *Protein Eng. Des. Sel.*, vol. 16, no. 5, pp. 323–330, 2003.

[58] J. Bacardit, M. Stout, J. D. Hirst, A. Valencia, R. E. Smith, and N. Krasnogor, "Automated alphabet reduction for protein datasets," *BMC Bioinformatics*, vol. 10, pp. 1–16, 2009.

[59] M. Masso, "Prediction of human immunodeficiency virus type 1 drug resistance: Representation of target sequence mutational patterns via an n-grams approach," *Proc. - 2012 IEEE Int. Conf. Bioinforma. Biomed. BIBM 2012*, pp. 173–178, 2012.

[60] S. Saha, M. Bhasin, and G. P. S. Raghava, "Bcipep: A database of B-cell epitopes," *BMC Genomics*, vol. 6, pp. 1–7, 2005.

[61] H. M. Berman *et al.*, "The Protein Data Bank," *Nucleic Acids Res.*, vol. 28, no. 1, pp. 235–242, 2000.

[62] J. Wang and W. Wang, "A computational approach to simplifying the protein folding alphabet," *Nat. Struct. Biol.*, vol. 6, no. 11, pp. 1033–1038, 1999.

[63] C. Branden and J. Tooze, "Introduction to protein structure," 2nd ed., New York, 1991, p. 302.

[64] I. I. Vaisman, A. Tropsha, and W. Zheng, "Compositional preferences in quadruplets of nearest neighbor residues in protein structures: Statistical geometry analysis," *Proc. - IEEE Int. Jt. Symp. Intell. Syst.*, pp. 163–168, 1998.

[65] L. R. Murphy, A. Wallqvist, and R. M. Levy, "Simplified amino acid alphabets for protein fold recognition and implications for folding," *Protein Eng. Des. Sel.*, vol. 13, no. 3, pp. 149–152, 2000.

[66] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[67] M. Claesen and B. De Moor, “Hyperparameter Search in Machine Learning,” *XI Metaheuristics Int. Conf.*, 2015.

[68] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization,” *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[69] M. J. Blythe, I. A. Doytchinova, and D. R. Flower, “JenPep: a database of quantitative functional peptide data for immunology,” *Bioinformatics*, vol. 18, no. 3, pp. 434–439, 2002.

[70] H. McSparron, M. J. Blythe, C. Zygouri, I. A. Doytchinova, and D. R. Flower, “JenPep: a novel computational information resource for immunobiology and vaccinology,” *J Chem Inf Comput Sci.*, vol. 43, no. 4, pp. 1276–87, 2003.

[71] H.-P. Adams and J. A. Koziol, “Prediction of binding to MHC class I molecules,” *J. Immunol. Methods*, vol. 185, no. 2, pp. 181–190, 1995.

[72] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[73] K. Fawagreh, M. M. Gaber, and E. Elyan, “Random forests: from early developments to recent advancements,” *Syst. Sci. Control Eng.*, vol. 2, no. 1, pp. 602–609, 2014.

[74] L. Breiman, “Bagging Predictors,” *Mach. Learn.*, vol. 24, pp. 123–140, 1996.

[75] T. K. Ho, “Random Decision Forests,” *Proc. 3rd Int. Conf. Doc. Anal. Recognit.*, vol. 1, pp. 278–282, 1995.

[76] T. K. Ho, “The Random Subspace Method for Constructing Decision Forests,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, 1998.

[77] Y. Amit and D. Geman, “Shape Quantization and Recognition with Randomized

Trees," *Neural Comput.*, vol. 9, no. 7, pp. 1545–1588, 1997.

[78] J. Huang, W. Honda, and M. Kanehisa, "Predicting B cell epitope residues with network topology based amino acid indices.," *Genome Inform.*, vol. 19, pp. 40–49, 2007.

[79] R. Bhaskaran and P. K. Ponnuswamy, "Positional flexibilities of amino acid residues in globular proteins," *Chem. Biol. Drug Des.*, vol. 32, no. 4, pp. 241–255, 1988.

[80] R. Grantham, "Amino Acid Difference Formula to Help Explain Protein Evolution," *Science (80-.).*, vol. 185, no. 4154, pp. 862–864, 1974.

[81] T. P. Hopp and K. R. Woods, "Prediction of protein antigenic determinants from amino acid sequences," *Proc. Natl. Acad. Sci. USA*, vol. 78, no. 6, pp. 3824–3828, 1981.

[82] G. W. Welling, W. J. Weijer, R. van der Zee, and S. Welling-Wester, "Prediction of sequential antigenic regions in proteins," *FEBS Lett.*, vol. 188, no. 2, pp. 215–218, 1985.

[83] M. Z. Tien, A. G. Meyer, D. K. Sydykova, S. J. Spielman, and C. O. Wilke, "Maximum allowed solvent accessibilites of residues in proteins," *PLoS One*, vol. 8, no. 11, 2013.

[84] P. Y. Chou and G. D. Fasman, "Prediction of the secondary structure of proteins from their amino acid sequence," *Adv. Enzym. Relat. Areas Mol. Biol.*, vol. 47, pp. 45–148, 1978.

[85] M. Masso, "Sequence-based prediction of HIV-1 coreceptor usage: utility of n-

grams for representing gp120 V3 loops," *ACM-BCB,* pp. 309–314, 2011.

[86] R. K. Singh, A. Tropsha, and I. I. Vaisman, "Delaunay Tessellation of Proteins : Four Body Nearest Neighbor Propensities of Amino Acid Residues," *J. Comput. Biol.*, vol. 3, no. 2, pp. 213–222, 1996.

[87] D. P. Yee, H. S. Chan, T. F. Havel, and K. A. Dill, "Does Compactness Induce Secondary Structure in Proteins?: A Study of Poly-alanine Chains Computed by Distance Geometry," *J. Mol. Biol.*, vol. 241, no. 4, pp. 557–573, 1994.

[88] M. J. Behe, E. E. Lattman, and G. D. Rose, "The protein-folding problem: The native fold determines packing, but does packing determine the native fold?," *Proc. Natl. Acad. Sci. USA*, vol. 88, pp. 4195–4199, 1991.

[89] T. J. Taylor and I. I. Vaisman, "Graph theoretic properties of networks formed by the Delaunay tessellation of protein structures," *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 73, no. 4, pp. 1–13, 2006.

[90] P. Haste Andersen, M. Nielsen, and O. Lund, "Prediction of residues in discontinuous B-cell epitopes using protein 3D structures," *Protein Sci.*, vol. 15, no. 11, pp. 2558–2567, 2006.

[91] C. B. Barber, D. P. Dobkin, and H. Huhdanpaa, "The Quickhull Algorithm for Convex Hulls," *ACM Trans. Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996.

[92] J. E. Sadler, "Biochemistry and genetics of von willebrand factor," *Annu. Rev. Biochem.*, vol. 67, pp. 395–424, 1998.

[93] R. Celikel, K. I. Varughese, Madhusudan, A. Yoshioka, J. Ware, and Z. M. Ruggeri, "Crystal structure of the von Willebrand factor A1 domain in complex

with the function blocking NMC-4 Fab," *Nat. Struct. Biol.*, vol. 5, no. 3, pp. 189–194, 1998.

[94] R. Celikel *et al.*, "Crystal Structure of NMC-4 Fab anti-von Willebrand factor A1 domain," *Blood Cells,Molecules, Dis.*, vol. 23, no. 1, pp. 123–134, 1997.

[95] D. Sehnal *et al.*, "Mol * Viewer : modern web app for 3D visualization and analysis of large biomolecular structures," *Nucleic Acids Res.*, vol. 49, no. May, pp. 431–437, 2021.

[96] J. Novotny *et al.*, "Antigenic determinants in proteins coincide with surface regions accessible to large probes (antibody domains)," *Proc. Natl. Acad. Sci. USA*, vol. 83, no. January, pp. 226–230, 1986.

[97] J. M. Thornton, M. S. Edwards, W. R. Taylor, and D. J. Barlow, "Location of 'continuous' antigenic determinants in the protruding regions of proteins," *EMBO J.*, vol. 5, no. 2, pp. 409–413, 1986.

[98] C. Chothia, "The nature of the accessible and buried surfaces in proteins," *J. Mol. Biol.*, vol. 105, no. 1, pp. 1–12, 1976.

[99] H. Chen and H. X. Zhou, "Prediction of solvent accessibility and sites of deleterious mutations from protein sequence," *Nucleic Acids Res.*, vol. 33, no. 10, pp. 3193–3199, 2005.

[100] W. Wu, Z. Wang, P. Cong, and T. Li, "Accurate prediction of protein relative solvent accessibility using a balanced model," *BioData Min.*, vol. 10, no. 1, pp. 1–14, 2017.

[101] H. Zhang, T. Zhang, K. Chen, S. Shen, J. Ruan, and L. Kurgan, "On the relation

between residue flexibility and local solvent accessibility in proteins," *Proteins Struct. Funct. Bioinforma.*, vol. 76, no. 3, pp. 617–636, 2009.

[102] S. Miller, J. Janin, A. Lesk, and C. Chothia, "Interior and surface of monomeric proteins," *J. Mol. Biol.*, vol. 196, no. 3, pp. 641–656, 1987.

BIOGRAPHY

Kiran K. Sewsankar graduated from Leicester High School, Leicester, Massachusetts, in 2011. He received his Bachelor of Science from Worcester State University in 2015.