Aston University

MSc Artificial Intelligence

Evolutionary Classification Systems for Epitope Prediction

Navroop Singh
190114871

supervised by
Felipe Campelo

September 29, 2023

Abstract

Identification of linear B-cell epitopes remains an important task for diagnostic tests and developing vaccines, by providing targets for experimental investigation. The field of epitope prediction has been rapidly evolving since its inception, from unreliable single propensity scales to state-of-the-art models to machine learning models being developed. Genetic programming has not been used for epitope prediction, this dissertation aims to provide insight into the efficacy of using genetic programming in the epitope prediction problem.

This dissertation presents a novel approach to linear B-cell epitope prediction using a genetic programming classification model, via gplearn. Through a robust data-mining pipeline, a Lentivirus dataset was cleaned and prepared for the modelling of the genetic programming classifier. The model was tested against the current state-of-the-art epitope prediction tool, BepiPred 3.0 which is widely used as a benchmark in this field. The genetic programming model demonstrated superior performance compared to BepiPred 3.0, showcasing the viability of using evolutionary methods for epitope prediction.

This dissertation provides valuable insight into the application of genetic programming in the epitope prediction problem and showcases the feasibility of using genetic programming in this context.

Acknowledgements

I would like to express my sincere appreciation to my supervisor Felipe Campelo, for his continuous guidance and support throughout the duration of the project.

Contents

1 Introduction

The immune system is a complex network that can be divided into two categories, innate and adaptive. "The innate immune system is the body's first line of defense against germs entering the body". (*The innate and adaptive immune systems* 2020) The innate immune system encompasses natural barriers such as the cough reflex, the skin barrier, and stomach acids. However, the innate immune system has only limited power to control the spread of antigens as it does not have the ability to differentiate between types of pathogens. In contrast, the adaptive immune system has the power to recognise and destroy invading pathogens individually. Furthermore, the adaptive immune system is able to remember pathogens, through their pathogen-specific long-lasting protective memory, and is able to produce stronger attacks each time the specific pathogen is re-encountered. (Paul 2012)

The adaptive immune system is comprised of T lymphocytes (T-cells), B lymphocytes (B-cells), and antibodies. When the body senses foreign substances (antigens), B-cells that match the attacking antigens are triggered to make antibodies specific to the antigen. The antibodies then lock onto the antigens and the T-cells are responsible for destroying the antigen. B-cells are activated when their receptors (BCR) recognise an epitope; a small sub-region on the antigen as shown in Figure 1. Identifying B-cell epitopes is an area of great interest in the medical field as it helps the development of vaccines and treatments for various diseases.

B-cell epitopes are generally classified as either linear (continuous) or conformational (discontinuous) depending on the structural arrangement of the epitope within the antigen. Linear B-cell epitopes consist of sequential amino acid sequences, whereas, in conformational B-cell epitopes amino acids which constitute the specific antigen are distant, therefore, protein folding is used to bring the amino acids together. Approximately 90% of B-cell epitopes are classified as conformational or discontinuous, however, the majority of work surrounding epitope prediction and classification is conducted on linear B-cell epitopes. This is due to requiring the 3-D structure of the protein to be able to predict the conformational B-cell, which is not commonly available, whereas, linear B-cell epitopes can be predicted from the amino acid sequences which are widely available. Furthermore, such conformational epitopes are more difficult to synthesize for vaccine production compared to linear B-cell epitopes. (Sanchez-Trincado et al. 2017)

In recent years, machine learning solutions for linear B-cell epitope identification have been employed more frequently compared to traditional techniques such as X-ray crystallography. There are multiple reasons for this: due to the laborious nature of conducting the identification, the process is very time-consuming, costly, and technically difficult compared to machine learning solutions. (Morris 1996) Furthermore, in recent years large public datasets, which are both labelled and diverse, on protein data sequences have become available fronting multiple machine-learning solutions for B-cell epitope classification and prediction. Examples of such machine-learning approaches include: Support Vector Machines methods implemented in BCPRED (EL-Manzalawy et al. 2008), LBEEP (Saravanan & Gautham 2015), and LBtope (Singh et al. 2013); neural network methods implemented in ABCpred (Saha & Raghava 2006), and DRREP (Sher et al. 2017); machine learning algorithms such as random forest used by Bepipred-2.0 (Jespersen et al. 2017, Galanis et al. 2021). In contrast to the extensive use of these machine-learning techniques, genetic programming has yet to be applied to the linear B-cell epitope problem. In many cases, a genetic programming-based classifier has outperformed other machine learning techniques in specific scenarios (Espejo et al. 2009), therefore exploring the effectiveness of genetic programming in the context of linear B-cell epitopes is essential.

In basics, genetic programming is a type of evolutionary algorithm, a subset of machine learning, which is inspired by Darwinian evolution and "breeds" programs. In genetic programming, an initial population of large computer programs is generated at random, each program is executed and the results are used to assign a fitness value to each program. A new population, the next generation, is created by copying selected programs with the best fitness scores, this population undergoes genetic processes such as mutation and crossover to generate "offsprings" based on the fitness score. This generation is once again evaluated and fitness is assigned to each program. This process is continued until the correct program of the termination criteria is fulfilled. (Bäck et al. 2018)

This work aims to provide valuable insight into the effectiveness of using a genetic programming classifier model for linear B-cell epitope classification. As mentioned before genetic programming has yet to be used in this context, therefore, it is anticipated that the use of genetic programming may potentially produce a more accurate classification model for linear B-cell epitopes.

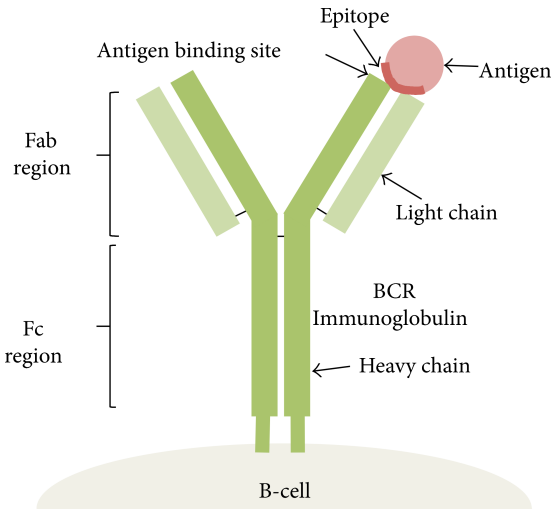


Figure 1: B-cell epitope recognition (Sanchez-Trincado et al. 2017)

1.1 Structure of Dissertation

In Section 2 related works are discussed, both in the field of epitope prediction and genetic programming. In Section 3 the project description and the requirements are detailed and analysed. In Section 4 a description of how the problem was tackled and the thought process behind certain decisions made are detailed. Section 5 describes how the project was managed. Sections 6 and 7 are the evaluation and conclusions of the project respectively. Finally, in the Appendix, materials related to this project are present, such as the final code and personal tutor meeting notes.

2 Literature Review

This literature review focuses on two themes, which are pertinent to our research. The first theme is the understanding of linear B-cell epitope prediction methods, literature around this will mainly entail the development of these methods over time, where the future development is headed, the applications of these methods and the current state-of-the-art. The second theme is exploring genetic programming and how to make an effective model around a large dataset. Ultimately, the review of the literature around both themes helped identify knowledge gaps.

2.1 Epitope Prediction

2.1.1 Epitope Prediction Methods

Epitope prediction using computational methods has seen continuous development since the 1980s. Early efforts by Hopp and Woods (Yang & Yu 2009) used Levitt's hydrophilicity scale, which assigns hydrophobicity values to each amino acid, and a local averaging of these values along the peptide chain to predict epitopes. This correlation between specific physicochemical properties of amino acids and the position of linear B-cell epitopes within protein sequences for epitope prediction has been reported in many studies (Karplus & Schulz 1985, Parker et al. 1986, Pellequer & Westhof 1993). Since these studies, many epitope prediction models used propensity scales, numerical values assigned to molecules in order to make predictions, for epitope prediction. The first wave of linear B-cell epitope prediction focused on using such propensity scales, antigenicity (Kolaskar & Tongaonkar

1990), which recorded a 75% accuracy, and surface accessibility (Emini et al. 1985). The next wave of such models involved using multiple physicochemical properties, methods included PREDITOP (Pellequer & Westhof 1993), BEPITOPE (Odorico & Pellequer 2003) and BcePred (Saha & Raghava 2004). The most notable of these models is BcePred (Saha & Raghava 2004), which used several physicochemical properties such as hydrophilicity, flexibility and polarity to predict B-cell epitopes. BcePred used a balanced dataset consisting of "1029 non-redundant B cell epitopes" with an equal number of non-epitopes and managed to achieve an accuracy varying between "52.92% and 57.53%".

Blythe and Flower (Blythe & Flower 2005) performed an extensive evaluation of propensity scales and their effectiveness in linear B-cell epitope prediction methods; in which they "assessed 484 amino acid propensity scales" to examine the correlation between propensity scales and the location of linear B-cell epitopes in a set of 50 proteins. In this study, they reported "that even the best set of scales and parameters performed only marginally better than random". They concluded that "amino acid propensity profiles cannot be used effectively to predict linear epitopes" and they also correctly suggested that more advanced approaches using artificial intelligence technology must be developed to advance the field.

With the poor quality of purely propensity scale-based methods and the rise of available biological data, machine learning techniques started to be introduced in B-cell epitope prediction. Motivated by the findings of Blythe and Flower (Blythe & Flower 2005), several authors explored methods to improve the predictive performance of propensity scales to predict linear B-cell epitopes through the introduction of machine learning methods. One such case was BepiPred (Larsen et al. 2006), which tested the propensity scales of "antigenicity, hydrophilicity, inverted hydrophobicity, accessibility and secondary structure" to find the best scale to combine with a Hidden Markov Model (HMM). Firstly, they tested each propensity scale on "Pellequer data set", which contained "14 protein sequences and 83 epitopes" with an epitope density of 0.34. Through their experiment, they concluded that the hydrophilicity scale by (Parker et al. 1986) performed the best, they then combined this hydrophilicity scale with an HMM to produce BeciPred. The BeciPred model was tested on the HIV dataset which contained "10 protein sequences and the epitope density" of 3.8. BeciPred achieved an Area Under ROC Curve (AUC) score of 0.6, which slightly improved the older methods involving singular propensity scales. Another method using improved propensity scale methods was Chen et al. (Chen et al. 2007), which developed an amino acid pair (AAP) antigenicity scale from their findings that certain APPs tend to occur more frequently in B-cell epitopes than in non-epitope peptides. With this information, Chen et al.(Chen et al. 2007) trained an SVM machine using the APP scale and tested it against a dataset of 872 unique epitopes and 872 non-epitopes, their results concluded using "AAP antigenicity scale approach has much better performance than the existing scales".

Other authors opted to use only machine learning methods for linear B-cell epitope prediction, instead of including propensity scales. One example is ABCPred (Saha & Raghava 2006), which uses an artificial neural network for predicting linear B-cell epitopes. In their study Saha et al. (Saha & Raghava 2006) evaluated both feed-forward and recurrent neural networks on a data set of "700 non-redundant B-cell epitopes" and an equal number of epitope peptides. The results concluded that the best-performing model was a recurrent neural network with a single hidden layer of 35 hidden units for a window length of 16. Their final network "yielded an overall prediction accuracy of 65.93% when tested by five-fold cross-validation". Another example includes LBtope (Singh et al. 2013), which utilized Support Vector Machines (SVM) to predict linear B-cell epitopes. In their study, the researchers retrieved validated epitopes and non-epitopes from the Immune Epitope Database (IED) and created several datasets from it. Their SVM was then trained on these datasets resulting in accuracies ranging from 54% to 86%.

In recent years, there has been great interest in learning protein representation using language modelling; which revolves around training large models in a self-supervised way on a large corpus. This approach has gained a lot of popularity because of its ability to extract meaningful features from protein sequences and structures. By leveraging pre-trained models such as BERT (Devlin et al. 2018) and ESM (Rives et al. 2021), researchers are able to encode proteins into dense vector representation to capture the most important information. Following the success of this application, many

researchers have begun to implement embeddings from protein language models to train classifiers for epitope prediction including BepiPred 3.0 (Clifford et al. 2022), EpitopeVec (Bahai et al. 2021) and EpiDope (Collatz et al. 2021). Such implementations are considered state-of-the-art models for linear B-cell epitope prediction, which are able to predict large amounts of epitope at once, however, require a larger demand of computational resources and time. Amongst these models, BepiPred 3.0 (Clifford et al. 2022) is the most used model for epitope prediction, it is able to predict both linear and discontinuous B-cell epitopes. BepiPred 3.0 used the BP3 dataset to train and test their model, BP3 was built from the Protein Data Bank and had 1,466 antigens, redundancy reduction was used to reduce the antigens to 603. Further redundancy reduction was done to create the dataset BP3HR. Antigen sequences were passed into a pre-trained ESM-2 transformer, and the resulting sequence representation was extracted from the model. "Feed Forward (FFNN), Convolutional (CNN), and Long Short-term Memory (LSTM) neural networks were trained" were then trained on these datasets using five-fold cross-validation and hyperparameter tuning was performed. Each of these models were then evaluated on an external dataset made from IEDB using performance parameters "AUC, AUC10, MCC, recall, precision, F1-score, and accuracy". Results concluded that their FFNN model gave the best results, feature engineering was then used to retrain their FFNN generating even better results of 0.774 AUC. They also made BepiPred 3.0 is also available as a web server available for anyone to use.

In addition to language models being introduced, some researchers have begun to experiment on datasets to improve linear B-cell epitope prediction performances. One such example is introduced in the paper (Ashford et al. 2021). The authors of this paper propose to train epitope prediction models on organism or taxon-specific datasets to introduce organism or taxon-specific models. Therefore, the authors introduced RF-OrgSpec, which is a random forest model trained on organism-specific data, and tested it against BepiPred 2.0, SVMTriP, LBtope, ABCpred and iBCE-EL. They concluded that even simple models trained on organism-specific data outperformed current state-of-the-art predictors, suggesting that organism-specific data can be important when developing predictive models for vaccine and drug discovery.

2.1.2 Applications

The identification of linear B-cell epitopes is very important in immunology. With the knowledge of an exact epitope sequence, peptides can be synthesized which mimic epitopes in diseases. This can be used to diagnose human diseases and "epitopes have received more and more attention in pathogen diagnosis in recent years" (Zhou et al. 2022). Epitope prediction has been used in the diagnosis of many diseases including fungal infections, parasitic infections, viral infections and bacterial infections (Zhou et al. 2022). Another important application of linear B-cell epitopes is epitope-based vaccines. A vaccine is a type of medicine that helps the immune system recognize and defend against harmful pathogens. A vaccine contains an inactive, dead or weakened form of a pathogen or molecule that mimics a disease-causing pathogen, which when administered to a person, the immune system to be triggered to produce corresponding antibodies to destroy the pathogen. The immune response creates memory cells, which are able to remember the pathogen so that the person, is able to be protected against the actual pathogen in the future, this protection can last a whole lifetime. Vaccines are able to provide immunity against specific diseases, thus reducing the severity and spread of the disease to the whole populace. Hence, vaccine design is of utmost importance in the medical field, and vaccines against fatal diseases are constantly being researched and tested for public protection.

Epitope prediction is able to determine the epitope of a specific antigen, which can be used to produce epitope-based vaccines. This is done via protein recombination technology, which is able to recreate an epitope, which when administered as a vaccine is able to stimulate an immune response in a person. Since these epitope-based vaccines only have the specific epitopes rather than the whole pathogen, the vaccines can be more targeted and specific to the desired immune response. This precision allows for a reduced risk of unwanted side effects commonly associated with traditional vaccines. Further advantages of the epitope-based approach to vaccines include the "opportunity

to rationally engineer epitopes for increased potency and breadth, and the ability to focus immune responses on conserved epitopes." (Sette & Fikes 2003).

2.2 Genetic Programming

Through the research and literature review conducted, there was no literature on implementing genetic programming in the epitope prediction field. Therefore, this literature review will focus on genetic programming for classification and implementations of genetic programming in similar applications in order to gain a deeper understanding of its potential utility in epitope prediction.

Genetic programming is traditionally represented as a tree structure, which consists of internal nodes representing predefined functions and leaf nodes representing predefined terminals. The subtrees originating from the function nodes serve as arguments for these functions (Nag & Pal 2019). Genetic programming has been used widely within the medical field but is not as popular as other machine learning methods. The authors in (Kumar et al. 2022) used genetic programming to detect chronic kidney disease (CDK) using a clinical data set of 400 patients, with 24 attributes and two classed CKD and non-CDK. In this study, a standard tree-based genetic programming model was used and performance metrics of "accuracy, sensitivity, specificity, F1-score, and the area under the ROC curve (AUC)" were used. The final performance of the model resulted in an accuracy of 99.33% and an AUC of 0.99, the author successfully showcased the utility of genetic programming in classification problems. Genetic programming presents an advantage compared to other classification techniques, as genetic programming does not need a prior definition of the structure. Genetic programming is able to automatically evolve until finding the best fit (Santoso et al. 2021).

There are many packages available for genetic programming in Python, including DEAP, gplearn and PyGAD, which can be used with Jupyter Notebook. For this project, gplearn was chosen as the library as it is made as an extension of scikit-learn, which will be used extensively throughout the project. This simplifies creating the model as it has an in-built genetic programming classifier, called Symbolic Classifier which can be used in binary classification, which is the aim of this project. The genetic programming classifier presented by gplearn is a tree-based model. The symbolic regression technique used aims to identify the underlying mathematical expressions which best describe a relationship (gplearn 2023).

Appendix D, showcases performance metrics commonly used for measuring model performances and it will be used in this project for model performance and comparison.

2.3 Summary

As highlighted through the extensive literature review, epitope prediction is an important and growing field of research. Section 2.1.1 highlights the evolution of linear B-cell epitope prediction from its infancy to the current state-of-the-art, which showcases important information on how to approach epitope prediction. The first is the importance of a large, labelled dataset to allow the models enough data to produce accurate results. Furthermore, the dataset must be balanced, meaning it should have equal numbers of epitope and non-epitope data in order for the training classifier to be unbiased. The literature demonstrated the importance of a well-defined and good data mining pipeline. In addition, many different performance metrics were used to evaluate models, however, the most utilized ones were accuracy and AUC scores.

Section 2.1.2 showcased the importance of researching and progressing the epitope-prediction field. Epitope prediction methods not involving machine-learning techniques such as X-ray crystallography, which are very reliable, are too expensive, time-consuming, and very difficult to apply for many targets (Sela-Culang et al. 2014). An accurate and reliable epitope prediction also benefits the general populace with faster and quicker production of vaccines which can save many lives in pandemics. Furthermore, due to the applications and advantages relied on in section 2.1.2 identification of B-cell epitopes is one of the major focuses of research for immunologists.

Section 2.2 provided insight into genetic programming and how it may be used to tackle epitope

prediction. An example of a genetic programming classifier applied to a very similar application to the one tackled throughout this project, was showcased. The example used a generic tree-based approach to genetic programming, which will also be used in this project. Furthermore, the genetic programming libraries were showcased and gplearn was chosen.

3 Problem Description and Problem Statement

The project is a linear B-cell epitope prediction model which utilises a genetic programming classifier. By utilizing genetic programming, which has never been used in this field but showed good results in a similar application of binary classification, the project aims to gain insight into the feasibility of a genetic programming approach for epitope prediction.

There are many potential benefits of this project, these benefits can be categorised as academic, technical, business and social aspects:

- Academic: This project has academic benefits as it investigates a novel approach to linear B-cell epitope prediction. Through the comparison of the model proposed and the current state-of-the-art model, this project will contribute to the existing literature.
- Technical: This project will identify and evaluate a new way of epitope prediction by investigating the use of a genetic programming classifier.
- Business: This project has business benefits as the genetic programming approaches may outperform the current state-of-the-art epitope prediction models. If the genetic programming model has good accuracy with lower cost or faster prediction times, this could enable researchers to investigate more samples and be faster in the production of new vaccines.
- Social: If the outcome of the project investigation, evolutionary computing approaches do outperform current state-of-the-art methods, the project could have significant social benefits by improving the ability to predict and prevent infectious disease, through vaccines.

Overall, this system not only addresses the immediate need for improved epitope prediction but also has the potential to contribute significantly to advancements in immunology, biomedical research, and healthcare, ultimately making a positive impact on both scientific understanding and economic considerations in these domains. The project aims to address the critical issue of predicting linear B-cell epitopes using a genetic programming classifier.

To allow for a fair evaluation of this approach against the current state-of-the-art solutions, a good and robust data mining pipeline must be followed. This will allow for the best results in accuracy for the genetic programming model, and this will allow for a fair comparison and evaluation to take place against the current state-of-the-art. The model which will be evaluated against the genetic programming classifier will be BepiPred 3.0, this conclusion was led by the literature review, which established BepiPred 3.0 as a state-of-the-art model. Furthermore, BepiPred 3.0 is available as a web server which will facilitate getting results for comparison.

As previously mentioned a robust data-mining pipeline must be used, this conclusion was also led by the literature review in the previous section 2. Therefore, after attaining the needed dataset the data mining pipeline shown in Figure 2 will be followed.

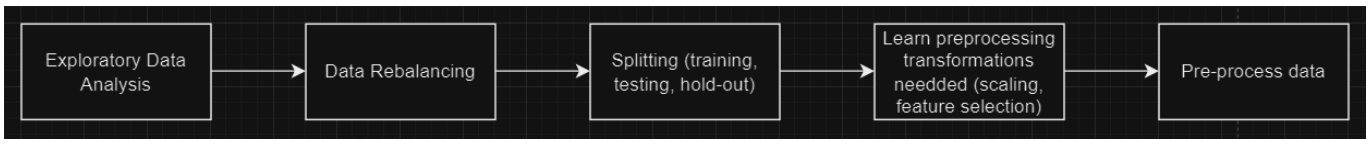


Figure 2: Data mining pipeline

The dataset used will be further elaborated in the next section 3. The overall project structure is shown in Figure 3.

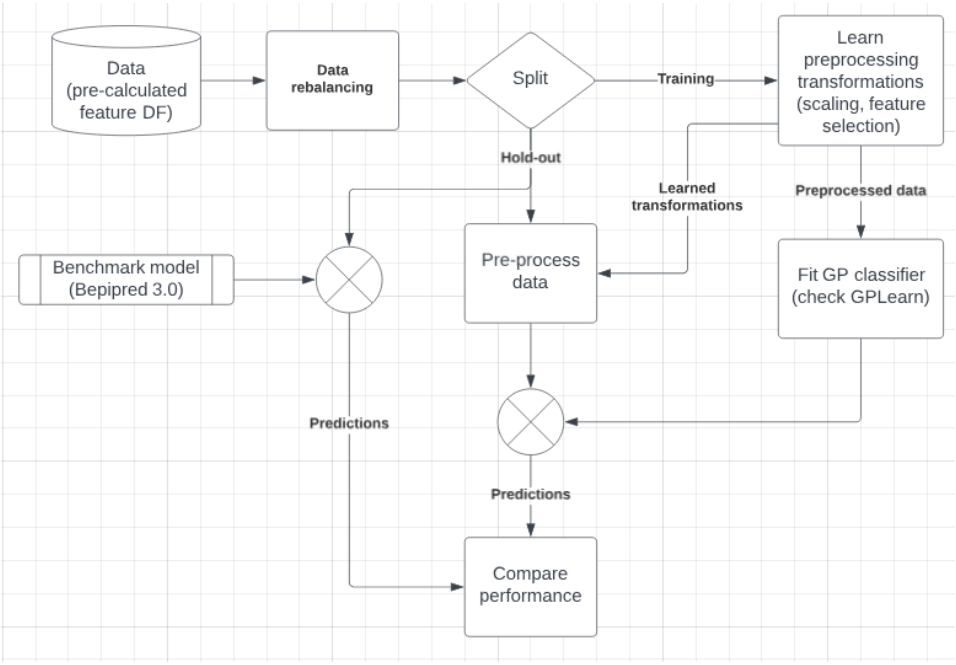


Figure 3: Project Requirements

To produce this project, Python will be the main programming language used as there are many genetic programming libraries available free to use. Furthermore, this project will be conducted on Jupyter Notebook, which is a popular tool for developing, documenting and data analysis in a wide range of scientific and computational fields. Jupyter Notebook was chosen due to its interactive and exploratory nature, it provides an interactive environment in which code can be written and executed in small chunks. This is very helpful for exploring large datasets and experimenting with models, parameters and data preprocessing steps. It also allows for immediate visualization of results and data, which will be very useful in understanding patterns in the dataset and model. Furthermore, Jupyter Notebook provides integration of popular Python libraries such as scikit-learn which allows the creation of plots and graphs for visualization. Lastly, it ensures that the project is reproducible, the transparency makes it easy for others to verify and reproduce the results of projects.

The end product of this project will be a well-constructed genetic programming model for linear B-cell epitope prediction. This model will be constructed using a robust data mining pipeline as shown in Figure 2, the model will then be compared against BepiPred 3.0, a current state-of-the-art epitope prediction model. The results will be compared and a stance will be taken on the feasibility of using genetic programming for epitope prediction. The project will be constructed in a well-documented Jupyter Notebook, which will be available publicly.

4 Method

Having gained valuable insight from the literature review and requirements established, this section lays out the procedures taken to create the project. The overall structure which the project will follow is outlined in Figure 3.

4.1 Data Source

The data used in this project was provided by the supervisor Felipe Campelo and derived as specified in (Ashford et al. 2021). The dataset was created by parsing and consolidating data retrieved from online databases the Immune Epitope Data Base (IEDB) (Vita et al. 2019), Genbank (Benson et al. 2012) and UniProtKB (*UniProt: the universal protein knowledgebase in 2021* 2021). Specific datasets for each pathogen were retrieved from IEDB and filtered such that only peptides marked as linear B-cell epitopes and non-epitopes of lengths between 8 and 25 were selected. Furthermore, labels of

"Positive", "Positive-High", "Positive-Intermediate" and "Positive-Low" were grouped under "Positive". Protein information was retrieved from Genbank based on protein IDs available in the epitope data. Conflicting class data were removed using simple majority and ties were removed. (Ashford et al. 2021). Lastly, features were extracted using the state-of-the-art feature embedder for protein data, ESM-1b.4 (Rives et al. 2019). The resulting dataset contained 13 information columns named: Info_PepID, Info_organism_id, Info_protein_id, Info_pos, Info_AA, Info_pubmed_id, Info_epitope_id, Info_host_id, Info_nPos, Info_nNeg, Info_type, Info_window, Info_cluster. These columns provide general information about the origin of the observations, for example, Info_organism_id has information on the origin of the peptide, Info_AA has the specific amino acid and Info_cluster has the information for the general grouping of the data. The dataset also contains one Class column, which is labelled 1 or -1 depending on whether the amino acid is an epitope, 1 represents epitopes and -1 represents non-epitopes. Lastly, the dataset has 1293 feature columns which contain features that were calculated for each observation using ESM-1b.4, Figure 4 showcases a preview of the dataset received.

Info_Pepl	Info_orga	Info_prot	Info_pos	Info_AA	Info_pubr	Info_epit	Info_host	Info_nPos	Info_nNe	Info_type	Info_winc	Info_clust	feat_esm:	feat_esm:	feat_esm
P31627.2::	11662	P31627.2	569	Y	15308714	75706	10000195	0	1	Epitope cc	WTREKAQ	52	0.105333	0.204399	-0.03038
P31627.2::	11662	P31627.2	570	S	15308714	75706	10000195	0	1	Epitope cc	TREKAQYS	52	0.080614	0.112012	0.209364

Figure 4: All virus dataset with ESM1b features

4.2 Dataset

In addition to this dataset, which contains all virus information, the supervisor Felipe Campelo also provided a spreadsheet containing taxonomy keys for each virus available in the dataset, showcased in Figure 5.

	ScientificName	Rank	UID	nPos	nNeg	nTot	IDs
0	Viruses	superkingdom	10239	6382	10201	16583	11320,383586,387147,370128,387139,382832,39280...
1	Riboviria	clade	2559587	4555	8544	13099	11320,383586,387147,387139,382832,392809,38283...
2	Pararnavirae	kingdom	2732397	410	188	598	11780,11908,2169971,10408,10407,35269,11676,11...
3	Artverviricota	phylum	2732409	410	188	598	11780,11908,2169971,10408,10407,35269,11676,11...
4	Revtraviricetes	class	2732514	410	188	598	11780,11908,2169971,10408,10407,35269,11676,11...

Figure 5: Taxonomy Data

As shown in Figure 5 the virus dataset can be grouped through biological classification from "Viruses", which contains all virus data to "Lentivirus" which will only contain data on the specific genus. This project will focus on extracting all Lentivirus data to train and model the genetic programming classifier.

Lentivirus is a non-oncogenic virus, meaning that they do not have the ability to cause cancer or transform normal cells into cancerous ones, in the Orthoretrovirinae family. Lentiviruses cause infections in a limited number of hosts and have a long incubation period ranging from months to years. Lentiviruses also cause progressive damage to organs and tissues, becoming lethal in many occurrences. A well-known virus in the Lentivirus genus includes HIV, which is the cause of acquired immunodeficiency syndrome (AIDS) in humans. With AIDS the human body is unable to fight off infections and cancers due to the extensive damage to the immune system (Shuljak 2006). As there are no current cures for HIV, there are substantial benefits in attempting to create a linear B-cell epitope classifier for this virus. Accurate predictions of linear B-cell epitopes causing HIV can assist researchers in the study of the disease and potential cures.

4.3 Data Extraction and Cleaning

With the given dataset and taxonomy spreadsheet, it was possible to create a dataset of only Lentivurs data. Firstly, both datasets were loaded onto Jupyter Notebook, and then all IDs related to Lentivirus were extracted from the taxonomy spreadsheet. The extracted IDs are then processed so empty spaces are removed and commas are added, this is added to two lists, one containing the lentivirus IDs as a string and the other as integer values.

```
#extract all IDs related to lentivirus
tmp = taxonomy_keys.iloc[8]['IDs']

#remove commas + whitespace + create list
lentivirusID_list = lentivirusID = [id.strip() for id in tmp.split(',')]

#turn string to int
lentivirusID = [int(x) for x in lentivirusID_list]
```

These lentivirusIDs are isolated from the dataset containing all viruses and a new dataset is created called Lentivirus. A check for missing values was done, which showed that there were no missing values in the Lentivirus dataset. Furthermore, the data type of the columns was done, to ensure that the data was consistent. As shown in Figure 6, there were no missing values and the data types were consistent so no work had to be done to clean the data further.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1795 entries, 0 to 1794
Data columns (total 18 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Info_PepID        1795 non-null   object
 1   Info_organism_id  1795 non-null   int64
 2   Info_protein_id   1795 non-null   object
 3   Info_pos          1795 non-null   int64
 4   Info_AA           1795 non-null   object
 5   Info_pubmed_id    1795 non-null   object
 6   Info_epitope_id   1795 non-null   object
 7   Info_host_id      1795 non-null   object
 8   Info_nPos         1795 non-null   object
 9   Info_nNeg         1795 non-null   object
 10  Info_type         1795 non-null   object
 11  Info_window       1795 non-null   object
 12  Info_cluster      1795 non-null   int64
 13  Class             1795 non-null   int64
 14  feat_esm1b_0      1795 non-null   float64
 15  feat_esm1b_1      1795 non-null   float64
 16  feat_esm1b_2      1795 non-null   float64
 17  feat_esm1b_3      1795 non-null   float64
dtypes: float64(4), int64(4), object(10)
memory usage: 252.5+ KB
```

Figure 6: Missing value and data type check of the Lentivirus dataset

4.4 Data Balancing

Having retrieved the Lentivirus dataset and done any cleaning needed, the next step was data balancing. In data mining, the imbalance problem is the name given to the process of inducing classification models when the distribution of class labels is not uniform. If a model is trained on imbalanced data, the models' predictions will be biased towards the majority class and predicting minority classes becomes challenging. One way to tackle imbalanced learning is through re-sampling the dataset, which includes over-sampling minority classes or under-sampling the majority classes, based on the number of samples for each class (Kang et al. 2019).

To tackle class imbalance in the Lentivirus dataset, the class balance of the dataset was visualized as shown in Figure 7

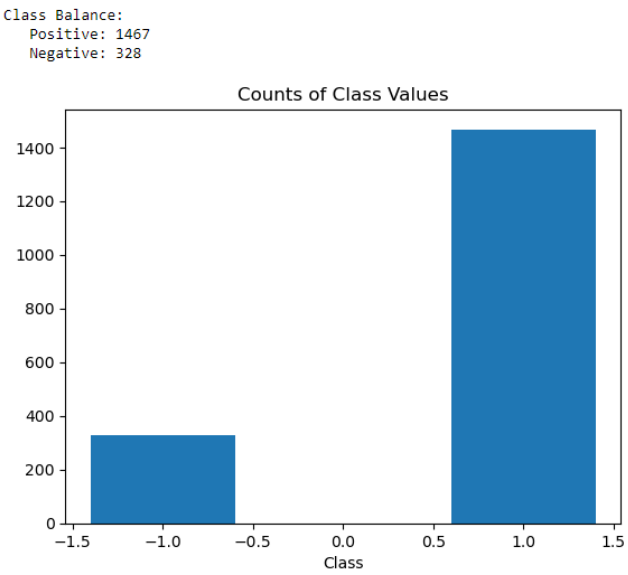


Figure 7: Initial investigation of the class balance of the Lentivirus dataset

As seen above, the dataset was very biased to positive, epitope data comprising over 81.7% of the dataset. Therefore, the appropriate approach to tackle such imbalance would be under-sampling the data, that is removing data of the majority class in the dataset. To do this, the dataset was grouped by the Info_cluster column, and data in each cluster would have similar patterns or behaviour, therefore, balancing the data in each cluster results in minimal information loss leading to a more accurate model. A visualisation of the data balance of each cluster was then produced as shown in Figure 8.

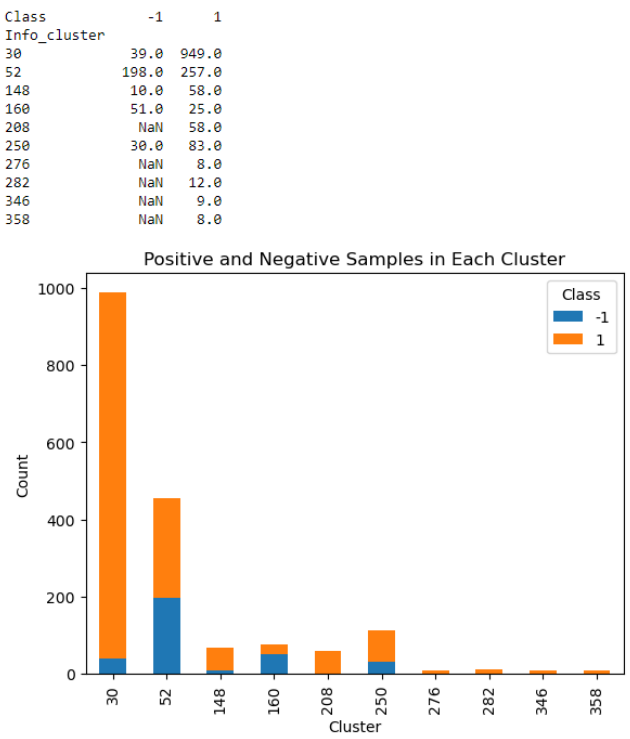


Figure 8: Class balance of each Info_cluster

As seen in the above Figure, only clusters 30, 52, 148 and 250 have data on both epitopes and non-epitopes and 208, 276, 282, 346 and 358 only have data on epitopes. Since the clusters which only have positive labels only have a total of 37 data points, these clusters will not be touched, the rest of the clusters which contain both labels will all be undersampled. To achieve this a method

was implemented which looped over each cluster and randomly under-sampled the majority class, by removing data, until the specific cluster had an equal split of both labels.

```
import numpy as np
#balance class for each cluster which has class imbalance
clusters_to_undersample = [30,52,148,250]
undersampled_dataset = pd.DataFrame()

#loop over each cluster and randomly remove majority class until 50:50
#create new dataset with undersampled data
np.random.seed(42)
for cluster in clusters_to_undersample:
    cluster_data = lentivirus_clusters[lentivirus_clusters['Info_cluster']
        == cluster]
    minority_class_len = int(cluster_data['Class'].value_counts()[-1])
    majority_class_len = cluster_data['Class'].value_counts()[1]

    majority_class_idx = cluster_data[cluster_data['Class'] == 1].index
    minority_class_idx = cluster_data[cluster_data['Class'] == -1].index

    random_majority_idx = np.random.choice(majority_class_idx,
        minority_class_len, replace=False)

    undersampled_idx = np.concatenate([minority_class_idx,
        random_majority_idx])
    undersampled_cluster = cluster_data.loc[undersampled_idx].copy()

    undersampled_dataset = pd.concat([undersampled_dataset,
        undersampled_cluster])

#add the other clusters back to undersampled_dataset
clusters_to_add = [160,208,276,282,346,358]
for cluster in clusters_to_add:
    c= lentivirus_clusters[lentivirus_clusters['Info_cluster'] == cluster]
    idx = lentivirus_clusters[lentivirus_clusters['Info_cluster']
        == cluster].index
    add_cluster = c.loc[idx].copy()
    undersampled_dataset = pd.concat([undersampled_dataset,add_cluster])

undersampled_dataset.reset_index(drop=True, inplace=True)
undersampled_dataset
```

As shown in the above code, the clusters to undersample were manually selected after looking through the graphs, a random seed of 42 was initialized so the random numbers generated become reproducible. Then a loop is created in which, the class count is calculated and stored in variables minority_class_len and majority_class_len respectively. Then the classes are indexed and stored in variables, using these indexes random undersampling is done by randomly selecting indices from the majority class until they are equal to the minority class, and replace=False is added so that each index chosen is unique. This resulted in a more balanced dataset with 328 instances of -1 labels and 397 instances of 1 labels in the dataset. This undersampling reduced the number of data points from 1,795 to 725.

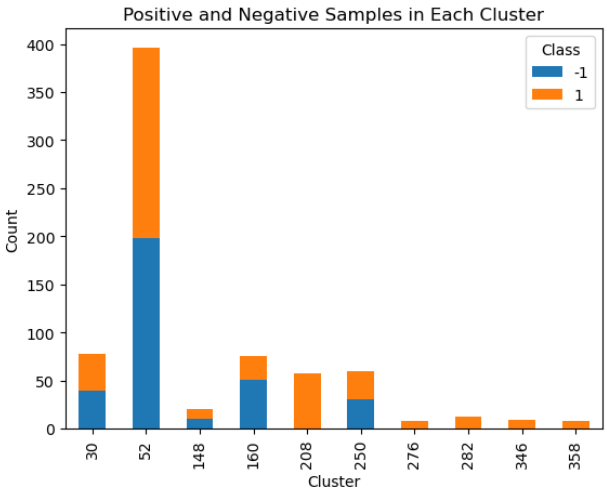


Figure 9: Balanced dataset after under-sampling was applied

4.5 Data Splitting

After the dataset was balanced, the next step was to split the dataset into train, test and hold-out sets. Data splitting is done to avoid over-fitting of the final model. Overfitting occurs when the model fits exactly against its training data and is unable to perform accurately on unseen data. Data splitting is an effective way to tackle overfitting as the model will be trained on the train set, which will be the largest of the 3 data splits to ensure that the model is able to learn the complex relationships between the dataset's features and the target variable. Then the test set will be used for hyper-parameter tuning and the holdout set will be used for comparison with other models, this ensures that the same dataset is not re-used.

A cluster-based split will be conducted to ensure that the training, test and hold-out sets will all be balanced. To achieve this GroupKFold was used. GroupKFold is a cross-validation technique in machine learning, which is able to split data with grouped or clustered data points remaining together in the cross-validation process. Through data splitting the train set had 396 data points, the test set had 165 data points and the holdout set had 164 data points, this resulted in a split of around 55% for the train set, 23% for the test set and 22% for the holdout set. Using GroupKFold allowed for the train, test and holdout sets to be balanced.

4.6 Data Pre-processing

The last step before modelling is preparing the data so that a genetic programming classifier is able to train on it. In this step, features which are unlikely to contribute to the predictive performance of the models are removed. Therefore, the 13 information columns shown in Figure 4 are removed, the only columns remaining for modelling are the feature columns (extracted from ESM-1b.4) and the class column. Furthermore, outliers are searched for and dealt with as outliers can skew the results of data analyses and hamper model performance. Outliers are data points that do not conform with the predominant pattern observed in the data, to find outliers in the dataset the skewness of the train set was analysed. Skewness is used to find outliers in data as outliers cause the skewness of the data to be far from zero (Heymann et al. 2012), therefore for the train set an outlier was defined as a datapoint which has a skewness of over 1.5 or below -1.5.

```
#check skewness of features
features=data.iloc[:, 13:-1]
skewness = features.skew()
skewed_columns_greater = skewness[skewness > 1.5].index
skewed_columns_less = skewness[skewness < -1.5].index
```

```
print(skewed_columns_greater)
print(skewed_columns_less)
```

The above code checked the skewness of all feature columns, resulting in feature feat_esm1b_877 being found as an outlier. As showcased in Figure 10, feat_esm1b_877 has a skewness of around -1.7.

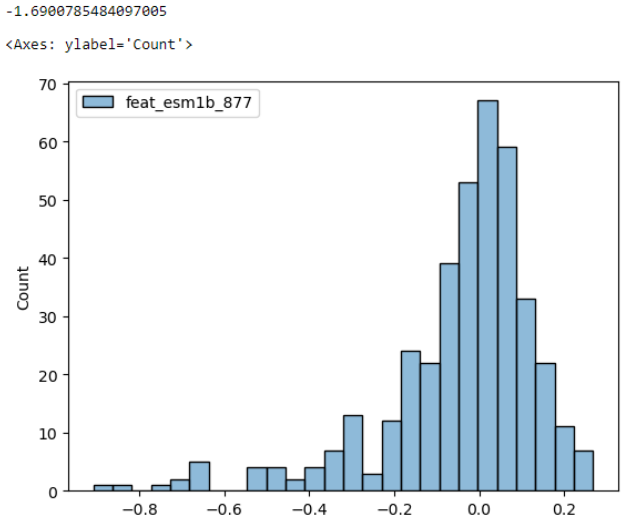


Figure 10: Diagram representing the skewness of feature feat_esm1b_877

As the training set has over 1000 feature vectors, the dataset can be considered as having high dimensionality. To further understand this dataset, t-distributed Stochastic Neighbor Embedding (t-SNE) was used to represent high-dimensional data in a lower-dimensional space.

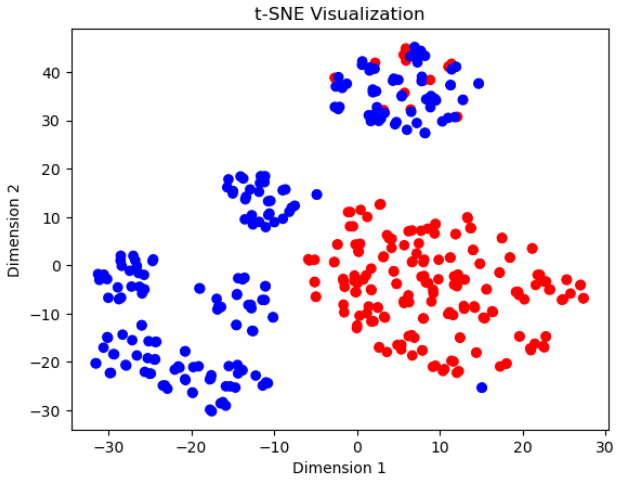


Figure 11: t-SNE graph of the training dataset

As shown in Figure 11, the data is separated well, the red dots represent class labels of -1 and the blue dots represent class labels of 1, representing non-epitope and epitope data in the training set. In the upper region of the graph (10,35), there is a combination of both classes, however, overall there is good separation between the two classes. The distance between the points shows the similarities between them, therefore, if data points of two different classes are close to each other it may lead to the model making inaccurate predictions. The combination of the data points at around (10,35) and (15,-25) in the t-SNE graph will not impact the overall model, as feature reduction was done as described in Section 4.6.2, which will remove these points in the final data being fed into the genetic programming model (the skewed data will also likely be removed).

4.6.1 Data Scaling

The next step is data scaling, also known as normalisation, in which values of features and variables are transformed in order to bring them to a similar scale or range. This is an important step as many machine learning algorithms are sensitive to the scale of input features and can produce better results when the data is normalized. Furthermore, features in the training set have negative values, which may interfere with modelling. To normalize the data a MinMaxScaler (sklearn 2023) was used, which is able to scale and translate each feature individually such that it is in the given range on the training set between zero and one. This is done through their equation of :

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

$$X_{scaled} = X_{std} * (max - min) + min \tag{2}$$

The MinMaxScaler was imported to the project and used to normalize the training set as showcased in the following code:

```
# MinMaxScaler for scaling data
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
feature_cols = list(features)

#Fit the scaler to the feature data and transform it + create new dataframe
data_norm = scaler.fit_transform(data[feature_cols])
data_norm = pd.DataFrame(data_norm, columns=feature_cols)
print(data_norm.shape)
data_norm.head()
```

4.6.2 Feature Selection

The last step before modelling is feature selection, it involves the process of reducing the inputs for procession and analysis for the model and finding the most meaningful inputs. Feature selection is essential as it reduces the complexity of the dataset, by removing data which is not needed for the model, which in return improves the model as only important data is used. Furthermore, feature reduction makes modelling more efficient, as the model will not need to learn data which is not impactful for the model. By removing redundant or noisy data, it becomes easier for the model to discover meaningful patterns. There are four main categories of feature selection, which are filtering, wrapper, hybrid and embedded methods. For this project, a filtering method approach to feature selection was chosen as the other methods would be time-consuming to perform since the genetic programming classifier would take a long time to get trained to test feature selection methods such as wrapper methods.

For feature selection, the Principal Component Analysis (PCA) method was chosen, as the training set was of high dimensionality (over 1000 features). PCA is able to map data points from a high dimensional space to a low dimensional space while keeping all the relevant linear structure intact (Parveen et al. 2012). In the project, PCA was imported from the scikit-learn library and implemented as such:

```
#PCA that will retain 95% of variance
pca = PCA(n_components = 0.95, whiten=True)
datapca= pca.fit_transform(data_norm)

data_reduced = pd.DataFrame(data=datapca, index=data_norm.index)
print(data_reduced.shape)
data_reduced.head()
```

The parameter for PCA n_component is set to 0.95 so that the PCA method will retain principal components until they have collectively explained at least 95% of the variance in the training set. This is done to help preserve important information from being removed via dimensionality reduction, furthermore, retaining 95% of variance means that the noise of the data will be reduced as less impacting features are removed. By using this method, the feature columns were reduced to a manageable 130 columns, showing that most of the features were redundant and not impactful, this saved a lot of time and resources when modelling the genetic programming classifier.

4.7 Modelling

As previously mentioned for the genetic programming classifier the gplearn symbolic classifier was used due to its seamless integration and compatibility with Jupyter Notebook and the work done on the dataset. Multiple models of the genetic programming classifier were used and tested. The first model investigated used the default parameters of the glearn symbolic classifier (gplearn 2023).

```
from gplearn.genetic import SymbolicClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
from sklearn.metrics import f1_score, accuracy_score, classification_report,
    confusion_matrix

est = SymbolicClassifier(population_size=100,
                         generations=20,
                         tournament_size=20,
                         stopping_criteria=0.0,
                         verbose=1,
                         random_state=42)

target = data['Class']
X_train, X_test, y_train, y_test = train_test_split(data_reduced, target
    , test_size=0.2, random_state=42)
est.fit(X_train, y_train)
```

As shown in the above code, the gplearn genetic programming classifier was imported and initialised as est using default parameters as per (gplearn 2023). Then the training set was split using the train_test_split function from the scikit-learn library into training and testing sets for the model, a test size of 0.2 ensured that 80% of the data was used for training at 20% was used for testing. The data_reduced was the dataset after the previous methods such as feature selection were all completed. Then the model was fit in est.fit. An initial classification report and AUC score was composed with the test set:

```
y_pred = est.predict(X_test)

report = classification_report(y_test, y_pred, zero_division=0)
print("Classification Report:\n", report)

y_prob = est.predict_proba(X_test)[:, 1]
auc_score = roc_auc_score(y_test, y_prob)
print("AUC Score:", auc_score)
```

Giving the following results:

```
Classification Report:
              precision    recall  f1-score   support
```

```
          -1       0.63      0.46      0.53        48
           1       0.42      0.59      0.49        32

    accuracy                           0.51        80
   macro avg       0.53      0.53      0.51        80
weighted avg       0.55      0.51      0.52        80

AUC Score: 0.48046875
```

Another model was created to test the effect of changing the default parameters on the measurements. Therefore, the following parameters were tested:

```
est2 = SymbolicClassifier(population_size=10000,
                          generations=200,
                          tournament_size=20,
                          stopping_criteria=0.0,
                          verbose=1,
                          random_state=42)
```

This was also useful as it tested the computational cost of the classifier. The time taken for this classifier to fit on the training set was 16.99 minutes compared to the 1.59 seconds taken with the model with the default parameters. Both the default parameter model and the model with larger parameters were tested on the holdout. As shown in Tablet 1, there was a significant increase in the accuracy and AUC score of the model. Therefore, optimal parameters need to be investigated, this will be discussed in Section 4.7.1.

Metric	Model 1		Model 2	
	(-1)	(1)	(-1)	(1)
Precision	0.44	0.46	1	0.55
Recall	0.46	0.44	0.15	1
F1-Score	0.45	0.45	0.26	0.71
Support	81	84	81	84
Accuracy	0.45	-	0.58	-
AUC Score	0.4486	-	0.5741	-

Table 1: Comparison of Model Performance

4.7.1 Hyperparameter Tuning

As shown in the previous section, there is a need to find the optimal parameters for the model for optimal performance. Therefore, hyperparameter tuning will be done. In the project, GridSearchCV is used for hyperparameter tuning, with this technique, a model for each possible combination of all the hyperparameter values provided will be made. Each of these models will be evaluated and the best combination of parameters will be given.

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer

param_grid = {
    'population_size': [100, 500, 1000],
    'generations': [5, 10, 50],
    'tournament_size': [10, 20, 50],
    'p_crossover': [0.6, 0.8, 0.9],
```

```
    'p_subtree_mutation': [0.001, 0.01, 0.1],
    'stopping_criteria': [0],
    'verbose': [1],
    'random_state': [42]
}

classifier = SymbolicClassifier()
X_tuning = tuning.iloc[:, 13:-1]
y_tuning = tuning['Class']

auc_scorer = make_scorer(roc_auc_score)
grid_search = GridSearchCV(classifier, param_grid, scoring= auc_scorer, cv=5)
grid_search.fit(X_tuning, y_tuning)

best_params = grid_search.best_params_

# Print current best parameters and their AUC score
print("Best Parameters:", grid_search.best_params_)
print("Best AUC Score:", grid_search.best_score_)
```

To implement the GridSearchCV, it was first imported and a parameter grid was constructed with the most important parameters in genetic programming, a range of values was assigned to each of the parameters. These values were derived from making informed changes to the default parameters provided by glearn, for example, the default parameter for the crossover rate is 0.9, therefore, 0.8 and 0.6 were selected. If 1 was chosen as the crossover rate, then in each generation, every pair of parents would be chosen for the crossover, and there would be no parents skipped over, meaning that an optimal solution would not be found. The best parameters gained from the GridSearchCV were the following:

```
Best Parameters: {'generations': 50, 'p_crossover': 0.9, 'p_subtree_mutation': 0.01,
    'population_size': 500, 'random_state': 42, 'stopping_criteria': 0,
        'tournament_size': 50,
    'verbose': 1}
```

A second hyperparameter tuning was done after getting the first optimal parameters, as the first hyperparameter tuning explored a wide range of hyperparameters. A second hyperparameter tuning would allow for more fine-tuning of parameters as one set of optimal parameters was found. The parameter grid for the second search was the following:

```
param2_grid = {
    'population_size': [250, 500, 750],
    'generations': [25, 50, 75],
    'tournament_size': [40, 50, 60],
    'p_crossover': [0.9],
    'p_subtree_mutation': [0.01],
    'stopping_criteria': [0],
    'verbose': [1],
    'random_state': [42]
}
```

The result of the second hyperparameter tuning was the same parameters as the first tuning run. Therefore, there is higher confidence that the hyperparameters found are the most optimal for this specific model. Therefore, using the optimal parameters for the genetic programming classifier gave the results shown in Figure 12.

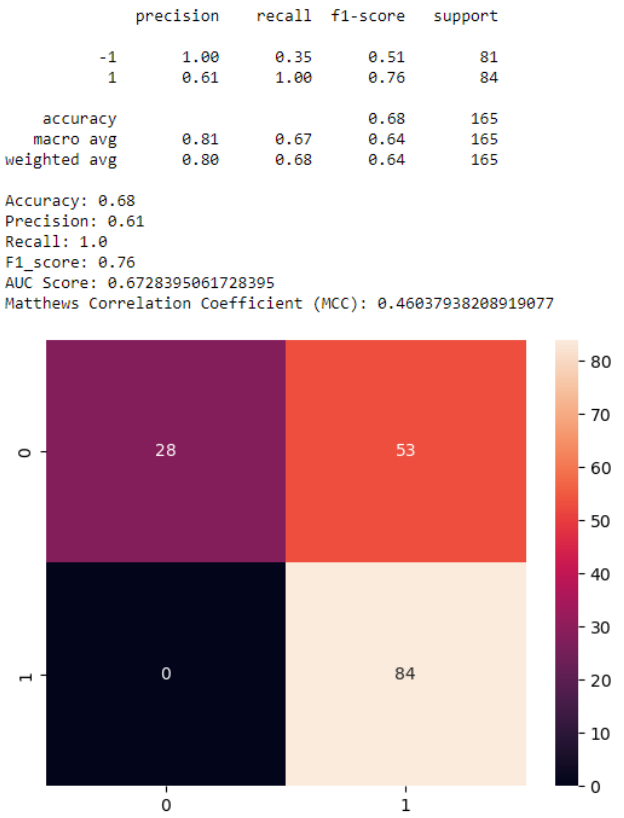


Figure 12: Final results from gplearn Symbolic Classifier

The precision of the -1 class was 1 and class 1 was 0.61, this shows the model is perfect for predicting the negative class it is always perfect. The recall score shows that the model is able to capture all actual positive instances for class 1, however, only 35% of class -1. The F1-Score is the mean of precision and recall, scores of 0.51 and 0.76 show that the model's overall performance is reasonable, but there is room for improvement. The accuracy score of 68% shows that the model is predicting correctly 68% of the time. Finally, the final AUC score, the ability to distinguish between positive and negative instances, was 0.67 which is a decent score for the model, however, could use some improvement. Overall, the models' performance was decent but could use improvement.

4.8 Comparison

For comparison, the genetic programming model was compared with BepiPred 3.0 using its web server. The protein sequences available in the holdout set were used for the BepiPred 3.0 results. The web server requires the protein sequences to be in FASTA format, to get this the previously mentioned holdout set was extracted as a CSV file, and all unique protein IDs were extracted. The dataset was sorted by Info_pos so that each amino acid in the sequence is in the correct position, for accurate prediction. A dictionary was then created to store each amino acid sequence from Info_AA, in order to create a full protein sequence. A for loop was then initiated iterating through the unique protein ID. Since some Info_pos values are missing, due to data splitting, missing Info_pos values are checked. Then a list of expected and actual positions is created, so that when the list is filled and a missing value is found the row with the missing value is deleted. This makes sure that the amino acids are placed in order. Lastly, the amino acids from Info_AA are joined together and stored as a dictionary:

```
# Create a dictionary to store the sequences for each Info_protein_id
protein_sequences = {}

# Iterate through unique Info_protein_id values
```

```
unique_protein_ids = test_data['Info_protein_id'].unique()
for protein_id in unique_protein_ids:
    # Filter the DataFrame for the current Info_protein_id
    protein_test_data = test_data[test_data['Info_protein_id'] == protein_id]

    # Check for missing Info_pos values
    expected_positions = set(range(1, protein_test_data['Info_pos']
        .max() + 1))
    actual_positions = set(protein_test_data['Info_pos'])

    # Remove rows with missing positions
    missing_positions = list(expected_positions - actual_positions)
    protein_test_data = protein_test_data[~protein_test_data['Info_pos'].
        isin(missing_positions)]

    # Create the sequence by joining Info_AA values in order
    sequence = ''.join(protein_test_data['Info_AA'])

    # Store the sequence in the dictionary
    protein_sequences[protein_id] = sequence

# Now, protein_sequences contains sequences for each unique Info_protein_id
for protein_id, sequence in protein_sequences.items():
    print(f'Info_protein_id: {protein_id}')
    print(f'Info_AA: {sequence}')
    print()
}
```

Finally, to get the FASTA format, > is added in front of the protein ID and the amino acid sequence is given underneath. Unfortunately, BepiPred 3.0 only accepts amino acid sequences of lengths between 10 and 6000, therefore, some amino acid sequences had to be removed, and the final amino acids used were the following:

```
>AAC03762.1
REITFIYKSSCTDSDHCQEYQCKKVNLNSSDSERQQVEETFNLIGCIERTHVFCHTG
>ABI20203.1
KRWIILGLNKIVRMYGPKEPFRDYVDRFYK
>P16082.1
SNSVRVEDVTNTAEYWGFK
>P23426.1
TGKIPWILLPGR
```

The default parameters were used for BepiPred 3.0 and the results were downloaded as a zip folder, the predictions were given as a FASTA file and each amino acid was either a capital letter, for epitopes, or a lowercase letter, for non-epitope predictions. To evaluate the model using the metrics discussed in the literature review, the FASTA file was made into a dataset:

	Info_protein_id	Info_AA	Class
0	ABI20203.1	k	1
1	ABI20203.1	r	1
2	ABI20203.1	w	1
3	ABI20203.1	i	1
4	ABI20203.1	i	1
...	...	...	...
113	P23426.1	L	-1
114	P23426.1	I	1
115	P23426.1	p	1
116	P23426.1	g	1
117	P23426.1	r	1

Figure 13: Dataset of BepiPred 3.0 predictions

Each amino acid was labelled as positive or negative 1 depending on if the amino acid was capitalised or not. The holdout dataset was reduced to the same columns as the dataset with the BepiPred 3.0 predictions. Furthermore, all Info_protein_IDs which had to be removed due to the BepiPred 3.0s limit of over 10 amino acid sequences, were also removed from the holdout dataset. This allowed us to gain performance metric scores for the BepiPred 3.0 predictions, shown in Figure 14.

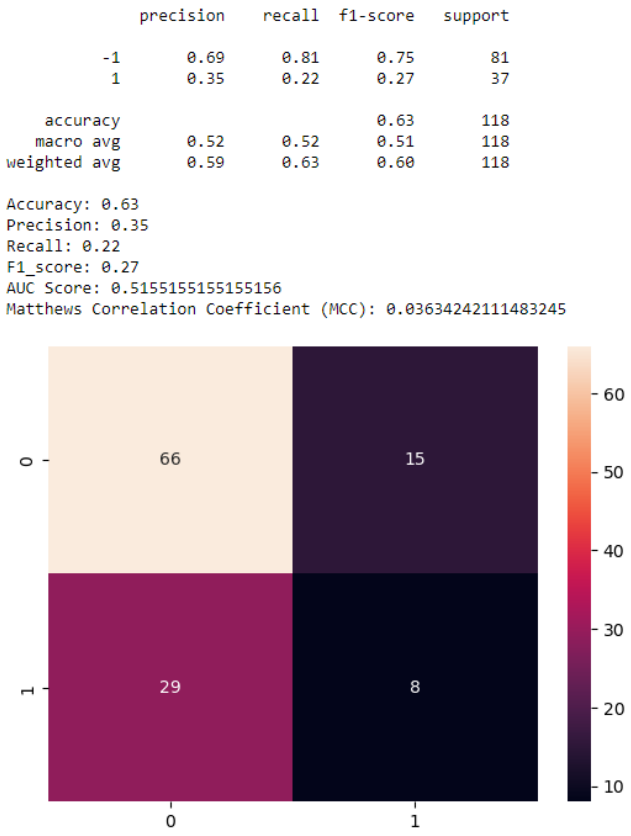


Figure 14: Performance metrics of the BepiPred 3.0 predictions

For a fair comparison between the two models, predictions using the gplearn classifier were made on a modified holdout set, with the same protein sequences removed.

Comparing the genetic programming classifier against the BepiPred 3.0 predictions shows that the genetic programming classifier is clearly superior in this use case. As shown in Table 2, the gplearn genetic programming classifier outperforms the BepiPred 3.0 in every metric. BepiPred 3.0s model has a lower AUC score than the genetic model, furthermore, the MCC of the BepiPred 3.0

model was 0.0363 suggesting that the prediction is close to a random guess and it indicated the poor ability to distinguish epitopes and non-epitopes.

Metric	Genetic		BepiPred	
	(-1)	(1)	(-1)	(1)
Precision	1	0.41	0.69	0.35
Recall	0.35	1	0.81	0.22
F1-Score	0.51	0.58	0.75	0.27
Support	81	37	81	37
Accuracy	0.55	-	0.63	-
AUC Score	0.673	-	0.5155	-
MCC Score	0.377	-	0.0363	-

Table 2: Comparison of Model Performance between Genetic Programming and BepiPred 3.0

5 Project management

Within the submitted project proposal an initial project plan was included, and key milestones and critical paths were detailed. This project plan was discussed with the project supervisor and a revised edition is available in the submitted meeting notes, available in the Appendix A. This project plan was not very strict and allowed for enough time for the project to be completed in adequate time, overall the project plan provided helpful guidance for meeting deadlines and project objectives to be met. However, as expected, the project plan was not followed completely, there was a two-week delay in the project due to family reasons, furthermore, some parts of the project took longer than expected to complete, these details can be seen in the meeting notes which have work being added to the next work weeks. Section 5.1 details the CRISP-DM methodology followed throughout the project and how it affected the project. Version control is discussed in section 5.2, section 5.3 discusses the project plan followed and provides a Gantt chart of the project. Lastly, section 5.4 discusses the bi-weekly meetings with the supervisor.

5.1 CRISP-DM

A modified version of the CRISP-DM framework was adopted for the development of the project. CRISP-DM stands for Cross-Industry Standard Process for Data Mining and is widely used in the industry to guide and organize steps involved in completing data-driven projects. The CRISP-DM framework consists of six main phases phases as shown in Figure 15.

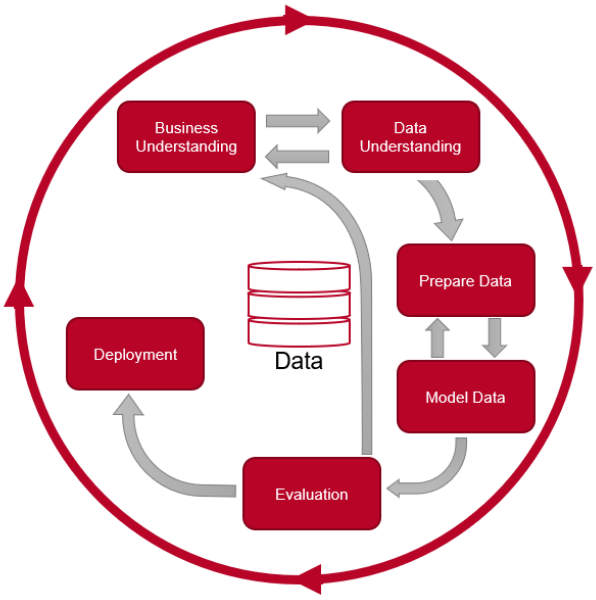


Figure 15: CRISP-DM framework (Fuchs 2020)

In the business understanding phase, project goals and requirements are defined, resulting in a rough descriptive plan for the project. This was completed in the project plan and through a few meetings with the supervisor. In the data understanding phase, an initial overview of the available data and its quality is observed, this was concluded with the literature review conducted at the start of the project. In this phase, problems with the quality of data are defined, this was accomplished at the start of the project when the dataset was first received. The next phase includes data preparation, where the final data set is created, this process is evident in Section 4, where the whole process is detailed. The next phase is modelling, where the models are trained and optimized, this process is also showcased in 4. The next phase is evaluation, in which the model is compared with an equivalent model, also showcased in 4. The deployment phase was not done in the modified CRISP-DM framework used for this project, as it was not necessary. The modified CRISP-DM framework used is displayed in Figure 16

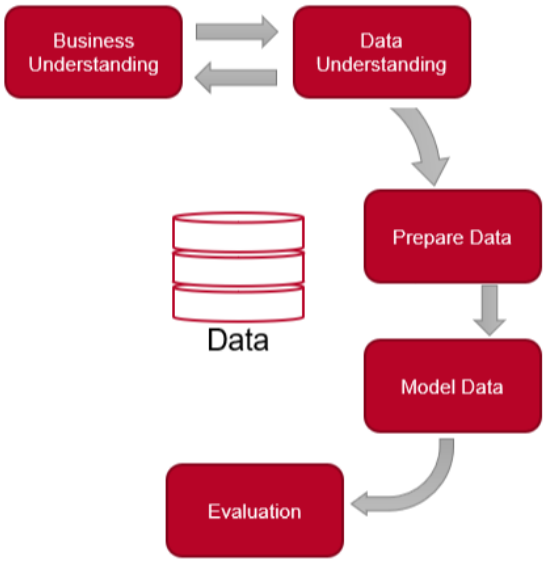


Figure 16: Modified CRISP-DM framework

In addition to this modified CRISP-DM framework used, bi-weekly meetings with the supervisor of this project were conducted. This was done to ensure that clear goals and objectives for each phase of the modified CRISP-DM process were thoroughly discussed and a clear understanding was developed. This greatly facilitated effective progress tracking and coordination throughout the project.

5.2 Version Control

Version control is a vital project development practice and it serves as a safety net to protect the source code from irreparable harm (GitLab 2023). Version control allows tracking of changes made in a project, allowing to view of the entire project development history and allowing rollbacks to an older version of the project.

In the project proposal, one risk assessed was the potential risk of the loss of the project. This was handled via the auto-save feature of Jupyter Notebook, which automatically saves the changes made to the project, furthermore, every time work was conducted on the project the file was saved on another laptop system in case an issue arose on the primary workbench. Lastly, the project was uploaded on GitHub when objectives were reached to allow for rollback to previous versions. For the dissertation write-up, Overleaf was used, an online cloud-based LaTeX editor. Overleaf is cloud-based an autosave all work done, therefore, there was no danger of losing the dissertation write-up. Overall, version control was a vital part of the project as on many occasions during the data mining stages of the project, bugs and errors occurred. Therefore, rolling back to the previous version of the second laptop system was very useful.

5.3 Gantt

To visualise the project plan a Gantt chart was utilised, shown in Figure 17. The main objectives to achieve are detailed and a deadline is also provided, as mentioned earlier this Gantt chart was not followed exactly. As seen in the Gantt chart the final deadline is 12/09, which allocated just over two weeks of buffer zone in case unexpected roadblocks occurred. This played a vital role as there was a delay of two weeks in the project due to having to go abroad for family reasons.

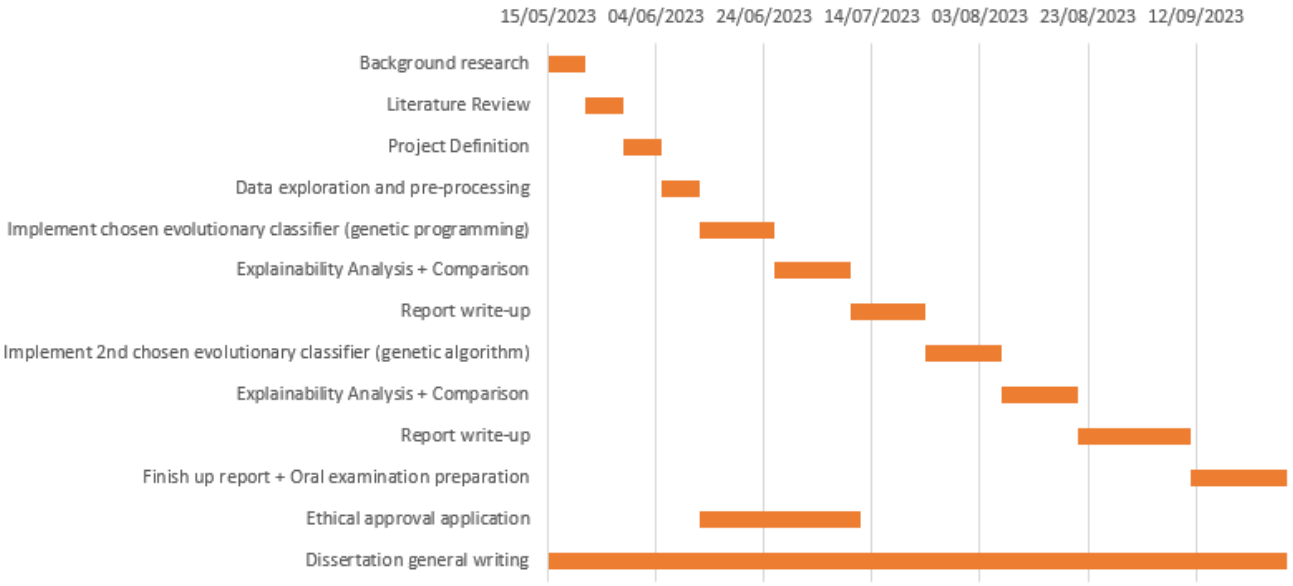


Figure 17: Gantt chart

Overall, the use of the Gantt chart allowed to mark the key project milestones to achieve, such as the ethical approval application and project definition deadlines. This was edited with the contribution of the project supervisor in one of the bi-weekly meetings, available in the Appendix A.

Using a Gantt chart was definitely appropriate as it was not time-consuming to fill out and acted as a rough weekly guide on what to complete each week. The use of a critical path and key milestones further, aided in ensuring enough time was put into development and finishing key features.

5.4 Meetings

Meetings were the most useful project management tool throughout the whole project. Bi-weekly meetings were arranged with the supervisor, there was a chance to discuss work done, lay out future objectives and timelines and discuss any problems or roadblocks of the project with the supervisor. The bi-weekly meetings were thirty minutes long they proved extremely helpful, after each meeting a meeting log was filled and given to the supervisor. These meeting logs detailed the discussion had with the supervisor, a brief description of the discussion was detailed and actions required were stated. The meeting logs are available in Appendix A.

6 Evaluation

As showcased throughout the method, Section 5, multiple evaluation metrics were used to explore the feasibility of the genetic programming classifier presented. The evaluation metrics used were precision, recall, f1-score, accuracy, AUC score and MCC scores. These were derived from the extensive literature review made. These metrics have been shown to be used to evaluate models and have also been used to compare against other models. An explanation of these performance indicators is available in Appendix D.

Table 2 provides a good insight into the performance of the proposed model. The model had a precision score of 1 for non-epitopes and 0.41 for epitopes, meaning that the model is able to predict an epitope correctly 41% of the time. The model also had a recall of 1, showing that the model was able to identify all true positive data, meaning that it was able to identify all actual epitopes. An AUC score of 0.673 suggests that the model has a moderate ability to discriminate between epitopes and non-epitopes, and the MCC score of 0.377 suggests that there is a weak correlation between the predicted and actual classifications. In terms of practicality, the model is able to capture all true epitopes but also predicts a lot of false positives. This is common in linear B-cell epitope prediction models, as even BepiPred 3.0 has a lot of false positive results (Clifford et al. 2022). Having a large amount of false positive results poses a problem as if this model is commercially used for epitope prediction, money and time would be wasted in exploring or synthesising the false positive epitope. Therefore, even though the comparison with BepiPred 3.0 is reassuring that the model is good, it still needs further refinement to ensure that minimal false positive predictions are made before it can be used for applications such as epitope synthesis. Also, there is no ethical implication for this project as the sources of the datasets used are based on publicly available data sources, and the data does not contain any personal information and is available for use without restriction and the model itself has no ethical implications as it does not impact any person.

In terms of the methodology used, it was very successful and appropriate for this project. Through the robust data-mining pipeline used and the extensive tweaking of the dataset, it was ensured that the proposed model would have the most optimal training set for the best results. This was showcased by the final proposed genetic programming model, as it was able to outperform the current state-of-the-art model BepiPred 3.0. The project was able to be performed on a computer with a low-end CPU, fitting the gplearn Symbolic Classifier with default parameters took 1.15 seconds, when tested with large parameters of 10,000 population, 200 generations and 20 tournament size, the time taken to fit the model were 18.02 minutes. In terms of training the model, it is computationally inexpensive. However, hyper-parameter tuning using GridSearchCV with a combination of 3 parameter values to test per parameter, took around 1 hour and 45 minutes to complete which was quite costly computationally. Overall, the proposed model was effective and efficient as it was quick and not very computationally expensive, which could allow researchers to save costs on creating different models.

Since there was a tight deadline for the project, in-depth tweaking of the proposed model was not possible. To improve the model further, testing other hyper-parameter tuning methods would prove fruitful. Approaches such as Bayesian optimization could be tested as it is shown to be very effective. Furthermore, the model was trained on a limited dataset of only Lentivirus data, training the model on a larger dataset with more generalised data would be an interesting test for the model. The ability of the proposed model to outperform BepiPred 3.0 could be attributed to the fact that the model was trained on organism-specific data as showcased by Ashford et al. (Ashford et al. 2021). Therefore, training and testing on more generalized data would provide further insight into the feasibility of the genetic programming approach to epitope prediction.

Overall, this project was successful as a robust genetic programming classifier for linear B-cell epitopes was proposed, which was able to outperform BepiPred 3.0. The model was also not very computationally expensive, making it beneficial for testing multiple models. This showcases that using genetic programming is very feasible in the epitope prediction problem and that it should be further experimented on by researchers.

7 Conclusion

In this dissertation, we proposed exploring genetic programming to tackle the epitope prediction problem. We began with an extensive literature review, showcasing the evolution of epitope prediction models, the limitations and stating the current state-of-the-art epitope prediction models and it was showcased that genetic programming was never used to solve this problem. Through this literature review requirements for this project were stated. The method used to create the gplearn model was extensively discussed and the feasibility was evaluated.

Overall, the aim of this project was to present a genetic programming model for predicting and classification of linear B-cell epitopes and explore the feasibility of such an approach. Extensive work was done to ensure the dataset used for training the model was of high quality, by following a robust data-mining pipeline. Comparisons with the current state-of-the-art linear B-cell epitope prediction model, BepiPred 3.0, were made to check the feasibility of using genetic programming in the epitope prediction problem. The proposed genetic programming model performed much better than BepiPred 3.0, which showed the promise of exploring genetic programming and other evolutionary techniques in the epitope prediction problem. Based on the results of the genetic programming model, it is evident that further research in using evolutionary classification systems for epitope prediction would be beneficial.

7.1 Limitations and Future Work

Despite the promising results of the genetic programming model in epitope prediction, there is much more future work which can be done to further improve and investigate the feasibility of this approach. Firstly, the model has to be tested on a larger and more diverse dataset, since the model was trained and tested on a lentivirus-specific dataset testing the model on a different dataset would allow further insight. Furthermore, testing other epitope prediction models against the proposed genetic programming model and BepiPred 3.0 would provide more insight into where the gplearn model stands in the current epitope prediction landscape. As mentioned in the evaluation, exploring and evaluating different hyper-parameter approaches could further increase the performance of the model. Adding these tests was, unfortunately, outside the scope of this project due to time constraints.

References

Ashford, J., Reis-Cunha, J., Lobo, I., Lobo, F. & Campelo, F. (2021), 'Organism-specific training improves performance of linear b-cell epitope prediction', *Bioinformatics* **37**(24), 4826–4834.

Bäck, T., Fogel, D. B. & Michalewicz, Z. (2018), *Evolutionary computation 1: Basic algorithms and operators*, CRC press.

Bahai, A., Asgari, E., Mofrad, M. R., Kloetgen, A. & McHardy, A. C. (2021), 'Epitopevec: linear epitope prediction using deep protein sequence embeddings', *Bioinformatics* **37**(23), 4517–4525.

Benson, D. A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. & Sayers, E. W. (2012), 'Genbank', *Nucleic acids research* **41**(D1), D36–D42.

Blythe, M. J. & Flower, D. R. (2005), 'Benchmarking b cell epitope prediction: underperformance of existing methods', *Protein Science* **14**(1), 246–248.

Chen, J., Liu, H., Yang, J. & Chou, K.-C. (2007), 'Prediction of linear b-cell epitopes using amino acid pair antigenicity scale', *Amino acids* **33**, 423–428.

Clifford, J. N., Høie, M. H., Deleuran, S., Peters, B., Nielsen, M. & Marcatili, P. (2022), 'Bepipred-3.0: Improved b-cell epitope prediction using protein language models', *Protein Science* **31**(12), e4497.

Collatz, M., Mock, F., Barth, E., Hölzer, M., Sachse, K. & Marz, M. (2021), 'Epidope: a deep neural network for linear b-cell epitope prediction', *Bioinformatics* **37**(4), 448–455.

Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. (2018), 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* .

EL-Manzalawy, Y., Dobbs, D. & Honavar, V. (2008), 'Predicting linear b-cell epitopes using string kernels', *Journal of Molecular Recognition: An Interdisciplinary Journal* **21**(4), 243–255.

Emini, E. A., Hughes, J. V., Perlow, D. & Boger, J. (1985), 'Induction of hepatitis a virus-neutralizing antibody by a virus-specific synthetic peptide', *Journal of virology* **55**(3), 836–839.

Espejo, P. G., Ventura, S. & Herrera, F. (2009), 'A survey on the application of genetic programming to classification', *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **40**(2), 121–144.

Fuchs, M. (2020), 'The Data Science Process (CRISP-DM)'.
URL: *https://michael-fuchs-python.netlify.app/2020/08/21/the-data-science-process-crisp-dm/*

Galanis, K. A., Nastou, K. C., Papandreou, N. C., Petichakis, G. N., Pigis, D. G. & Iconomidou, V. A. (2021), 'Linear b-cell epitope prediction for in silico vaccine design: A performance review of methods available via command-line interface', *International journal of molecular sciences* **22**(6), 3210.

GitLab (2023), 'What is version control?'.
URL: *https://about.gitlab.com/topics/version-control/.*

gplearn (2023), 'Symbolic Regressor'.
URL: *https://gplearn.readthedocs.io/en/stable/reference.html*

Heymann, S., Latapy, M. & Magnien, C. (2012), Outskewer: Using skewness to spot outliers in samples and time series, *in* '2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining', IEEE, pp. 527–534.

Jespersen, M. C., Peters, B., Nielsen, M. & Marcatili, P. (2017), 'Bepipred-2.0: improving sequence-based b-cell epitope prediction using conformational epitopes', *Nucleic acids research* **45**(W1), W24–W29.

Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J. & Kalantidis, Y. (2019), 'Decoupling representation and classifier for long-tailed recognition', *arXiv preprint arXiv:1910.09217* .

Karplus, P. & Schulz, G. (1985), 'Prediction of chain flexibility in proteins: a tool for the selection of peptide antigens', *Naturwissenschaften* **72**(4), 212–213.

Kolaskar, A. S. & Tongaonkar, P. C. (1990), 'A semi-empirical method for prediction of antigenic determinants on protein antigens', *FEBS letters* **276**(1-2), 172–174.

Kumar, A., Sinha, N., Bhardwaj, A. & Goel, S. (2022), 'Clinical risk assessment of chronic kidney disease patients using genetic programming', *Computer Methods in Biomechanics and Biomedical Engineering* **25**(8), 887–895.

Larsen, J. E. P., Lund, O. & Nielsen, M. (2006), 'Improved method for predicting linear b-cell epitopes', *Immunome research* **2**(1), 1–7.

Morris, G. E. (1996), *Epitope mapping protocols*, Vol. 66, Springer.

Nag, K. & Pal, N. R. (2019), 'Genetic programming for classification and feature selection', *Evolutionary and swarm intelligence algorithms* pp. 119–141.

Odorico, M. & Pellequer, J.-L. (2003), 'Bepitope: predicting the location of continuous epitopes and patterns in proteins', *Journal of Molecular Recognition* **16**(1), 20–22.

Parker, J., Guo, D. & Hodges, R. (1986), 'New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correlation of predicted surface residues with antigenicity and x-ray-derived accessible sites', *Biochemistry* **25**(19), 5425–5432.

Parveen, A. N., Inbarani, H. H. & Kumar, E. S. (2012), Performance analysis of unsupervised feature selection methods, *in* '2012 international conference on computing, communication and applications', IEEE, pp. 1–7.

Paul, W. E. (2012), *Fundamental immunology*, Lippincott Williams & Wilkins.

Pellequer, J. & Westhof, E. (1993), 'Preditop: a program for antigenicity prediction', *Journal of molecular graphics* **11**(3), 204–210.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J. & Fergus, R. (2019), 'Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. biorxiv'.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J. et al. (2021), 'Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences', *Proceedings of the National Academy of Sciences* **118**(15), e2016239118.

Saha, S. & Raghava, G. P. S. (2004), Bcepred: prediction of continuous b-cell epitopes in antigenic sequences using physico-chemical properties, *in* 'International conference on artificial immune systems', Springer, pp. 197–204.

Saha, S. & Raghava, G. P. S. (2006), 'Prediction of continuous b-cell epitopes in an antigen using recurrent neural network', *Proteins: Structure, Function, and Bioinformatics* **65**(1), 40–48.

Sanchez-Trincado, J. L., Gomez-Perosanz, M. & Reche, P. A. (2017), 'Fundamentals and methods for T- and b-cell epitope prediction', *J. Immunol. Res.* **2017**, 2680160.

Santoso, L. W., Singh, B., Rajest, S. S., Regin, R. & Kadhim, K. H. (2021), 'A genetic programming approach to binary classification problem', *EAI Endorsed Transactions on Energy Web* **8**(31), e11–e11.

Saravanan, V. & Gautham, N. (2015), 'Harnessing computational biology for exact linear b-cell epitope prediction: a novel amino acid composition-based feature descriptor', *Omics: a journal of integrative biology* **19**(10), 648–658.

Sela-Culang, I., Benhnia, M. R.-E.-I., Matho, M. H., Kaever, T., Maybeno, M., Schlossman, A., Nimrod, G., Li, S., Xiang, Y., Zajonc, D. et al. (2014), 'Using a combined computational-experimental approach to predict antibody-specific b cell epitopes', *Structure* **22**(4), 646–657.

Sette, A. & Fikes, J. (2003), 'Epitope-based vaccines: an update on epitope identification, vaccine design and delivery', *Current opinion in immunology* **15**(4), 461–470.

Sher, G., Zhi, D. & Zhang, S. (2017), 'Drrep: deep ridge regressed epitope predictor', *BMC genomics* **18**(6), 55–65.

Shuljak, B. (2006), 'Lentiviruses in ungulates. i. general features, history and prevalence', *Bulgarian Journal of Veterinary Medicine* **9**(3), 175–181.

Singh, H., Ansari, H. R. & Raghava, G. P. (2013), 'Improved method for linear b-cell epitope prediction using antigen's primary sequence', *PloS one* **8**(5), e62216.

sklearn (2023), 'sklearn.preprocessing.MinMaxScaler'.
URL: *https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html*

The innate and adaptive immune systems (2020), Institute for Quality and Efficiency in Health Care (IQWiG).

UniProt: the universal protein knowledgebase in 2021 (2021), *Nucleic acids research* **49**(D1), D480–D489.

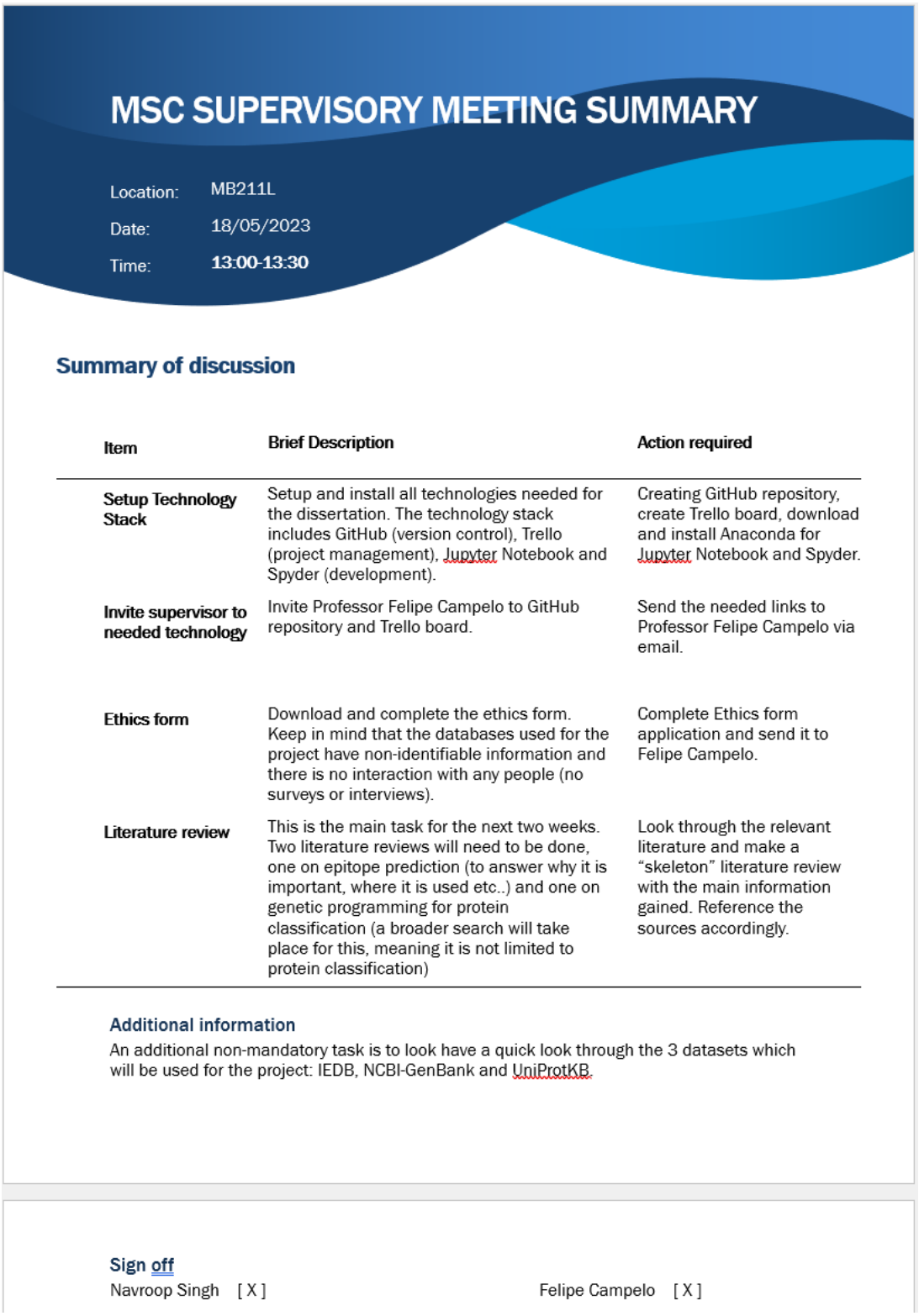
Vita, R., Mahajan, S., Overton, J. A., Dhanda, S. K., Martini, S., Cantrell, J. R., Wheeler, D. K., Sette, A. & Peters, B. (2019), 'The immune epitope database (iedb): 2018 update', *Nucleic acids research* **47**(D1), D339–D343.

Yang, X. & Yu, X. (2009), 'An introduction to epitope prediction methods and software', *Reviews in medical virology* **19**(2), 77–96.

Zhou, J., Chen, J., Peng, Y., Xie, Y. & Xiao, Y. (2022), 'A promising tool in serological diagnosis: Current research progress of antigenic epitopes in infectious diseases', *Pathogens* **11**(10), 1095.

Appendices

A Appendix A - Project Management



MSC SUPERVISORY MEETING SUMMARY

Location: MB211L

Date: 18/05/2023

Time: 13:00-13:30

Summary of discussion

Item	Brief Description	Action required
Setup Technology Stack	Setup and install all technologies needed for the dissertation. The technology stack includes GitHub (version control), Trello (project management), Jupyter Notebook and Spyder (development).	Creating GitHub repository, create Trello board, download and install Anaconda for Jupyter Notebook and Spyder.
Invite supervisor to needed technology	Invite Professor Felipe Campelo to GitHub repository and Trello board.	Send the needed links to Professor Felipe Campelo via email.
Ethics form	Download and complete the ethics form. Keep in mind that the databases used for the project have non-identifiable information and there is no interaction with any people (no surveys or interviews).	Complete Ethics form application and send it to Felipe Campelo.
Literature review	This is the main task for the next two weeks. Two literature reviews will need to be done, one on epitope prediction (to answer why it is important, where it is used etc..) and one on genetic programming for protein classification (a broader search will take place for this, meaning it is not limited to protein classification)	Look through the relevant literature and make a "skeleton" literature review with the main information gained. Reference the sources accordingly.

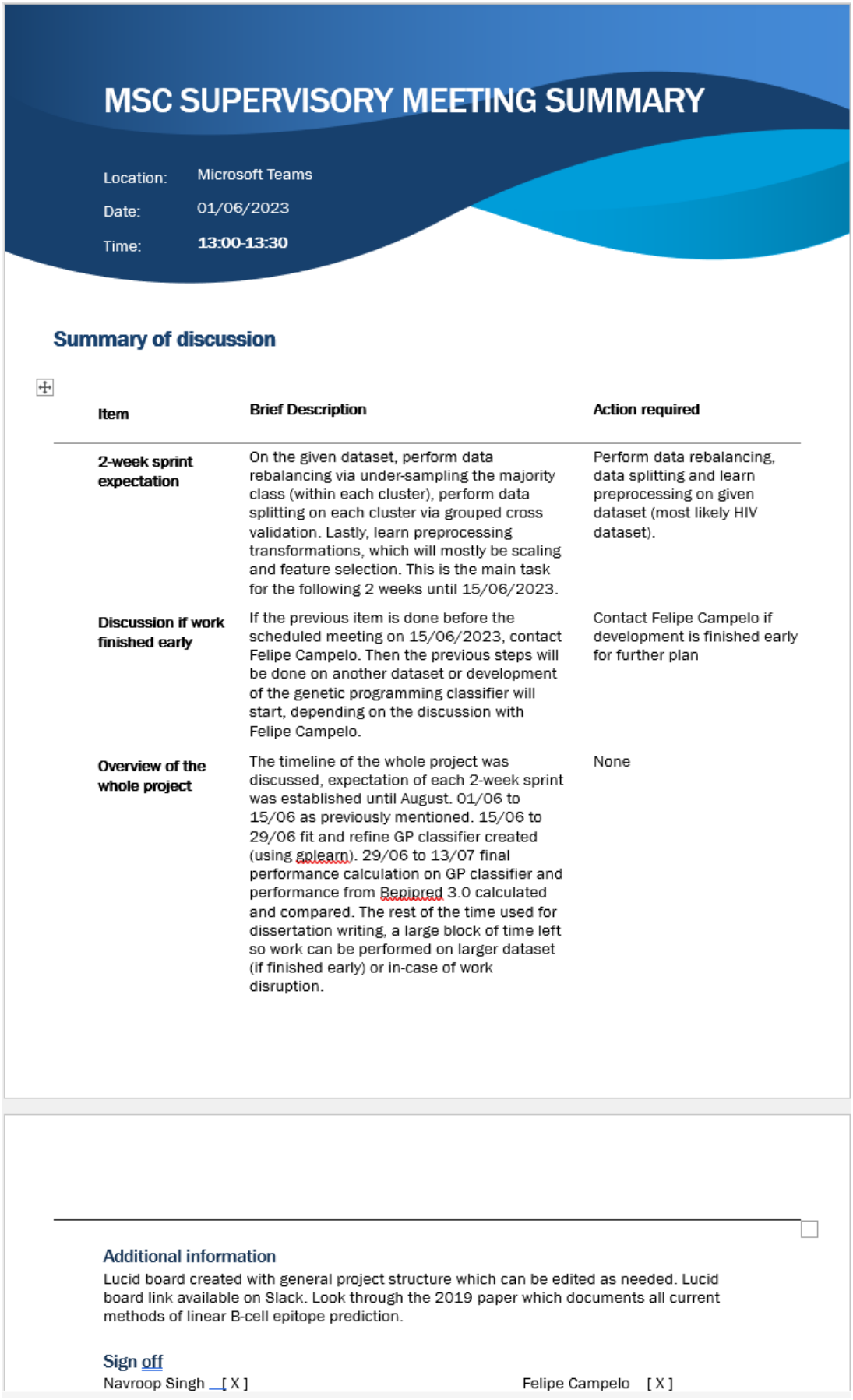
Additional information

An additional non-mandatory task is to look have a quick look through the 3 datasets which will be used for the project: IEDB, NCBI-GenBank and UniProtKB.

Sign off

Navroop Singh [X] Felipe Campelo [X]

Figure 18: Meeting notes 1



MSC SUPERVISORY MEETING SUMMARY

Location: Microsoft Teams

Date: 01/06/2023

Time: 13:00-13:30

Summary of discussion

Item	Brief Description	Action required
2-week sprint expectation	On the given dataset, perform data rebalancing via under-sampling the majority class (within each cluster), perform data splitting on each cluster via grouped cross validation. Lastly, learn preprocessing transformations, which will mostly be scaling and feature selection. This is the main task for the following 2 weeks until 15/06/2023.	Perform data rebalancing, data splitting and learn preprocessing on given dataset (most likely HIV dataset).
Discussion if work finished early	If the previous item is done before the scheduled meeting on 15/06/2023, contact Felipe Campelo. Then the previous steps will be done on another dataset or development of the genetic programming classifier will start, depending on the discussion with Felipe Campelo.	Contact Felipe Campelo if development is finished early for further plan
Overview of the whole project	The timeline of the whole project was discussed, expectation of each 2-week sprint was established until August. 01/06 to 15/06 as previously mentioned. 15/06 to 29/06 fit and refine GP classifier created (using gplearn). 29/06 to 13/07 final performance calculation on GP classifier and performance from Bepipred 3.0 calculated and compared. The rest of the time used for dissertation writing, a large block of time left so work can be performed on larger dataset (if finished early) or in-case of work disruption.	None

Additional information

Lucid board created with general project structure which can be edited as needed. Lucid board link available on Slack. Look through the 2019 paper which documents all current methods of linear B-cell epitope prediction.

Sign off

Navroop Singh [X]

Felipe Campelo [X]

Figure 19: Meeting notes 2

MSC SUPERVISORY MEETING SUMMARY

Location: Microsoft Teams

Date: 16/06/2023

Time: **13:00-13:30**

Summary of discussion

Item	Brief Description	Action required
Learn preprocessing transformations	Perform feature selection using the PCA method and scale data. Encapsulate the preprocessing transformations into a preprocessing-only pipeline.	Feature selection using PCA, data scaling, make preprocessing-only pipeline.
Explore genetic programming models	Look through GP-learn and understand how it works, in order to understand what parameters to add and understand the limitations and potential of it. Do the same with other genetic programming python packages.	Explore and understand GP-learn and other genetic programming python packages.
Expected timeline	Expectations were set on how much work to be done until 27/07 (sticky notes on lucid board).	

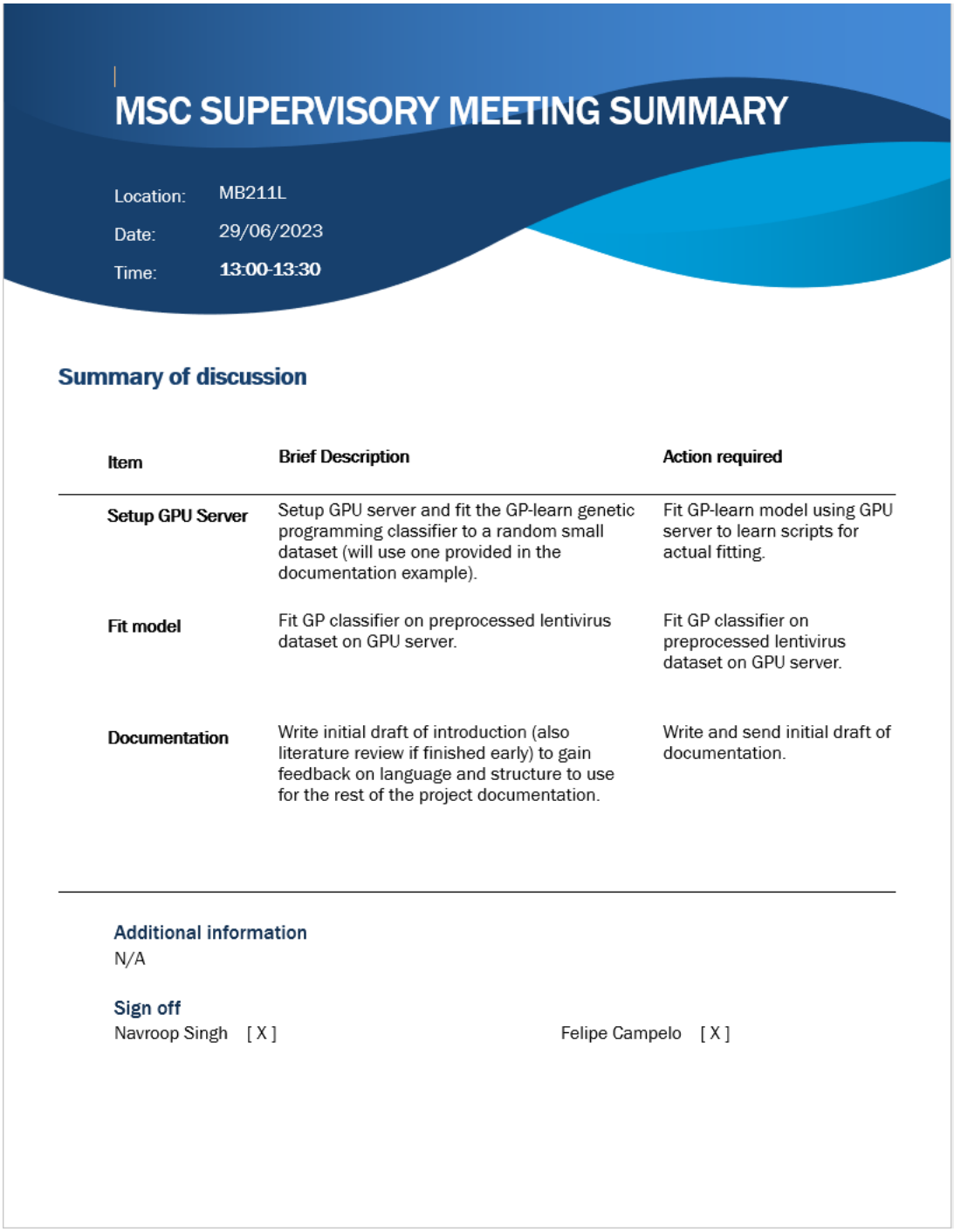
Additional information

Next meeting (29/06), bring laptop to in-person meeting to review code so far.

Sign off

Navroop Singh [X] Felipe Campelo [X]

Figure 20: Meeting notes 3



MSC SUPERVISORY MEETING SUMMARY

Location: MB211L

Date: 29/06/2023

Time: **13:00-13:30**

Summary of discussion

Item	Brief Description	Action required
Setup GPU Server	Setup GPU server and fit the GP-learn genetic programming classifier to a random small dataset (will use one provided in the documentation example).	Fit GP-learn model using GPU server to learn scripts for actual fitting.
Fit model	Fit GP classifier on preprocessed lentivirus dataset on GPU server.	Fit GP classifier on preprocessed lentivirus dataset on GPU server.
Documentation	Write initial draft of introduction (also literature review if finished early) to gain feedback on language and structure to use for the rest of the project documentation.	Write and send initial draft of documentation.

Additional information

N/A

Sign off

Navroop Singh [X]

Felipe Campelo [X]

Figure 21: Meeting notes 4

MSC SUPERVISORY MEETING SUMMARY

Location: Microsoft Teams

Date: 14/07/2023

Time: **10:00-10:30**

Summary of discussion

Item	Brief Description	Action required
Comparison	Discussed future work on comparison with gplearn and BeciPred. This will be further elaborated on the next meeting (20-07).	N/A
Sending work	The Jupyter notebook and first draft of introduction needs to be sent to Felipe by 17-07 for review.	Email Felipe the notebook and first draft.
Hyperparameter tuning	Finish hyperparameter tuning of gplearn (preferably by 17-07).	Finish hyperparameter tuning of gplearn.

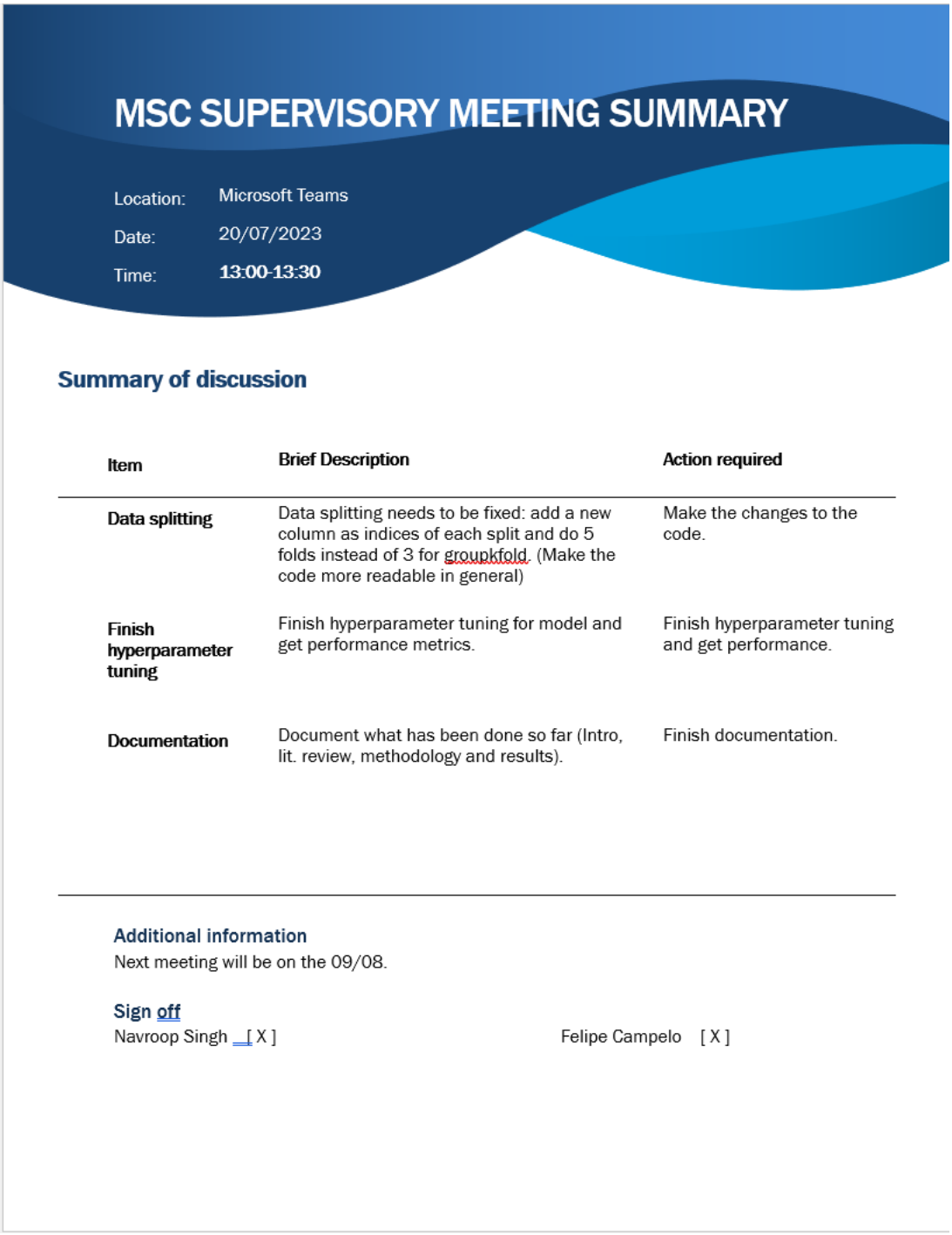
Additional information

N/A

Sign off

Navroop Singh [X] Felipe Campelo [X]

Figure 22: Meeting notes 5



MSC SUPERVISORY MEETING SUMMARY

Location: Microsoft Teams

Date: 20/07/2023

Time: **13:00-13:30**

Summary of discussion

Item	Brief Description	Action required
Data splitting	Data splitting needs to be fixed: add a new column as indices of each split and do 5 folds instead of 3 for groupkfold. (Make the code more readable in general)	Make the changes to the code.
Finish hyperparameter tuning	Finish hyperparameter tuning for model and get performance metrics.	Finish hyperparameter tuning and get performance.
Documentation	Document what has been done so far (Intro, lit. review, methodology and results).	Finish documentation.

Additional information

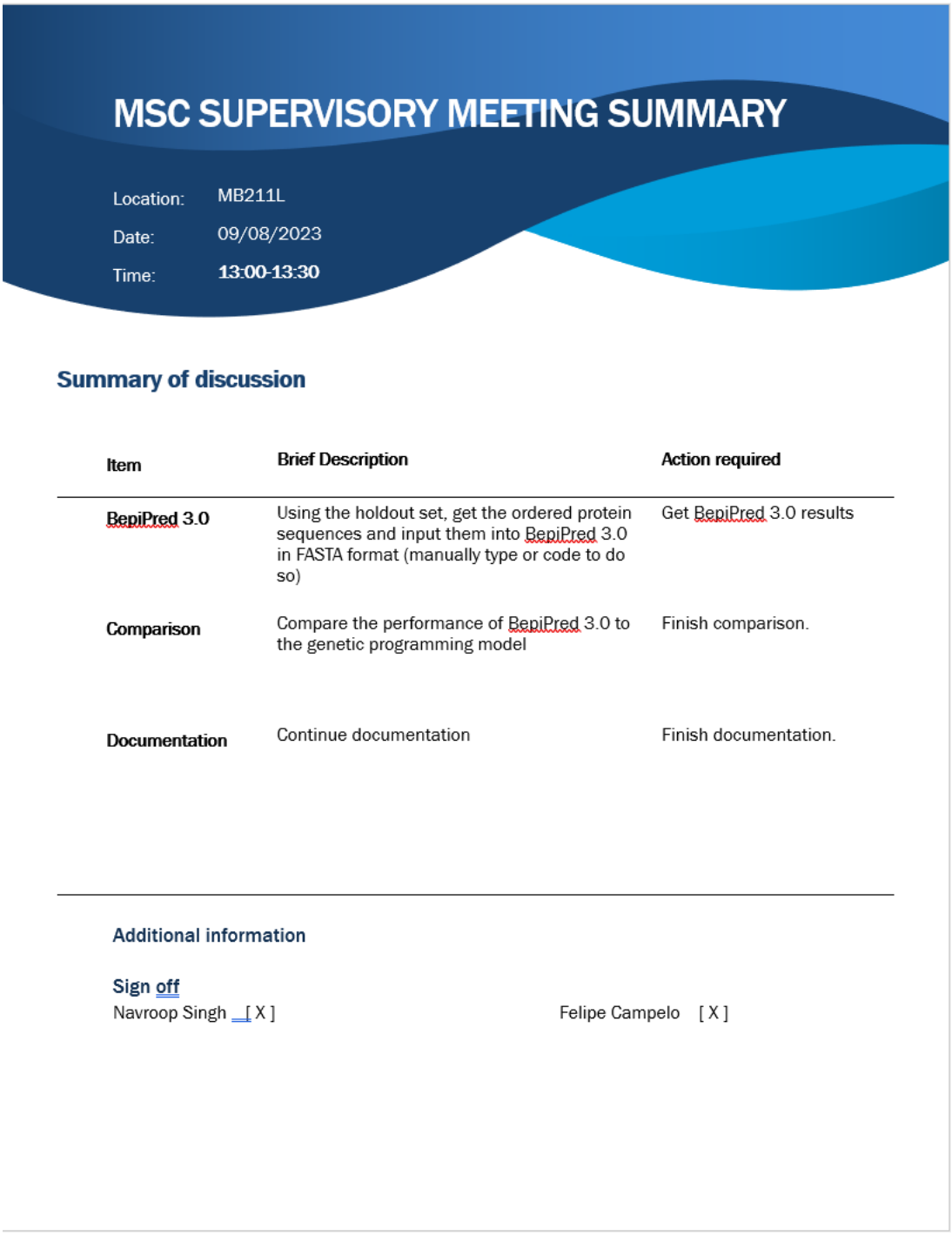
Next meeting will be on the 09/08.

Sign off

Navroop Singh [X]

Felipe Campelo [X]

Figure 23: Meeting notes 6



MSC SUPERVISORY MEETING SUMMARY

Location: MB211L

Date: 09/08/2023

Time: 13:00-13:30

Summary of discussion

Item	Brief Description	Action required
BepiPred 3.0	Using the holdout set, get the ordered protein sequences and input them into BepiPred 3.0 in FASTA format (manually type or code to do so)	Get BepiPred 3.0 results
Comparison	Compare the performance of BepiPred 3.0 to the genetic programming model	Finish comparison.
Documentation	Continue documentation	Finish documentation.

Additional information

Sign off

Navroop Singh [X]

Felipe Campelo [X]

Figure 24: Meeting notes 7

MSC SUPERVISORY MEETING SUMMARY

Location: Microsoft Teams

Date: 12/09/2023

Time: **09:00-09:30**

Summary of discussion

Item	Brief Description	Action required
Dissertation	Finish the dissertation	Finish disseration

Additional information

Sign off

Navroop Singh [X]

Felipe Campelo [X]

Figure 25: Meeting notes 8

B Appendix B - Ethics Approval

Aston University

Ethics Declaration Form for CS4700/CS4705 Student Dissertation Projects

EPS Department: Computer Science

Project Title: Evolutionary Classification Systems for Epitope Prediction

Supervisor Name and e-Mail: Felipe Campelo, f.campelo@aston.ac.uk

Student Name, Number and e-Mail: Navroop Singh, 190114871, 190114871@aston.ac.uk

Section 0: Ethics Declaration Questions

If you are conducting a study for your supervisor where your supervisor already has ethical approval for the work to be conducted, please consult with your supervisor to ensure that (s)he has applied for an amendment to the existing ethical approval in order to permit your involvement. A letter from the College Research Ethics Committee confirming that permission will need included with your project report. You do not need to complete the remainder of this form.

Please answer the following questions honestly and carefully:

Will your project involve any of the following?	*Delete as applicable*
The NHS – either patients selected via the NHS or clinical staff working for the NHS	Not Permitted
Participants under the age of 18	Not Permitted
Vulnerable groups – individuals with physical disabilities, mental health disabilities/ill health, individuals with learning difficulties, prisoners/detained persons, people over 65 years of age, and/or pregnant women	Not Permitted
Any of the following: i) clinical procedures or ii) physical intervention or iii) penetration of the participant's body or iv) prescription of compounds additional to normal diet or other dietary manipulation/supplementation or v) collection of bodily secretions or vi) involve human tissue which comes within the Human Tissue Act (e.g., surgical operations; taking body samples including blood and DNA; exposure to ionizing or other radiation; exposure to sound light or radio waves; psychophysiological procedures such as fMRI, MEG, TMS, EEG, ECG, exercise and stress procedures; administration of any chemical substances)	Not Permitted
Delving into topics that might be sensitive, embarrassing or upsetting or where it is possible that criminal or other disclosures requiring action could take place (e.g., during interviews) – including but not limited to projects focusing on mental health	Not Permitted
Human participants *(including all types of interviews, questionnaires, focus groups, records relating to humans, use of online datasets or other secondary data, observations, usability testing, etc.)*	No
Testing of apparatus *(including where you have developed new apparatus and are testing it for accuracy, including on yourself)*	No
Risk to you, including:	No

Page 1 Form Version 1: 01.12.2022

Figure 26: Ethical Declaration Form Page 1

lone working during data collection	No
travel to areas where you may be at risk	No
risk of emotional distress	No
other: *please outline*	No
Any risk to the environment	No
Any conflict of interest	No
Work/research that could be considered controversial or be of reputational risk to Aston University	No
Social media data and/or data from internet sources that could be regarded as private	No
Any other ethical considerations *No further ethical considerations. The datasets to be used are based on publicly available data sources maintained by the National Institutes of Health/USA, containing data contributed by scientists worldwide derived from laboratory assays (in vitro, in vivo and ex vivo) to confirm immunogenicity of proteins or peptides. The data does not contain any personal information and is available for use without restriction. The NCBI statement reads: "Databases of molecular data on the NCBI Web site include such examples as nucleotide sequences (GenBank), protein sequences, macromolecular structures, molecular variation, gene expression, and mapping data. They are designed to provide and encourage access within the scientific community to sources of current and comprehensive information. Therefore,* ***NCBI itself places no restrictions on the use or distribution of the data contained therein. Nor do we accept data when the submitter has requested restrictions on reuse or redistribution****."* [https://www.ncbi.nlm.nih.gov/home/about/policies/#data]	No

If you answer 'yes' to any of the questions above, you will need to continue to complete the rest of this form as well as any required participant-facing documentation (see Ethics Guidance provided). You and your supervisor will need to sign the declarations section at the end of this form before submitting it, along with all other paperwork, for ethical approval. No data collection and use (including recruitment of participants) may take place before ethical approval has been granted. All ethics paperwork, including evidence of ethical approval, should be included in your final report submission.

If you answered 'no' to all of the questions above, you may skip to Section 2. You and your supervisor will need to sign the declarations section at the end of this form. The form will need included in your final report submission.

Section 1: Study Details

Please provide the following information about your study. Be as detailed as possible. Where a question is not relevant, please indicate 'Not Applicable' but also explain why you believe that to be so.

Study Details	
Project Objectives *Please provide a brief outline of your overall project objectives*	
Study Objectives *Please explain how the study you are seeking ethical approval to conduct contributes to your overall project objectives*	

Figure 27: Ethical Declaration Form Page 2

Data Collection Method(s) to be Used *Please outline your proposed data collection methods – e.g., questionnaire/survey, interview, observational study, etc. Justify their use and explain how you will conduct the data collection in practice, including timeframe*	
Data to be Collected *Please briefly outline the type of data to be collected*	
Location of Data Collection *Please briefly outline where you will be collecting the data – e.g., where you will be conducting your study.*	
Participant Recruitment *Please outline how you will recruit participants to your study and how you will ensure that participants are not coerced to participate.*	
Data Storage *Please outline where you will store your data (ideally, on an encrypted server; USB drives are not permissible)*	
Data Deletion *Please outline when you will erase the data you collected. For personally identifiable information, it should be deleted once an anonymised version is created, e.g., audio recordings should be deleted once the corresponding anonymised transcripts are created. For anonymised data, it should be kept until the final grade of the dissertation is released OR there is agreement from participants (via the consenting process) to retain and share data for/with future research(ers). Please provide justification if you have no plan to delete the collected data.*	
Data Access *Please indicate who will have access to the data you collected - e.g., your supervisor and any other members of the research team.*	

Figure 28: Ethical Declaration Form Page 3

For Secondary Data/Dataset Use Only: Compliance with Terms & Conditions of Use *If you will be in receipt of secondary data OR will be using an online, publicly available dataset, please provide evidence that you are observing any terms and conditions associated with its use and have permission to use it. Be mindful that just because data is available online does not mean that you are ethically entitled to use it for your study; this needs proven. If you are being given data by, for example, a third party, you need to be sure that the individual has permission to share the data with you.*	
Risk *Please outline any risks to either the participants in your study and/or yourself in the conduct of the study and what you have done to mitigate that risk*	

Section 2: Declarations

The following declaration should be signed by both you, the student, and your supervisor
Student: I confirm the following: • The above is an accurate representation of my study activities; • That I shall not commence participant recruitment and/or data collection without ethical approval to do so (where applicable); • That I shall seek further ethical approval should I need to make any changes to my study after ethical approval has been granted (where applicable); • That I shall conduct my study with integrity and in accordance with the ethical approval granted (where applicable); • That, where necessary, I shall use existing or secondary data in accordance with terms and conditions of use or ethical approval, as applicable; • That I understand that if I breach the terms of the approval granted I may not be able to use the data collected in my project report and may face disciplinary procedures; and • That I shall respect my participants (where applicable) and the data I have collected and am using. Supervisor: I confirm the following: • That I have reviewed the content of this form and all associated paperwork and am happy with its standard and accuracy; • That I shall monitor the student's conduct of the study in accordance with the ethical approval granted (where applicable); and • That I shall report to the person(s) granting ethical approval any breaches of approval and ensure that no data is included in the student's work that has been collected in breach of approval.

Figure 29: Ethical Declaration Form Page 4

Both student and supervisor should sign* and date below:	
Signatures: Navroop Singh Felipe Campelo Digitally signed by Felipe Campelo Date: 2023.06.06 18:05:02 +01'00' * note, typed/e-signatures are acceptable.	Date: 31/05/2023

Page 5 Form Version 1: 01.12.2022

Figure 30: Ethical Declaration Form Page 5

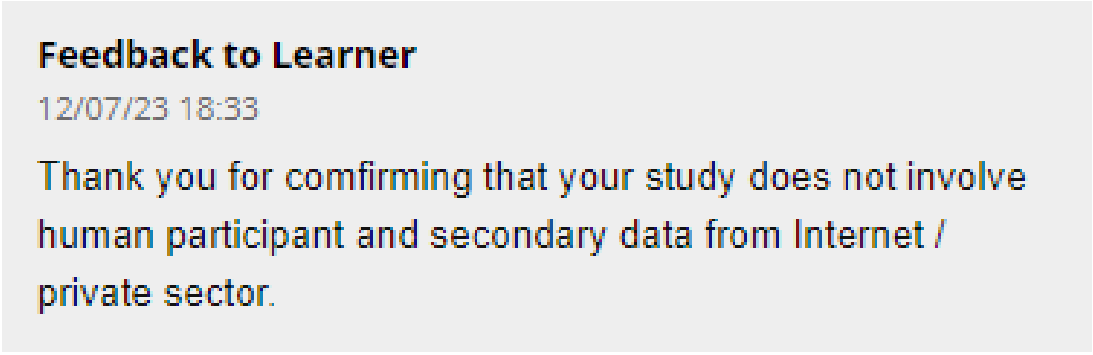


Figure 31: Ethical Approval

C Appendix C - Performance Metrics

Performance indicators

Several performance indicators were calculated to provide comparability with different references in the literature, and to explore distinct aspects of the predictive behaviour of the models. In the definitions below, TP, TN, FP and FN denote the counts of True Positives, True Negatives, False Positives and False Negatives, respectively.

- **Positive Predictive Value**,

$$PPV = \frac{TP}{TP + FP}.$$

  Quantifies the probability of a peptide actually representing an epitope, given that it was classified as *positive*. This measure provides an indication of how trustworthy the positive predictions of each model are, which is relevant for justifying the application of resources to the experimental validation of the predicted targets. PPV is also referred to as *Precision*.

- **Negative Predictive Value**,

$$NPV = \frac{TN}{TN + FN}.$$

  Quantifies the probability of a peptide not representing an epitope, given that it was classified as *negative*. This measure provides an indication of how assertive the model is when not recommending a peptide, i.e., how likely it is not to miss potentially valuable targets.

- **Sensitivity**,

$$SENS = \frac{TP}{TP + FN}.$$

  Measures the probability of correctly identifying an epitope (positive class) from the data. Sensitivity is also known as the **True Positive Rate** (TPR).

Figure 32: Performance Metrics 1 (Ashford et al. 2021)

- **Accuracy**,

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}.$$

Quantifies the overall rate of correct classifications. Although it represents a global performance index, accuracy is sensitive to class imbalance and implicitly assumes equal misclassification costs for false positives and false negatives, which is not necessarily the case for epitope prediction.

- **Matthews Correlation Coefficient**,

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

A combined performance index that "generates a high score only if the binary predictor was able to correctly predict the majority of positive data instances and the majority of negative data instances" (Chicco and Jurman, 2020). The MCC provides a unified metric for the comparison of the overall performance of the predictors.

- **Area under the ROC curve** (AUC), which quantifies the robustness of the trade-off between the Sensitivity and the False Positive Rate (FPR),

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

of a classification model, under different classification thresholds.

Figure 33: Performance Metrics 2 (Ashford et al. 2021)

D Appendix D - How to run the project

The project is available at: https://github.com/Nav59/Dissertation.git

To run the project Jupyter Notebook is required, the project can then be uploaded to Jupyter Notebook and can be run from it. To replicate the project. The dataset named:

all_virus_LBCE_with_ESM1b_features.csv

must be downloaded from the following link:

https://aston.app.box.com/s/iv0bokeuusr2mu3ql8u4n2t3kj8twze0