Norges miljø- og
biovitenskapelige
universitet

Master's Thesis 2022 60 ECTS

Faculty of Chemistry, Biotechnology and Food Science (KBM)

Benchmarking sequence-based machine learning methods for prediction of antibody polyreactivity

Ingvild Frøberg Mathisen

MSc Bioinformatics and Applied Statistics - Specialisation: Bioinformatics

1 Acknowledgement

This thesis is part of a Master's study in Bioinformatics at the Norwegian University of Life Sciences (NMBU) and the work that is presented was carried out at the Department of Immunology, Rikshospitalet, Oslo from August 2021 until August 2022.

First I would like to thank my supervisor Dr. Philippe Robert. My sincerest gratitude for all the advice and support you have given me. Your advice on planning and carrying out research for my thesis has been invaluable. Mercí for sharing your wisdom with me and teaching me much about research.

I would also like to thank my co-supervisor and leader of my lab Prof. Victor Greiff for first and foremost allowing me to be a part of his research group and conduct my thesis there, and also for advice he has provided underway. And I would like to thank Dr. Rahmad Akbar for providing valuable advice and feedback.

I also want to express my gratitude for the help provided by my internal supervisor Prof. Lars Gustav Snipen. My sincerest gratitude for advice and help with the administrative work related to the thesis. And thank you for all you have taught me about bioinformatics through the years.

Thanks to all the members of GreiffLab for their support and for being both welcoming and supportive. It has been a pleasure to work with you. In particular, I would like to thank Robert Frank for introducing integrated gradients and thank you to Maria Chernogovskaya, Puneet Rawat and Eva Smorodina for their advice.

Lastly, I would like to thank my family for supporting me throughout my masters and previous studies. My thanks are also extended to my dog Sjakka. Even though she will never read this, I want to express my appreciation for 12+ years together. You have brought me much joy over the years and been a truly enriching presence in my life.

Oslo, July 2022
Ingvild Frøberg Mathisen

2 Summary

Antibodies are of great importance as therapeutics and a prerequisite for their success is high specificity to a desired target. Some antibodies however, termed polyreactive, are able to recognize and react to a diverse range of antigens. Therapeutic antibodies have been successful in the treatment of cancers, autoimmune conditions as well as infectious diseases. The process of developing a new treatment requires a lot of resources and time. Many candidates are not approved following clinical trials, one of the negative indicators of success is polyreactivity. *In silico* methods that predict polyreactivity can aid in prioritizing good candidates for further development. In this work we benchmarked machine learning approaches for predicting polyreactivity, using data from the simulation framework `Absolut!`, with various combinations of sequence encodings and machine learning architectures. We found that polyreactivity could be predicted with close to 90% macro f1 using logistic-regression and amino acid composition. Marginal but significant improvements in macro f1 score were obtained by inclusion of positional information using logistic regression and feed-forward neural network with one hidden layer was able to achieve macro f1 of ~93%. The best logistic regression model and neural network were able to generalize to sequences that were more than 50% different to the sequences used for training the model. Further we looked into interpretability of the models and size of possible sequence motifs, suggesting that short motifs can be predictive of polyreactivity. Our results demonstrate how different approaches compare in predicting polyreactivity given large amounts of data and information on the CDRH3 sequence. As of now, experimental (*in vitro*) datasets containing information on polyreactivity have been limited in size. Larger datasets are being produced and in the future, we anticipate that our findings will be relevant for guiding the development of machine learning models for predicting polyreactivity.

3 Sammendrag

Antistoffer er av stor betydning som legemidler og en forutsetning for at de skal lykkes er høy spesifisitet til et ønsket mål. Noen antistoffer, kalt polyreaktive antistoffer, er imidlertid i stand til å gjenkjenne og reagere på flere ulike antigener. Terapeutiske antistoffer har hatt stor suksess i behandling av kreft, autoimmune sykdommer, så vel som infeksjonssykdommer. Prosessen med å utvikle en ny antistoffbehandling krever imidlertid mye ressurser og tid. Mange kandidater blir ikke godkjent etter kliniske studier og polyreaktivitet kan være en av årsakene til dette. *In silico* metoder som predikerer polyreaktivitet kan hjelpe til med å prioritere kandidater med større sannsynlighet for å lykkes. I dette arbeidet har vi sammenlignet maskinlærings metoder med varierende måter å fremstille sekvensdata og arkitektur for å forutsi polyreaktivitet ved hjelp av data fra en antistoff-antigen bindings simulator kalt `Absolut!`. Vi fant at polyreaktivitet kunne forutsies med nær 90% makro f1 score ved bruk av logistisk regresjon og aminosyresammensetning alene. Marginal men signifikant økning i f1 score ble registret når informasjon om aminosyrenes posisjon var inkludert ved bruk av logistisk regresjon. Nevrale nettverk var i stand til å oppnå makro f1 på ~93%. Den logistiske regresjonsmodellen og det beste nevrale nettverket var i stand til å generalisere til sekvenser som var mer enn 50 % forskjellige fra sekvensene som ble brukt for trening av modellen. Videre så vi på tolkbarhet av modellene og størrelsen på eventuelle sekvens motiver, og fant at korte motiver kan være prediktive for polyreaktivitet. Resultatene våre viser hvordan forskjellige tilnærminger sammenlignes for å forutsi polyreaktivitet gitt store mengder data og informasjon om CDRH3-sekvensen. Tidligere har de fleste eksperimentelle (*in vitro*) datasett brukt til å etterforske polyreaktivitet vært av begrenset størrelse. Det produseres nå større datasett og i fremtiden regner vi med at funnene våre vil være relevante for styre utvikling av maskinlæringsmodeller for å forutsi polyreaktivitet.

4 Abbreviations

AA = amino acid

ABS = antigen binding site

ADCC = antibody dependent self mediated toxicity

BCR = B-cell receptor

CDC = complement dependent cytotoxicity

CDR = complementarity determining region

CDRH3 = 3rd CDR on the antibody heavy chain

CNN = convolutional neural network

DL = data leakage

DT = decision tree

FNN = feed-forward neural network

LD = levenshtein distance

LR = logistic regression

ML = machine learning

RF = random forest

SHM = somatic hypermutation

UMAP = Uniform Manifold Approximation and Projection

PDB = Protein Data Bank

5 Introduction

Antibodies have become an increasingly important class of therapeutics. Therapeutic antibodies are used to treat several types of illnesses such as cancer, autoimmune disease, and increasingly infectious diseases (Pecetta, Finco, and Seubert 2020). In 2018, the value of the global antibody market was about 115.2 billion US dollars (Lu et al. 2020) and 7 out of the top 10 most sold drugs worldwide were antibodies (Urquhart 2019). It is estimated that the value of the antibody market will rise to 300 billion US dollars by 2025. However, the elimination (attrition) of candidate antibodies along discovery pipelines requires a lot of time and money. In 2017, the cost of preclinical development for therapeutic antibodies was more than 1 billion dollars (Elgundi et al. 2017). Not all candidates that advance to clinical trials are successfully approved for use. Specifically, the rate of success for antibody therapies from phase 1 trials to approval has been reported to lie between 17% and 25% (Kaplon and Reichert 2018). To speed up the development of novel antibody therapeutics, the design and prioritization of candidates with desirable properties can increase the chance of successful approval following clinical trials.

In silico approaches to antibody design and selection have been proposed to reduce the time and cost of development and increase the chance of successful approval (Akbar et al. 2022; Norman et al. 2020). Traditional low-throughput computational methods are already used as aids for this purpose (Norman et al. 2020). The development of computationally less expensive and high-throughput machine-learning approaches has the potential to significantly reduce expenditures and improve the time it takes for drugs to become available for patients.

Success of antibodies as drugs can be attributed to their desirable druglike properties, such as stability and solubility, favorable pharmacokinetics, high affinity etc. as well as high specificity (Starr and Tessier 2019). Antibodies are often assumed to be highly specific with minimal non-target interactions. Some antibodies however exhibit broad cross-reactive binding to structurally different antigens, a phenomena called polyreactivity. Polyreactivity in therapeutic antibodies is a negative indicator for success through clinical trials (Cunningham et al. 2021), in particular due to off-target binding.

Despite pioneering works identifying predictors of polyreactivity and modeling (Rabia et al. 2018; Boughter et al. 2020), few studies have attempted to predict antibody polyreactivity. Thus no such tool has yet been used in real-world antibody development. As polyreactivity is a hindrance to drug approval, it is of current interest to develop and benchmark better machine learning (ML) methods.

Most polyreactivity datasets however are small and likely insufficient for reliably benchmarking. To address a general lack of large thoroughly annotated antibody-antigen datasets available for benchmarking, a framework for simulating antibody-antigen binding called `Absolut!` was created (Robert et al. 2022). `Absolut!` is based on a lattice-based framework representing proteins-protein interactions called `Ymir`, (Robert, Arulraj, and Meyer-Hermann 2021), that simulates the binding between antibodies and antigens through exhaustive docking of the antibody sequence onto a 3D grid representation of the antigen. `Absolut!` can be used as an oracle to predict how certain methods will perform when predicting various levels of antibody-antigen binding. In this work, we will be using `Absolut!` data to benchmark methods for predicting polyreactivity.

5.1 Background

5.1.1 Innate and Adaptive Immunity

The cellular and molecular mechanisms of the immune system have historically been separated into two layers, namely innate and adaptive immunity. The innate immune system makes up the first response mechanisms which initially detect an infection and is able to respond rapidly. These responses are initiated within minutes (Kaur and Secord 2019). Innate immunity usually recognizes evolutionarily conserved structures that are characteristic of different classes of pathogens (Gao et al. 2022). This allows it to detect a foreign attack. If the innate immune response is not successful in eliminating the threat, the adaptive immune response (of which B-cells are a part) is initiated.

Adaptive immune system responses typically take a few days to develop the first time a particular pathogen is encountered (Janeway et al. 2001). The antigens from the pathogen must be presented to the lymphocytes (B- and T-cells) by cells from the innate immune system (antigen presenting cells) (Charles Molnar and Gair 2015), for the naive B- and T-cells that recognize the antigens to be activated. A naive B-cell (or T-cell) is a mature lymphocyte that has not yet recognized an antigen. Recognition is based on the B- or T-cell receptor sequence which varies between cells. Then, following activation, these lymphocytes have to go through processes of cell growth, proliferation, production of soluble signals, and (for B-cells) a process of somatic hypermutation (SHM) that specifically mutates the B-cell receptor (BCR) gene (Eisen 2014).

SHM is an important part of the affinity (strength of antigen binding) maturation process. Affinity maturation is a process by which the affinity of the antibody is tuned to the target. Several steps of hypermutation and subsequent selection of high affinity clones which are allowed to survive and expand more rapidly, lead to the production of antibodies with increased affinity to the antigen over time. This process takes place in anatomical structures called germinal centers (Victora and Nussenzweig 2022). Sequence variation and modification through affinity maturation allow the adaptive immune system to mount more specific and customized responses to the pathogen.

5.1.2 Role of Antibodies in Adaptive Immunity

In nature, antibodies are weapons of the immune system against disease-causing agents such as viruses, toxins, and bacteria as well as tumor-transformed cells. Antibodies are 'Y'-shaped glycoproteins produced by B-cells (Parham 2014). When these proteins are bound to the B-cell membrane, they are referred to as B-cell receptors (BCRs), while 'antibodies' refer to the BCRs in their secreted form. Antibodies bind to particular molecular structures (called antigens) either in the extracellular medium, or on the surface of pathogens. After binding to an antigen, antibodies can participate in different modes of immune defense mechanisms. They can neutralize the target's ability to interact with cells in the body (i.e., preventing viral entry) (Klasse 2014). They can also make the target more recognizable to other parts of the immune system, such as tagging a pathogenic cell for destruction by other immune cells (ADCC) or the complement system (CDC) (Ravetch and Bolland 2001; Shakib 2016; Hashimoto, Wright, and Karzon 1983). In this way, antibodies help the body rid itself of specific pathogens by screening the extracellular medium, including the mucosa.

5.1.3 Affinity and Specificity

The properties of antibody-antigen binding can be quantified with different measures, in particular affinity and specificity. Antibody-antigen affinity refers to the strength of binding while specificity refers to the clarity of preference for that antigen over others. An antibody with high affinity binds strongly to a target and forms a stable complex. Binding strength is measured by the dissociation constant (Libretexts 2015), which in the case of antibody-antigen binding describes the tendency of the antibody-antigen complex to dissociate into antibody and antigen. This property is the primary property tuned during affinity maturation. Antibody specificity on the other hand is determined by discriminative affinity differences between encountered antigens. Specificity is a relative measure of the propensity of an antibody to bind to one antigen to the exclusion of others (Starr and Tessier 2019). Affinity describes the strength of binding between the antibody and one antigen, specificity describes the strength of binding to one or a few cognate antigens versus the strength of binding to other antigens.

5.1.4 The Antigen Binding Site

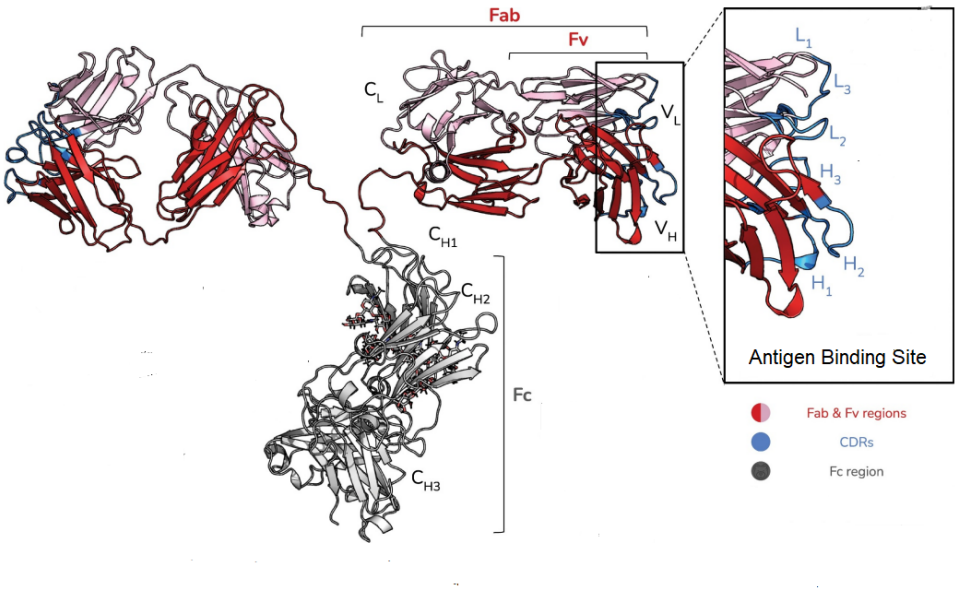


Figure 1**: Structure of an antibody highlighting regions involved in antigen recognition.** Antibodies share a characteristic 'Y'-shape with one Fc region and two Fab regions. The Fc region can interact with other cells of the immune system (it is also the part that is bound to the B-cell in non-secreted form). The antigen-binding sites are located at the ends of the Fab region, defining two identical antigen binding sites per antibody. The Fab of the antibodies is made up of four segments, two heavy (VH) chains and two light (VL) chains. The variable (Fv) regions of the antibody is located farthest away from the Fc region. The Fv region is made up of the variable domains of the heavy and light chains as opposed to the rest of the Fab (and Fc) region which is made up of the constant domains. The Fv region is host to the complementary determining regions (CDRs). There are three CDRs belonging to the light chain (L1, L2 and L3) as well as three belonging to the heavy chain (H1, H2 and H3). (Adapted from Figure 4 of (Akbar et al. 2022), used by consent of Dr. Rahmad Akbar)

The affinity and specificity of an antibody to an antigen is largely determined by the sequence and structure of the antigen binding site (ABS) (black box in Figure 1). The ABS are located at the ends of the Fab regions (Figure 1) and is the structure where the antibody binds antigens. These parts of the antibodies are made up of two segments, one heavy chain and one light chain (Chiu et al. 2019) which defines the ABS. There is one ABS on each Fab region. The part of the ABS that binds to the antigen is called the paratope (the part of the antigen that the paratope binds to is called the epitope). Each ABS contains six hypervariable loops which often dominate the paratope (Akbar et al. 2021), called complementary determining regions (CDRs). The CDRs are typically thought of as the main determinants of antibody-antigen binding as the sequence diversity of the ABS is centered in these loops.

5.1.5 V(D)J recombination gives rise to naive antibody repertoire diversity

Prior to affinity maturation, the sequence variation in the hypervariable loops is determined by a mechanism called somatic V(D)J-recombination (Parham 2014). Amongst somatic cells only lymphocytes (e.g., B-cells and T-cells) have the capacity to rearrange their own genome. The genes coding for the heavy and light chains of the antibodies are segmented and spread out across parts of the chromosome termed the heavy- and light chain loci (one heavy-chain loci and two light-chain loci). Where the loci are the positions of the genes/ gene fragments on the chromosome. For the antibody chain genes to become functional, the segments need to be combined into one continuous gene. The segments encoding the variable regions of the immunoglobulins are classed into three types of segments (The V, D and J segments). The heavy chain genes include one of each type of segment and the light chain genes include one V and one J segment. In mice (*Mus musculus*) there are estimated to be 112 V-genes, 12 D-genes and 4 J-genes included in the locus encoding the variable region of the heavy chain when excluding pseudogenes (Wardemann and Busse 2017). During VDJ-recombination one version of each type of relevant segments is combined to form a complete gene. This variation included in the genes encoding the variable region of the antibody along with other factors creates substantial ABS diversity.

5.1.6 The Special Role of the CDRH3

Of the three CDRs on each light- and heavy chain, the CDR3 is the most variable. The CDR1 and -2 are coded by the V segment whereas the CDR3 stems from the junction between segments (VDJ in heavy chain genes and VJ in light chain). The CDR3 is often considered the most determinant, in the sense that by only changing the CDR3 and leaving the CDR1 and 2 the same the antibodies can recognize and distinguish between many targets (Xu & Davis, 2000). Inspection of 825 antibody-antigen complexes from the Antibody Database (AbDb) found that the CDRH3 was the only region that was involved in ligand-binding in all studied complexes (Akbar et al., 2021). Research suggests that this particular loop is also important in determining polyreactivity, specifically as grafting polyreactive CDRH3 onto non-polyreactive antibodies has been shown to make said antibody polyreactive (Ditzel, Itoh, and Burton 1996).

5.1.7 Levels of Antibody Cross-Reactivity

When an antibody binds more than one antigen, the phenomenon is called cross-reactivity. Cross-reactivity can be divided into several categories depending on the level of binding specificity: conserved recognition, promiscuity and polyreactivity (Robert, Marschall, and Meyer-Hermann 2018). The first two categories describe binding to closely related antigens, where the epitopes are conserved (conserved recognition) or mutated variants (promiscuity, typically point mutations) where the difference due to the mutations does not abrogate binding respectively. Polyreactivity, however, denotes binding to unrelated or diverse antigens (Notkins 2004). By definition, polyreactivity can be expected to display the least specific binding patterns compared to the aforementioned types of cross-reactivity.

Some have argued for another class defined as polyspecificity (Cunningham et al. 2021). Polyspecificity would then denote a more "specific" binding to multiple antigens (resembling the idea of "oligo-reactivity" characterizing an antibody that is specific to only a few antigens (Notkins 2004)) while polyreactivity would refer to more "unspecific" binding to a larger number of antigens. However, polyspecificity and polyreactivity have been used somewhat interchangeably to describe binding to structurally and sequentially different antigens, and their meaning is variable in different studies. Here, the general concept of polyreactivity will be used for antibodies that bind more than one (i.e., an arbitrary number of) unrelated antigens. One could further subdivide polyreactivity into unspecific polyreactivity and oligo-reactivity based on the number of antigens bound.

5.1.8 Implications of Polyreactivity

Polyreactivity has been suggested to have various implications for antibodies' role in fighting disease and regulating bodily homeostasis (regulation of normal steady functioning). Evidence has been presented that polyreactive antibodies play important parts in regulating gut homeostasis in mice (Bunker et al. 2017) and development of tolerance (Dimitrov et al. 2013). Polyreactivity has also been suggested to be a feature of broadly neutralizing antibodies, fighting certain viruses with rapidly mutating genomes like HIV (Mouquet et al. 2010) and Influenza (Guthmiller et al. 2020). However, possibly the most important role of polyreactivity (and cross-reactivity in general) is to increase the number of targets the adaptive immune system can recognize.

Because antibodies can recognize more than one antigen the immune system is able to react to a larger variety of antigens than would otherwise be possible (and that some antibodies neutralizing a virus strain can still neutralize, to a certain extent, mutated variants). Even though the sequence diversity in the naive antibody repertoire is large (estimated at 10^{13})(Greiff et al. 2017), there exist a finite number of potential ABS sequence variants but an infinite number of potential targets (Dimitrov et al. 2013). If each antibody only recognized one target, this could leave holes in the adaptive immune response where antibodies would not be able to respond to every potential intruder. Holes in adaptive immunity due to the limited number of recognizable structures could be exploited by pathogens which mutate their genome to evade the immune system.

However, among the different types of cross-reactivity, polyreactivity is usually considered an undesirable trait in therapeutic antibodies. Partly because antibodies recognizing non-target self antigens can cause harm to healthy tissue (Zorn and See 2017). Polyreactive antibodies also clear faster from the bloodstream (Kelly et al. 2015). A study showed that, between antibodies that progressed to step 2 or 3 clinical trials and those approved, red flags for polyreactive behavior were increasingly rare in antibodies that had progressed farther in the process (Jain et al. 2017). In general, despite polyreactivity being favorable for immunity, polyreactivity in therapeutic antibodies can decrease the chance of successfully obtaining approval as a marketable therapeutic.

5.1.9 Properties of Polyreactive Antibodies

Some features that are thought to be correlated with polyreactivity. The most commonly mentioned feature is perhaps the increased flexibility of the antibody binding interface (Dimitrov et al. 2013). Molecular dynamics simulations have been used to show increased number of- and flexibility between conformational states in polyreactive antibodies (Fernández-Quintero et al. 2020). Other features that have been proposed include a longer CDRH3 (Aguilera et al. 2001), though this has been found to be insignificant in anti-HIV antibodies (Mouquet et al. 2010) and in a study of therapeutic antibodies (Lecerf et al. 2019), as well as the proportions or numbers of certain amino acid residues in the CDRs.

Amino acid composition of the ABS has been linked to increased polyreactive binding. Multiple studies reported a relationship between the number of one or several amino acids or physicochemical properties belonging to certain amino acids and polyreactivity (Rabia et al. 2018; Lecerf et al. 2019; Birtalan et al. 2008; Kelly et al. 2018). Among the amino acids which have been mentioned are the charged amino acids. Positively charged amino acids (Arginine and Lysine) have been linked to increased polyreactivity (Rabia et al. 2018; Lecerf et al. 2019). Correspondingly negatively charged amino acids have been negatively correlated with polyreactivity (Rabia et al. 2018). In the same study the authors reported that polyspecificity could with some accuracy be predicted from the net charge of the CDRs. Tyrosine, Serine and Glycine were found to contribute to specific but not non-specific binding in a study looking at their contribution to antibody affinity (Birtalan et al. 2008), whereas Arginine contributed to non-specific binding. Tyrosine in particular has been identified as a major contributor to both affinity and specificity (Koide and Sidhu 2009), although several studies have reported increased presence of Tyrosine in polyreactive sequences (Thompson et al. 2012; Harvey et al. 2022). Other amino acids that have been implicated in polyreactivity include Tryptophan, Valine and Glutamine (Kelly et al. 2018; Lecerf et al. 2019). However, the impact of amino acids can be context dependent (i.e., depending on other amino acids either in a neighboring position or farther away in the sequence) and dependent on the target antigens and the experimental conditions.

It has been reported that the correlation between amino acids and polyreactivity can vary due to sequence position and due to the local sequence environment. In a study of 1000 antibodies from different sources, the properties of the amino acid sequence of the CDR loops were reported to vary depending on position, while charge and hydrophobicity were linked to polyreactivity (Boughter et al. 2020). A study looking at contributions of sequence motifs to non-specificity found that tryptophan was tied to polyreactivity overall but in particular when occurring in motifs, also while tyrosine was depleted in polyreactive sequences two consecutive tyrosine residues contributed to polyreactivity (Kelly et al. 2018). An examination of binding interfaces from two anti-Amyloid β antibodies showed that, while both contained similar amounts of arginine, the arginine mutations in the less specific antibody contributed more to binding and the surrounding sequence were comparatively more hydrophobic (Tiller et al. 2017).

The type of assay performed, and the experimental conditions, have been found to influence observed polyreactivity (Labrousse, Adib-Conquy, and Avrameas 1997; Lecerf et al. 2019). It has been demonstrated that temperature can alter whether an antibody is polyreactive (Labrousse, Adib-Conquy, and Avrameas 1997). When different experiments are used to evaluate polyreactivity, the amino acid composition and physicochemical properties that are enriched in polyreactive sequences can differ (Lecerf et al. 2019), both in terms of significance of enrichment but also in terms of what group (specific or polyreactive) was enriched with certain amino acids in the CDRH3. For example ELISA and baculovirus particle (BVP) assay yielded polyreactive antibodies significantly enriched in positively charged antibodies. Using polyspecificity reagent (PSR) yielded polyreactive antibodies with significantly reduced number of hydrophobic amino acids.

5.1.10 Experimental Assessment of Polyreactivity

Several experimental methods have been developed and used to assess polyreactivity. Competitive binding assays have been utilized to assess cross-reactive binding (Tiller et al. 2017). Large protein microarrays have been used to assess polyreactivity as well (Planchais et al. 2019). An assay has been described assessing polyreactivity to a selection of 32 proteins (Frese et al. 2013). However, there are some methods that are commonly used to evaluate polyreactivity between antigens.

Common methods include the enzyme-linked immunosorbent assay (ELISA), against small panels of selected antigens (Wardemann et al. 2003). The antigens are chosen to cover a range of different physicochemical properties and include different types of macromolecules. Reactivity is determined by setting a threshold based on binding of negative control antibodies (Mouquet et al. 2010; Mietzner et al. 2008). Flow cytometric methods have also been developed for evaluating polyreactivity, such as the polyspecifisity reagent (PSR) assay which evaluates polyreactivity of IgG antibodies in yeast display against proteins from Chinese Hamster Ovary (CHO) cells (Xu et al. 2013). This method enables negative selection for polyreactivity while simultaneously screening for high affinity to the desired target. Another flow cytometry essay was recently developed where monoclonal antibodies are displayed on tiny magnetic beads (Makowski et al. 2021).

5.1.11 Prediction of Polyreactivity in Silico

No widely used *in silico* method exists to predict polyreactivity as of now, unlike some developability metrics (Akbar et al. 2022; V. K. Sharma et al. 2014). Developability describes the ability of an antibody treatment to progress from discovery to development (Bailly et al. 2020). Developability parameters refer to certain key features predictive of developability. Examples include thermal stability, solubility and clearance to name a few (Mason et al. 2021; Bailly et al. 2020). There exists *in silico* methods for scoring antibodies based on several of these parameters (Mason et al. 2021). Some developability parameters are tied to specificity such as hydrophobicity and presence of positively charged patches (Raybould et al. 2019). However, these parameters are not directly predicting polyreactivity.

5.1.12 In Silico Prediction of Antibody-Antigen Binding

Besides polyreactivity, computational methods for predicting the properties of antibody binding (affinity, specificity and paratope-epitope prediction) have been developed in recent years. Robust prediction and generation of antibodies that are able to bind desired targets have great potential in speeding up and reducing cost of antibody discovery and development (Akbar et al. 2022). *In silico* methods for antibody design have traditionally included methods such as homology modeling and protein-protein docking (Norman et al. 2020). Machine Learning methods have also widely been applied to antibody-antigen datasets albeit small size datasets (Akbar et al. 2022).

5.1.13 Machine Learning

ML refers to methods by which a computer can make predictions regarding an outcome based on data. Instead of having a human manually identify rules and patterns of information to create predictive models, model creation is outsourced to a machine. The learning part of machine learning refers to how parameters of ML models are updated to improve model accuracy through iterative learning processes when exposed to more examples (Naqa and Murphy 2015).

Supervised learning, a subfield of machine learning, teaches a computer algorithm to predict an outcome through training on labeled data. Labeled data describe data where the outcome is available as well as other data regarding the features of each given case. The algorithm can learn to make predictions regarding a certain outcome on unseen data through training an algorithm to make predictions by recognizing patterns in labeled examples.

5.1.14 Overfitting and Underfitting

A fundamental problem in ML is the tradeoff between model underfitting and overfitting (Jabbar and Khan 2014). Overfitting describes a state where the model performs well on previously seen (training) data but does not generalize well to unseen data. A model which underfits is not able to capture the relationship between the independent variables and the label both in training and unseen data. Overfitting is due to learning biases in the training data and often affects flexible models capable of picking up on complex patterns (Yin et al. 2015). Underfitting is due to overly simple models that are not capable of capturing the relevant complexity. During model selection over- and underfitting can be diagnosed by high variance or bias respectively. High variance is observed as low performance on test/validation data as compared to the training data and high bias as low performance overall (Raschka and Mirjalili 2019). When selecting a model one typically wants to select a model that is not complicated enough to overfit but is complex enough to capture relevant relationships in the data.

5.1.15 Machine Learning Architectures

There exists several popular ML architectures used to solve various supervised prediction tasks (Raschka and Mirjalili 2019). These architectures can roughly be divided into traditional (not neural network) machine learning architectures and neural networks. Examples of traditional models include models such as linear and logistic regression, support vector machines, decision trees and ensemble methods like random forests. Neural networks are models made up of one or more layers of artificial neurons, which are processing units inspired by biological neurons. In this thesis we will refer to the traditional models as “shallow” and the neural networks (more than one layer) as neural networks, though neural networks without several hidden layers are often also referred to as “shallow” . The various ML architectures have different advantages and limitations. It is not always obvious what architectures perform the best for any given problem (“Machine Learning in Bioinformatics: A Brief Survey and Recommendations for Practitioners” 2006; Libbrecht and Noble 2015).

ML architectures differ in their ability to model nonlinearities and interactions between input features (Christoph Molnar 2022c; Dreiseitl and Ohno-Machado 2002; Tu 1996). Feature interactions refer to how the value of one feature can influence the effect that another feature has on the final prediction. A logistic regression model can only handle each input feature separately and is not able to capture nonlinear relationships between the input features unless this relationship is provided as a separate feature (Dreiseitl and Ohno-Machado 2002). In the case of classification models like logistic regression, they work well when the classes are linearly separable based on provided features. Tree-based models or neural networks (Tsang, Cheng, and Liu 2017) on the other hand are able to capture dependencies between the individual input features.

Tree-based models (including decision trees and random forests) and neural networks capture nonlinearities in different ways. Decision trees make predictions through a series of questions parting the data points into nodes with increasingly homogenous data points based on the answer. The series of questions forms a hierarchy with the end nodes (termed leaf-nodes) determining prediction (Kingsford and Salzberg 2008). Random forests are ensemble models which base decisions on averaging the outputs of multiple decision trees. Decision trees can be prone to overfitting. Random forests leverage the ability of individual trees to fit well to subsets of the data to build models which are typically more robust (Tang, Garreau, and von Luxburg 2018). Because of the hierarchical nature of decision trees (the next question is determined by the answer to the previous) they are capable of modeling feature interactions.

The ability of feed-forward neural networks to model non-linear interactions comes from hidden layers of several neurons whose output is decided by a non-linear activation function (Tsang, Cheng, and Liu 2017). The artificial neurons in the network take the weighted sum of the input features (plus bias) and apply an activation function. A single neuron can resemble models like logistic regression (Raschka and Mirjalili 2019) (though the activation function may not be sigmoid). Without introducing a nonlinearity, the artificial neuron behaves like a linear model. A neural network with at least one hidden layer requires a nonlinear function applied to each hidden layer, otherwise it could be seen as equivalent to a linear model (S. Sharma, Sharma, and Athaiya 2020) (a logistic regression model if we assume that the output layer contains a sigmoid function).

Convolutional neural networks (CNNs) are a special type of feed forward neural networks particularly developed to capture local feature patterns. CNNs were inspired by the organization of neurons in the part of the visual cortex which process visual information (Fukushima 1980). They contain convolutional layers, consisting of filters which perform convolutions between sections of the input and a kernel with learnable weights. The results from each convolution are stored in a feature map and passed on to the next layer. Non-linearities in CNNs are introduced through non-linear activation functions and non-linear pooling operations (Zhang et al. 2021; Gulcehre et al. 2014; Tsang, Cheng, and Liu 2017; Wei et al. 2019). CNNs are popular models in several fields including biomedical applications, and predicting antibody-antigen binding (Mason et al. 2021). Due to their ability to capture local patterns they have also been used for data mining purposes in bioinformatics research like mining regulatory motif discovery in genomics (Quang and Xie 2016; Alipanahi et al. 2015).

5.1.16 Machine Learning for Antibody-Antigen Binding Prediction

It has long been an open question whether or not antigen-antibody binding prediction could even be possible (Akbar et al. 2021). It has often been taught that the complexity of the task was too great for reliable prediction, because of the large amount of possible variation in antibody sequences and in antigens, and the comparatively small size of available datasets. There is also a lack of data on antibody-antigen binding due in part to experimental assays only isolating antibody sequences with high and low affinity, without further knowledge of affinity or the paratope-epitope interface. However, methods have been developed to predict paratope-epitope binding interfaces (Pittala and Bailey-Kellogg 2020). A study (Akbar et al. 2021) has shown that antigen-antibody interaction can be predicted using a simplified encoding of antibody-antigen interactions (called structural interaction motifs), creating a vocabulary of antibody-antigen interactions with reduced complexity. The existence of a reduced complexity vocabulary describing antigen-antibody binding suggests that there are universal rules for antibody-antigen binding that can be learnt from small-scale structural datasets (~825 structures).

The ability to identify candidate antibodies with increased affinity and good developability profiles (according to estimated developability parameters), using a CNN has been demonstrated on binders of the HER2 antigen (Mason et al. 2021). The authors were able to increase the number of promising candidates using the CNN to filter a large number of potential candidate sequences. The authors showed experimentally that 30 of the selected sequences had increased affinity to HER2, demonstrating the usefulness of *in silico* prediction tools for this purpose.

5.1.17 Benchmarking

Benchmarking refers to comparing strategies based on chosen metrics, often in comparison to a chosen benchmark metric (Olson et al. 2017; Reichenbach et al. 2019). In ML, modeling approaches are compared based on performance according to metrics of prediction correctness (though other metrics can be used such as speed) (Thiyagalingam et al. 2022). Examples can include metrics like accuracy, recall and precision for classification tasks. Benchmarking can refer to comparison between ML architectures, but it can also refer to comparison between other factors like encodings or different methods of splitting the data. In this thesis we will be benchmarking combinations of classification schemes (labeling of the classes), feature encodings and model architectures.

In order to get reliable estimates of prediction metrics, the predicted labels have to be compared to ground truth labels. Ground truth is referring to parameters of the data that are known. In ML ground truth typically describes information that is known to be true, which is used to evaluate the model inferences (Techopedia 2017). As an example, for classification of images (based on what they portray) true class-labels can be determined by empirical observation.

5.1.18 Simulated Data

Data that is generated *in silico* to replicate real-world data on a particular phenomenon, (simulated data), can be used to evaluate the effectiveness of ML methods. Simulated data has been used to guide methodological development when large scale ground-truth datasets have not been available (Prakash, Shrikumar, and Kundaje 2022; Earl et al. 2014; Weber et al. 2020). The premise of using simulated data is that simulated data is capable of reproducing important characteristics of the data that the methods are designed to be used on. With simulated data, the process behind how the data is generated and the features of the data is known. For the benchmarking of ML methods to predict antibody-antigen specificity, the simulation software `Absolut!` was created (Robert et al. 2022).

In order to overcome the lack of data available for benchmarking ML approaches for prediction of various levels of antibody binding `Absolut!` was made to allow for unconstrained generation of data on antibody-antigen complexes (Robert et al. 2022). A simplification of antibody-antigen binding, the simulations estimate binding energy between discretized lattice representations of antigens and CDR sequences of fixed length. Through exhaustive docking, the best folding pose of the antibody onto the antigen is identified, and the energy calculated based on experimentally derived energy potentials. `Absolut!` is a tool to enable investigation into different methods and approaches, with the data capturing important levels of biological complexity. Such levels include biological amino acid composition and topology of the antigen, amino acid composition and sequential dependencies as a result of using experimentally derived sequences, and reactivity networks similar to ones observed with experimental data. That conclusions from `Absolut!` are transferable to real-life experimental data was validated for a select number of tasks, such as ranking models for predicting affinity to the cancer antigen HER2 and evaluating the impact on data augmentation on pose classification.

5.1.19 ML-Based Prediction of Antibody Polyreactivity

When it comes to ML prediction of polyreactivity, one *in silico* pipeline has been proposed using a Support Vector Classifier (SVC) on centered positional matrices representing CDR sequences experimentally determined to be polyreactive (Boughter et al. 2020). The antibodies included in the study were evaluated for polyreactivity using ELISA. The authors proposed that polyreactivity was due to a neutral binding surface. The reported accuracy of the SVC model was above 0.75. However, the utilized data exclusively included antibodies that showed greater levels of non-specific binding (4 out of 4-7 antigens) compared to antibodies binding none of the antigens in the ELISA arrays.

It is not clear which kinds of ML architecture or feature encodings would be good for predicting polyreactivity. To further improve machine learning tools for polyreactivity prediction, benchmarking approaches could help solve this problem.

5.1.20 Works on ML Prediction of Polyreactivity Published During this Project

In a recent preprint, (Harvey et al. 2022), the authors used ML methods to predict (AUC ~0.8) and modify the degree of polyreactivity in nanobody sequences. The most successful method used on data of size 110,931 was logistic regression on one-hot encoded sequences. When the size of the dataset was increased to > 2 million clones they found comparable AUC for an RNN and CNN model. In their study they designed the training and test datasets in such a manner that there were not highly similar sequences to the training sequences included in the test data.

5.1.21 Data Leakage

The suitability of a ML method is usually determined by the performance on chosen accuracy metrics when generalizing to unseen data (generalization). When there are samples in both the training and test dataset which are very similar, it is hard to know if a trained model has reached high accuracy by learning generalizable patterns (desirable), or if it has overfitted elements in the training dataset and predicts based on similar known elements in the training (undesirable). Sequence similarity can pose problems for ML predictions of protein functions based on sequences (Petti and Eddy 2022). In particular, if the function that is being predicted (for instance polyreactivity), is tied to or heavily correlated with similarity between sequences, high accuracy due to similarity could become a problem. Having exceedingly similar members of the same protein family separated into training and test data can be seen as a form of data leakage (information in the test dataset was already given in the training dataset), and having a lot of highly similar sequences in training and test datasets can lead to problems of poor generalization.

Problems of similarity biasing protein function predictions is often due to the evolutionary relationship of sequences within the same protein-family (Petti and Eddy 2022). Members of a protein family typically share biological functions. Naive antibody sequences have not yet evolved by affinity maturation. Of note, the hosting lab has shown on `Absolut!` data that very similar antibody sequences can bind to different antigens (Robert et al. 2022). However, it is not to our knowledge known whether polyreactivity is predictable by similarity (neither in *in silico* data nor in real-world experimental data). Exploring how similarity may bias polyreactivity prediction would help elucidate whether model performance could be due to similarity or to what extent the model learns generalizable patterns.

5.1.22 Interpretability

The main task of ML is usually to find a good approximation of a function, which does not necessarily need an explanation for high accuracy. However, it can often be advantageous to examine the model more closely and to find how a model achieved high accuracy (i.e. input patterns that were learnt by a model) (Lipton 2018). Indeed, in domains where trust in the model predictions is imperative, traditional metrics such as model accuracy is often not sufficient and there is a greater emphasis on model interpretability and explainability (Ahmad, Eckert, and Teredesai 2018). Interpretation of the model can be useful in order to examine potential biases which may lead to overfitting (an example is when an image-based animal recognition model recognizes wolves based on the background snow) and to better understand the underlying problem and use the insight to inform further inquiry.

In order to facilitate such examination the model would have to be somehow interpretable. That is, there would have to be a way to interpret how the model makes a decision in a way that is understandable to humans (Gilpin et al. 2018). Some models are typically considered interpretable, such as short decision trees and logistic regression (Christoph Molnar 2022c). These models possess internal model features that can be extracted to directly interpret the model, such as the coefficients of linear models (Christoph Molnar 2022a). Other models are more challenging to interpret. Models which are challenging to examine are traditionally called “black boxes". Neural networks are examples of such models. There have been several methods proposed for extracting feature attributions (i.e., giving a weight on the features of a given input element, related to the contribution of those features in the model output). These methods aim at ascertaining how the features of an input contribute to the output and what features the model focuses on when making predictions. One of these is called integrated gradients (Sundararajan, Taly, and Yan 2017), which computes the average over gradients for an output with respect to linear interpolations between the input and an arbitrarily defined baseline. With integrated gradients, elements of input with a high impact on the output are attributed a higher gradient, which can help localize predictive patterns in the input elements.

6 Aim of the Thesis

The overarching goal of the thesis is to benchmark supervised ML classifiers for antibody polyreactivity in terms of how well they classify sequences with varying levels of polyreactivity. Benchmarking classification algorithms can help elucidate how best to predict polyreactivity. We are interested in what types of model architectures work well for these tasks and what types of sequence features may be predictive. Specifically, we will focus our attention on the following goals.

6.1 Goal 1: Comparing combinations of model architecture and sequence encoding for creation of supervised ML models to predict polyreactivity

The first aim we will pursue is comparing ML architectures for prediction of polyreactivity based on > 100.000 sequences. Which model architectures are best for predicting polyreactivity is an open question. We will compare architectures, both traditional "shallow" models as well as neural networks. Polyreactivity will be defined by the number of antigens bound by the sequences between 142 proteins, as binary or multiclass problems. The macro f1 scores achieved by each model will be examined as well as confusion matrices.

1. **Between a selected list of ML model architectures, how do these models compare in their ability to correctly classify antibody CDRH3 sequences based on observed levels of "polyreactivity"?**

In order to evaluate what type of sequence features are important for predicting polyreactivity we will use different encodings, primarily amino acid composition and one-hot encoding. We will also compare the performance of models capable of capturing complex feature interactions in comparison to models that are not (i.e. logistic regression).

2. **Does providing explicit positional information improve the models ability to predict levels of polyreactivity?**
3. **Is polyreactivity prediction improved by using models that capture nonlinear relationships between features?**

4. **In the interest of identifying minimal pattern length predictive of polyreactivity, how does prediction accuracy change in response to increase in the possible length of recognized patterns?**

6.2 Goal 2: Investigate overestimation of model generalizability due to close similarity between sequences in the training and validation datasets (Data Leakage)

For the models to be able to generalize to unseen sequences they should be able to learn functionally relevant patterns and not predict based on functionally irrelevant similarity. We will evaluate whether the performance of selected models are due to data leakage, referring to similarity between training and validation/test sequences in terms of Levenshtein distance (LD).

5. **Is the predictive power of the models dependent on close similarity in terms of LD to already observed sequences?**

6.3 Goal 3: Look into interpretability of select models

Finally, we will look into interpretability of the selected models. Demonstrating how the logistic regression model can be interpreted and, in accordance with the goal of determining what features are most predictive, examining how the models make predictions. To look into interpretability of the neural networks we will extract feature attributions and determine if the integrated gradients method can find relevant attributions.

6. **What type of sequence features does the logistic regression model emphasize when predicting polyreactivity?**
7. **Can attribution methods, like integrated gradients, capture sequence features which impact the prediction of the model?**

7 Results

7.1 Goal 1: Comparing combinations of model architecture and sequence encoding for creation of supervised ML models to predict polyreactivity

7.1.1 Formalization of Antibody Polyreactivity as an ML Problem

In this work, we focus on predicting the degree of antibody polyreactivity from the sequence of their CDRH3 (treated as 11-mers), using different ML strategies. In order to predict polyreactivity, the problem had to be formulated into an ML task. There could be different ways to define antibody polyreactivity in terms of defining the model outputs.

Table 1: **Formulation of problem as a machine learning task** We decided to approach the problem of predicting polyreactivity by predicting binding to several antigens among many (unspecific) rather than binding to particular antigens.

Aim:	Input-output:
Predict the degree of (unspecific) polyreactivity of an antibody, as a measure of how many antigens it is estimated to bind	CDRH-3 11-mer sequence, → ML → class (based on degree of polyreactivity)

Formalizing a "degree of polyreactivity" requires a quantitative definition. Antibody polyreactivity generally refers to binding several structurally different or sequence unrelated antigens (i.e., not homologous nor with highly similar sequence). There are different polyreactivity properties of an antibody that could be predicted. We decided to predict what we call "degree of polyreactivity" as the number of antigens bound by an antibody, among a predefined set of antigens. Predicting other levels of polyreactivity, such as predicting the paratopes or epitopes to the multiple bound antigens or measures of dissimilarity between the epitopes that one antibody binds, would also be possible and is left for future work. In order to predict "degree of polyreactivity" based on the number of bound antigens we had to further define the task based on how the number of antigens should correspond to a label.

We base our work on an *in silico* database of antibody-antigen complexes containing 142 peptide/protein antigens. The included antigens are non-redundant, meaning that they represent different proteins (they are not closely related mutants of the same protein) or they are treated as different proteins by the framework (see Methods). For these antigens the binding energy of 6.9 million CDRH3 sequences has been assessed (Robert et al. 2022) (further information to be found under Methods). The binding energy was calculated by stretches of 11 AAs of the CDRH3s (11-mers). In this work we are working with the 11-mers that achieved suitable binding energy per a threshold as binders. To get a label by which the CDRH3 sequences could be classified based on "degree of polyreactivity" we had to decide a threshold for how many antigens the sequence had to bind in order to be considered "polyreactive".

The number of antigens, of any given list of antigens, an antibody has to bind to be defined as polyreactive is not known (Boughter et al. 2020)**.** To our knowledge, there does not exist an agreed upon definition of polyreactivity that can be extended to binding between any selection of antigens. There are different possible ways to define polyreactivity based on the number of bound antigens between the 142 antigens included. A first possible formalization is to consider any 11-mer that binds more than one antigen as polyreactive. Alternatively, polyreactivity can be defined as binding to x number of antigens where x is a number greater than 2.

It has been proposed that antibodies can bind more than one antigen in a "specific"-manner (Cunningham et al. 2021)**.** In that case not demonstrating broad reactivity but rather a pattern of binding two or few antigens to the exclusion of others. Such could be the case in our data as well. One can also define a third group of antibodies as oligo-specific, where the antibody can bind to a small number of unrelated antigens (Notkins 2004). It may also be that polyreactivity is best described through more than two or even tree classes but is for example better defined through multiple classes with increasing polyreactivity. In the previously published elife paper (Boughter et al. 2020) looking at prediction of polyreactivity only the most specific and most polyreactive antibodies were included in the study to avoid confusion due to varying levels of polyreactivity. Since the purpose of their analysis was to ascertain the possibility of predicting polyreactivity the removal was deemed to not be a big issue. In this work we want to include all levels of polyreactivity as removing sequences with intermediate levels of polyreactivity may distort measures of model accuracy. The reported accuracy would not reflect how well the models work on potentially more difficult to classify "borderline" cases. **To investigate these different possibilities, several binary and multiclass tasks were formalized for antibody polyreactivity, and datasets were created accordingly, differing by the definition of polyreactivity classes (**Figure 2**).**

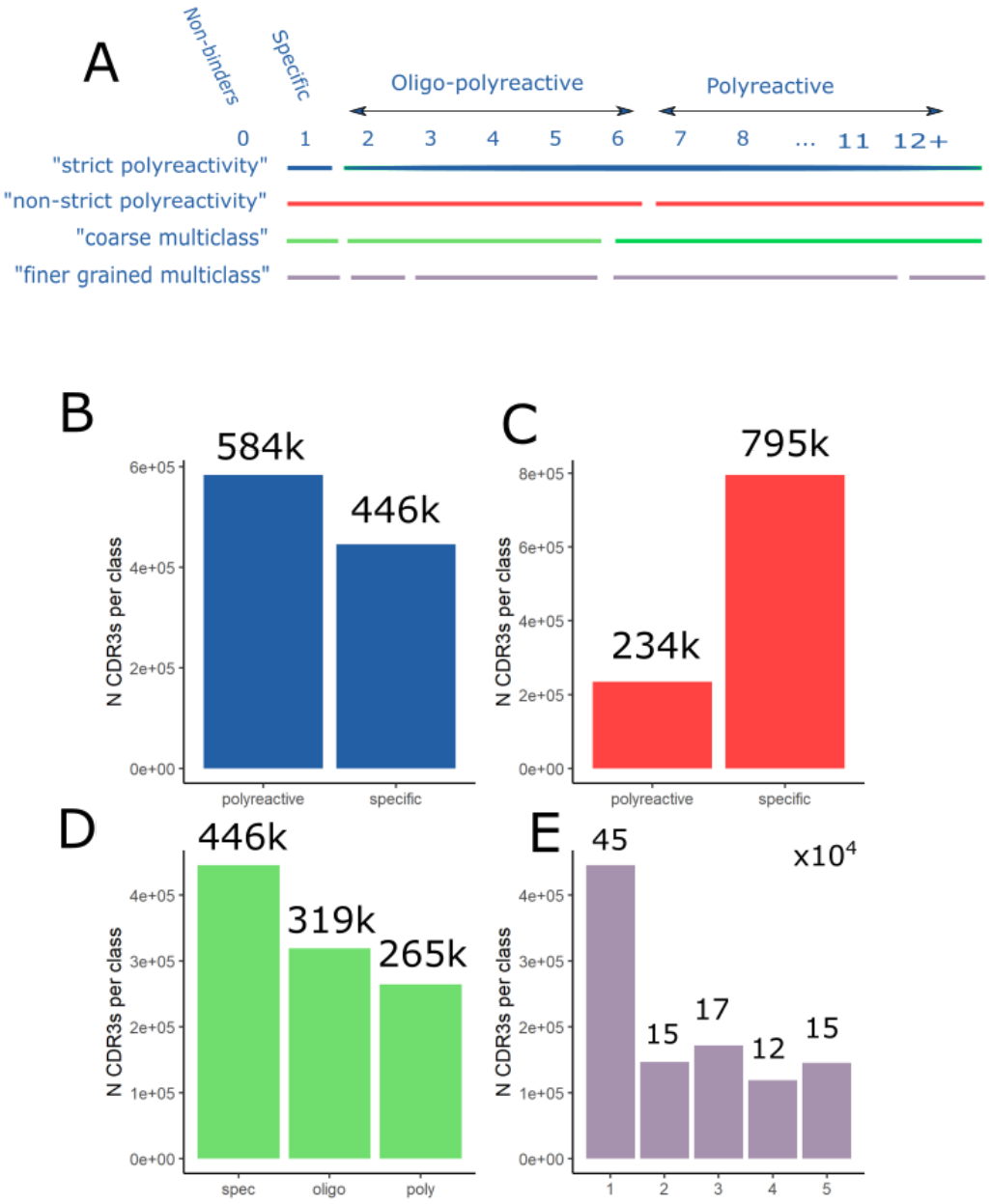


Figure 2: **Formalization of antibody polyreactivity prediction tasks as classification problems,** and size of the classes within the dataset available for each task before balancing. (A) definition of four ML problems varying in their definition of classes, depending on the degree of polyreactivity of antibodies. (BCDE) Number of available sequences for each class of each ML problem. The numbers above each column represents the number of sequences in the class, rounded to the nearest 1000. Except for in sublot E where they were rounded to the nearest 10.000.D. B) The size of classes in data classified as per task "strict polyreactivity". C) The size of classes in data classified as per task "non-strict polyreactivity". D) The size of classes in data classified as per task "coarse multiclass". E) The size of classes in data classified as per task "finer grained multiclass".

The separation between classes was manually defined in order to have minimal inequality of class size while fulfilling the role of the datatask (Figure 2**).** Emphasis on class size equality had the purpose of avoiding defining classes with very few sequences. The classes in "strict polyreactivity" (Figure 2) were almost balanced to begin with, the specific antibodies making up about 40% of the dataset (446k vs 584k which is approximately 0.43). In "non-strict polyreactivity" the polyreactive sequences made up a bit less than 25% of the data as the decision boundary was decided by the third quartile of the binding distribution. For the "finer grained multiclass" task, the smallest number of sequences in any class was 12k, which would leave 12k sequences for each class after balancing. The exact number of sequences in each class is given in Table 3 (see Methods).

7.1.2 Amino Acid (AA) Composition differs between Polyreactivity Classes

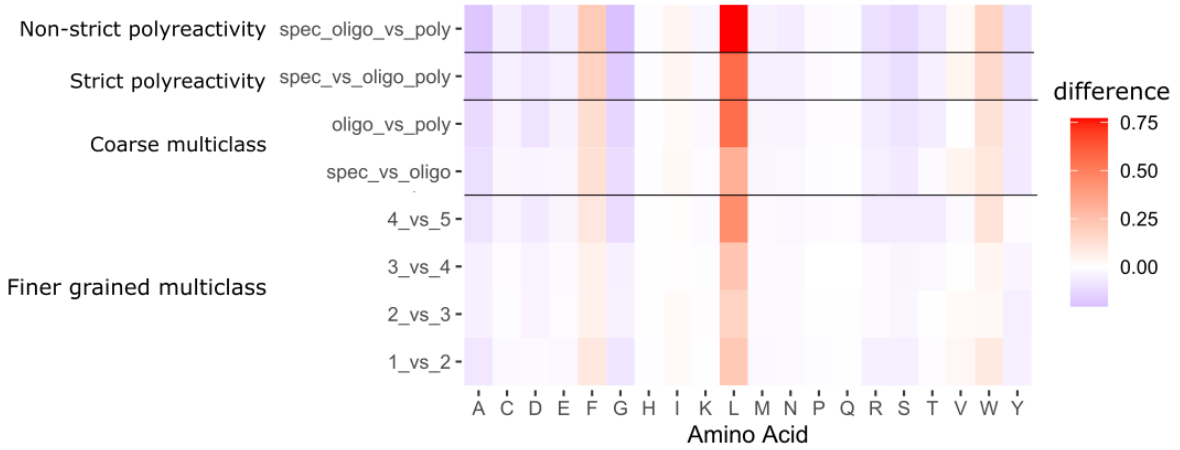


Figure 3**: Amino acid composition correlates with levels of polyreactivity.** Difference (subtraction) in expected occurrence of each amino acid in one class (higher polyreactivity) from another (lower polyreactivity). The y-axis denotes the classes being compared. Comparisons are sorted by datatask (left of the y-axis). For the finer grained task the number represents degree of polyreactivity from 1 (specific) to 5 (most polyreactive). Amino acids which were more common in polyreactive sequences are colored in red and those were more frequent in specific sequences blue.

We first looked for differences in amino acid (AA) occurrences between classes which could potentially be predictive of polyreactivity. To this end we quantified the AA composition of sequences depending on class according to the defined tasks (Figure 3). We show the pairwise enrichment of AAs between two classes where the more polyreactive class of the two is compared to the more specific one. The enrichment was calculated as the expected occurrence of the AA in sequences of the more polyreactive class subtracted by the expected occurrence in the more specific class. The expected occurrence was calculated as the mean occurrence. The results thus represent the increase in the number of times a particular residue would be expected to occur in sequences that are more polyreactive vs less.

AA composition differed between all classes that were compared (Figure 3**).** We observed that there were differences in expected occurrence of several residues for all compared classes. The expected AA occurrence differed between classes for most residues. These differences in AA composition could potentially be leveraged to predict degree of polyreactivity. Expected leucine occurrence represented the largest enrichment of any AA within sequences of the more polyreactive classes. Though the expected occurrence of Leucine was high in all classes (see Appendix), Leucine was consistently more common in the sequences of the more polyreactive class. The direction of enrichments were largely consistent across class comparisons.

Which AAs were enriched or depleted in more polyreactive sequences remained mostly consistent between all class comparisons. Expected occurrence of Leucine (L), Phenylalanine (F), Tryptophan (W), Isoleucine (I) and in most cases Valine (V) was also higher in increasingly polyreactive sequences. Expected occurrence of Alanine (A), Aspartic acid (D), Glycine (G), arginine (R), Serine (S) and Tyrosine (Y) was higher in more specific sequences. The exceptions being the differences in Valine occurrence between class 4 and 5 as defined by the “finer-grained” multiclass task, and Tyrosine between specific vs oligoreactive sequences per the coarse grained multiclass task, compared to all other comparisons.

The magnitude of the differences varied between the different polyreactivity classes. As an example, the difference in expected Leucine occurrence was larger between the classes in "non-strict polyreactivity" than "strict polyreactivity". They were smaller between class “spec” and “oligo” then they were between class “oligo” and “poly” for task “coarse grained multiclass”. For the “finer grained polyreactivity” task the magnitudes were generally smaller between class 2 and -3 as well as 3 and 4 then they were between class 1 and 2 as well as 4 and 5. The latter comparison yielded the largest differences. **The differences in AA composition were in general larger when the most polyreactive classes were included (“polyreactive” or class 5), which were also the classes with the broadest range of- and extreme high values, in terms of number of antigens.**

7.1.3 Classifying Polyreactivity Based on Leucine Content Left Room for Improvement

We were interested to see whether any single amino acid occurrence would be a good predictor of polyreactivity. Since leucine was the AA which displayed the biggest frequency difference between classes, we looked at how the distributions of leucine content differed for the classes of each task and assessed whether leucine could be a good predictor of polyreactivity.

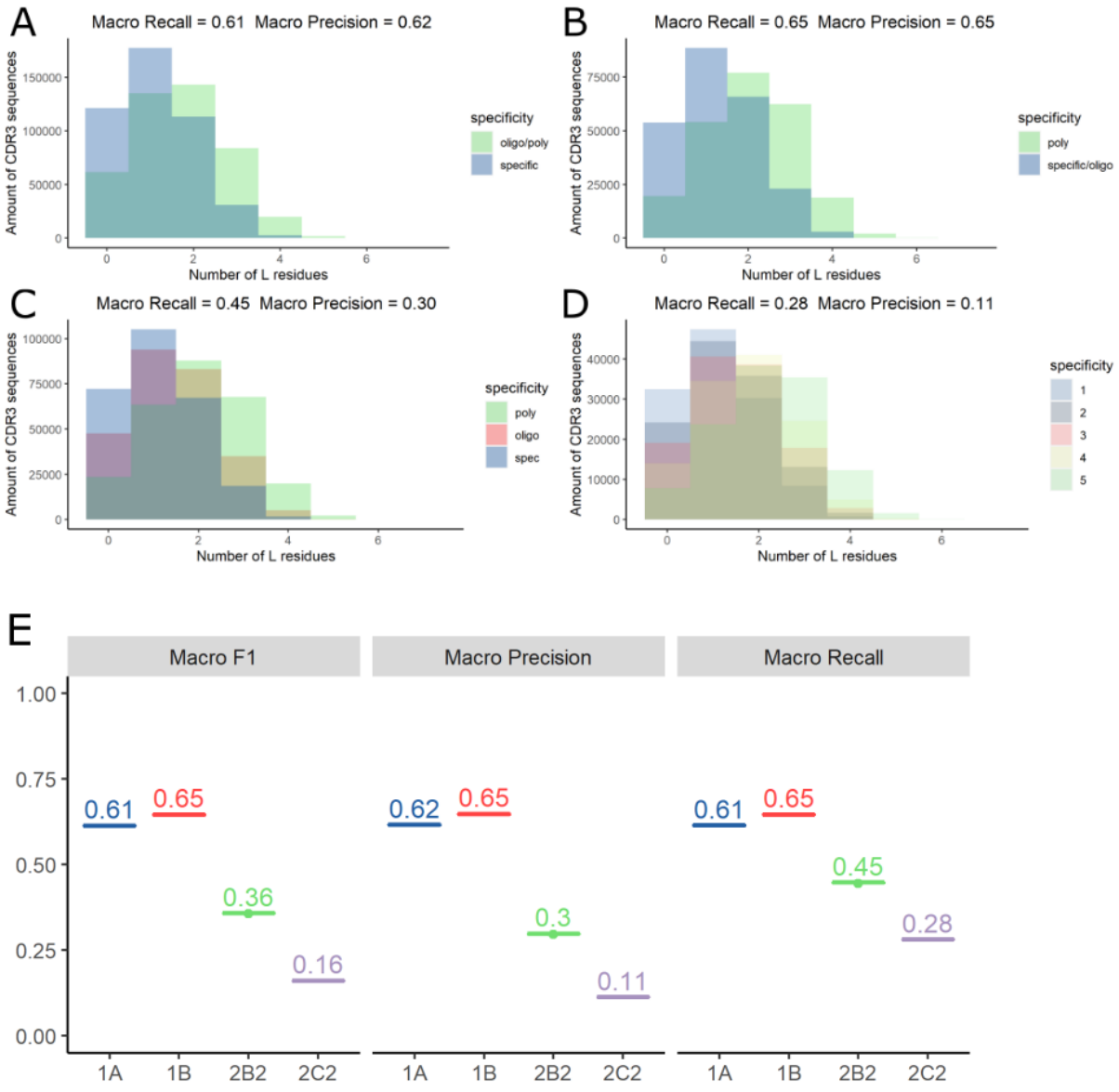


Figure 4: **Leucine content is not sufficient to predict polyreactivity.** ABCD) Histogram of how many Leucines occur in the CDR3 sequences of each class based on how the classes are defined (datatask).E) Macro averaged f1-score, precision and recall for logistic regression trained using only Leucine content as predictor. The horizontal axis represents each datatask ("strict polyreactivity": 1A, "non-strict polyreactivity": 1B ,"coarse multiclass": 2B2, "finer grained multiclass": 2C2.)

Polyreactive sequences often contained a similar number of leucines as more specific sequences. The distributions of Leucine content (Figure 4 A-D) within each class overlap. If Leucine content is high then the sequence is more likely polyreactive, however most polyreactive sequences contain a similar number of Leucines as observed in specific sequences (Figure 4 A and -B). A similar pattern can be observed for the multiclass classification schemes (Figure 4 C and -D). The leucine content distributions of the classes at each extreme of reactivity in "finer grained multiclass" were more separated, however even between these classes there was overlap.

Leucine content was not a good predictor of polyreactivity in and of itself. For binary classification of low vs high polyreactivity (Figure 4 E) the logistic regression achieved precision and recall of 0.65 which leaves room for improvement. For multiclass classification the model performed about as well as random (“coarse multiclass”) in terms of f1-score, precision being lower than recall. For the finer grained classification task the model achieved a low precision of 0.11. This was possibly due to the model not predicting any instances of certain classes as the distributions overlapped.

No other single amino acid occurrence separated the classes better than Leucine. The distributions of all AA frequencies separated by class, for both binary classes and "coarse multiclass", were included (see Appendix). No individual AA content distributions clearly separated the classes. Thus, no single AA frequency looked to be a good predictor by itself. We then later assessed whether AA composition as a whole (not only one AA) would be enough to classify the sequences using ML.

7.1.4 AA Enrichment in Polyreactive Sequences is Position Dependent

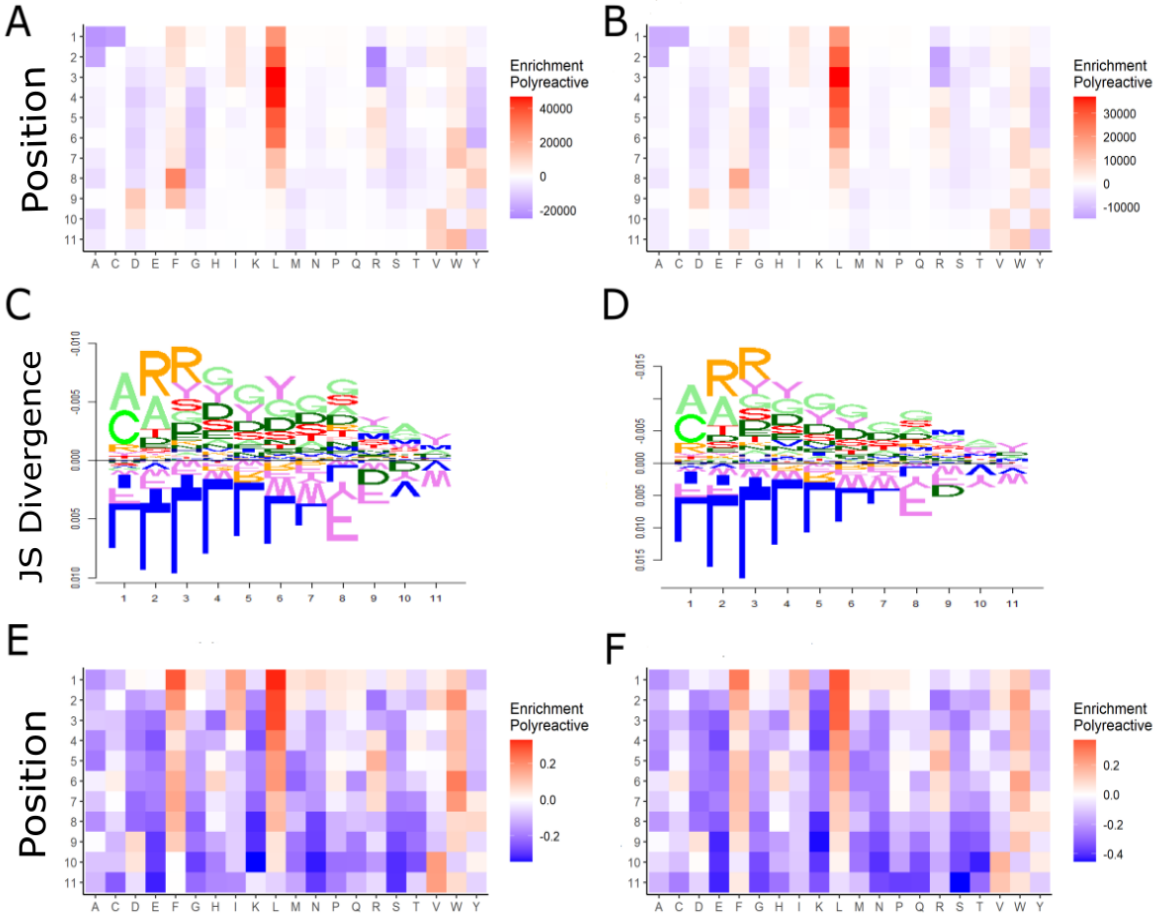


Figure 5: **Difference in amino acid frequency between classes depending on the position in the antibody 11-mer sequence.** A and B) Difference between the number of polyreactive sequences containing each residue (x-axis) at each 11-mer sequence position (y-axis) compared to specific sequences. The figures display differences per the "strict polyreactivity" and "non-strict polyreactivity" tasks respectively. C) Difference in sequence logos between strict polyreactivity classes where the height of each column on the y-axis displays Jensen-Shannon (JS) divergence. The size of the letters represent the probability of that amino acid occurring within the distribution associated with one class compared to the other (enrichment) normalized by the sum of the absolute probability differences in the distributions ("normalizedDifferenceOfProbabilities: Normalized Probability Differences in DiffLogo: DiffLogo: A Comparative Visualisation of Biooligomer Motifs" 2020), where the distribution is the probability distribution of each amino acid occuring in the given position (one direction of the column from the midline). The distributions above the midline represent probability distributions for each position in specific sequences and below the midline the same for polyreactive). D) JS divergence between non-strict polyreactivity classes. Similar to C. E) Similar to A but displaying differences relative to how frequently the residue (x-axis) occurs in that position (y-axis) overall. F) Same as E for "non-specific polyreactivity".

Analysis of AA composition dependent on position revealed that the enrichment in AA content within polyreactive sequences differed between positions (Figure 5**).** Besides the AAcomposition across the entire length of the 11-mers we wanted to look at the differences in AA composition for each position. Enrichment refers to the number of polyreactive sequences in the dataset, depending on the task, that contain the given feature (contained x AA at y position) subtracted by the number of specific sequences with the same feature. We observed that the AA enrichments differed between positions both in terms of magnitude and whether the residue was enriched or depleted in polyreactive sequences.

The enrichments observed in polyreactive versus specific sequences, defined by "strict polyreactivity" (Figure 5**A) and "non-strict polyreactivity" (**Figure 5**B), showed similar positional AA composition.** Examples of these similarities include the enrichment in Leucine and Phenylalanine. While not taking the total frequency of observed occurrences into account, Leucine was enriched in polyreactive sequences between position 1 and 8 without observable enrichment at later positions (Figure 5 A and -B, -E and -F). Phenylalanine was enriched at positions 1 to 9 in both cases with the greatest difference at the eighth position (Figure 5 A and -B). Polyreactive sequences had enriched V and W at the end position and D at the 9th as well as Y at the 8th and 10th position. They also displayed lower C at the 1st, A at the 1st and 2nd and R at the first 3 positions (but enrichment of R at the 4th to 7th position). While there were some differences between the positional enrichments in polyreactive sequences depending on the task they shared marked similarities.

Taking into account the total occurrence of the feature in the dataset displayed features depleted in polyreactive sequences. When the enrichments were divided by the observed frequency of the feature the resulting heatmaps (Figure 5E and -F) mainly differed from the heatmaps showing total enrichments (Figure 5 A and -B) by highlighting the features which were depleted in polyreactive sequences. Examples of such features included Glutamate (E), Glutamine (Q) and Lysine (K) across sequence positions. These features did not occur as often in the sequences of any class, but were more likely to appear in specific sequences.

Leucine and to lesser extent phenylalanine and tryptophan dominated polyreactive sequences (Figure 5**), the majority of residues were depleted in polyreactive compared to specific sequences at most positions.** Both the enrichment heatmaps and JS divergence plots show that most residues are depleted in polyreactive sequences throughout the sequences. At the positions in the middle of the sequences the enrichments in the polyreactive sequences are dominated by a few residues (Figure 5C and -D), whereas there are more residues characterizing specific sequences. Generally there were fewer residues characterizing polyreactive sequences at each position.

The positions with larger JS divergence contain more information in terms of difference between specific and polyreactive sequences. The JS divergence was higher at positions in the beginning of the 11-mers (i.e., greater divergence between distributions of AAs at these positions) especially at the second and third position, and became smaller towards the end of the sequences (reduced to less than a half/a third of it is at the third position). At the start positions of the 11-mers where the divergence was largest L, F and I among other residues were enriched in polyreactive sequences whereas C, A and R were depleted.

The ends of the CDRH3 sequences were unequally distributed between the classes. Both the JS divergence plots and the heatmaps (Figure 5) indicate that it is more common for specific sequences to contain the start (CAR) signal and a lesser tendency for end motifs (DVW/DYW) to be more common in polyreactive sequences. CAR is the motif characterizing the beginning of the CDRH3 loop, and it typically ends with the motif DYW or DVW. It could appear that there was preferential usage of 11-mers from the end of the CDRH3 among the polyreactive sequences rather than from the beginning.

7.1.5 Logistic Regression Trained on One-Hot Encoding Reaches ~0.9 Macro f1 score

In order to examine whether the AA composition or one-hot encoding of the sequences is enough to predict its degree of polyreactivity, we evaluated the performance of several traditional shallow learning models. Predictions were made based on either the 11-mer AA composition or the one-hot encoded sequence. All models were trained and tested on their ability to make correct predictions based on both encodings separately. Macro f1 scores were recorded for each combination of shallow model, encoding and classification task (Figure 6).

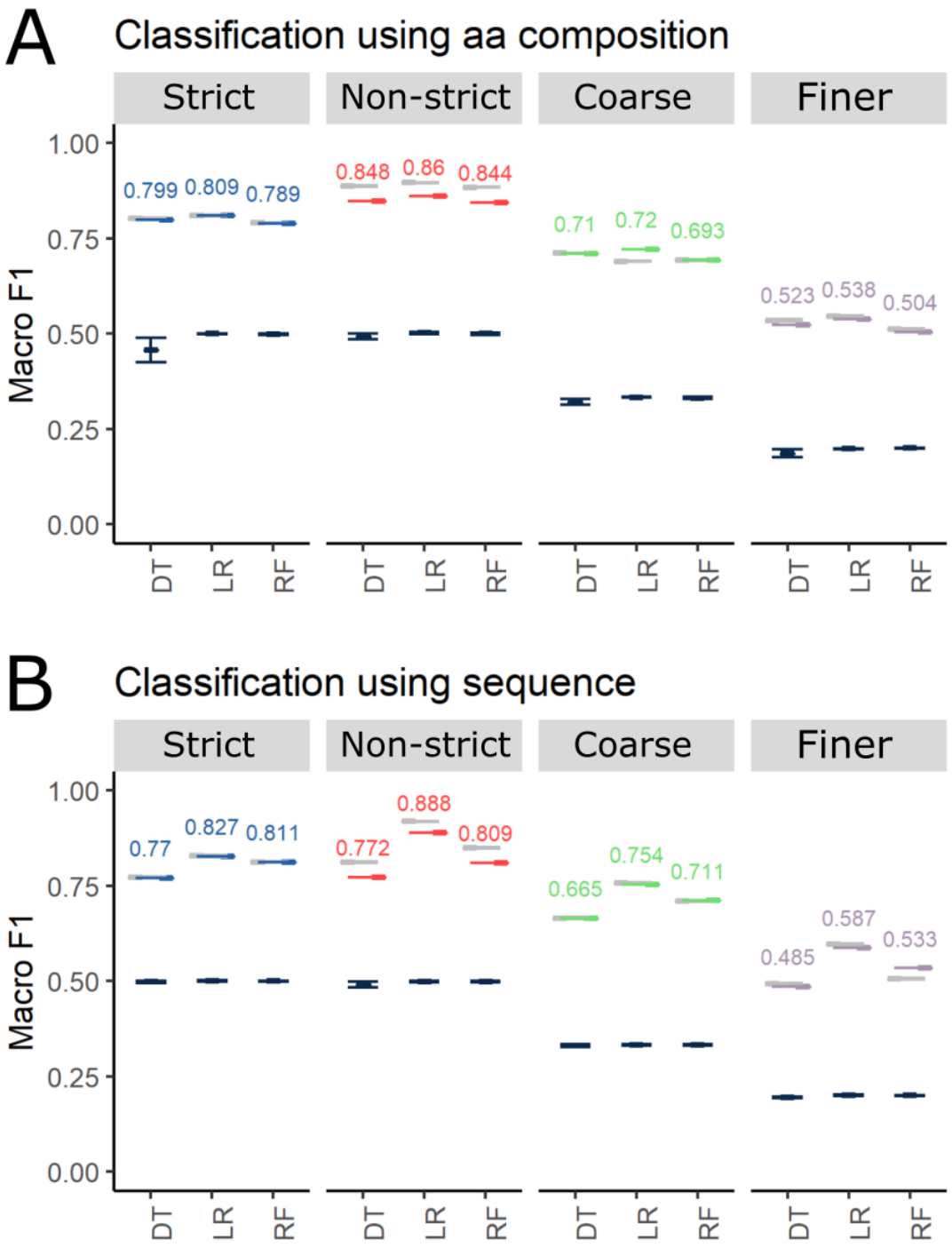


Figure 6: **Positional information allows for higher prediction f1 score than amino acid composition alone** Macro f1 of shallow ML models, using the amino acid composition (A) and one-hot encoded sequences (B), either macro f1 on the validation dataset (gray) or the test dataset (colors). Macro f1 of controls predicting shuffled labels is included in black. Facet columns represent the different data tasks (strict: "strict-polyreactivity", non-strict: "non-strict polyreactivity", coarse: "coarse multiclass", finer: "finer grained polyreactivity").

For all tasks, the models outperformed the randomized controls, which tested macro f1 of the models on randomized class labels (Figure 6**).** When labels were shuffled the models achieved about a 0.5 macro f1 score for both binary classification tasks, about 0.33 for the "coarse multiclass" task and 0.2 for the "finer-grained multiclass" task. These scores were expected due to the chance of guessing the correct class at random for the given number of classes. When the labels were not shuffled the macro f1 of each model were higher than the control by a wide margin (> 0.2).

The macro f1-score between models was depending on the task. The macro f1 was lower for the multiclass tasks and highest for the "non-specific" polyreactivity task. The macro f1 scores were the lowest for task "finer grained multiclass", the task with the most classes. **Increases in macro f1 between architectures were similar in terms of ranking for all tasks.**

F1 scores were highest when classifying the sequences based on one-hot encoding. Adding explicit positional information (one-hot encoding) impacted the performance of the models. The macro f1 scores of the decision trees decreased, but the macro f1 of the random forests increased for all tasks but "non-strict polyreactivity" and for all tasks macro f1 of logistic regression increased. **The increased macro-f1 score of the logistic regression model allowed for improvement of macro f1 score overall.**

For all tasks, logistic regression was the best at predicting polyreactivity in terms of macro f1 score, based on both encodings. Logistic regression achieved the highest macro f1 score on the test data regardless of the encoding used or the task the classes were defined by. However, when AA composition was used, the increase in f1 from that of decision tree and random forest was smaller than when the models were trained on one-hot encoding. Classifying polyreactivity based on AA composition, logistic regression achieved about 0.01 point higher f1 (Figure 6 A). When the given encoding was one-hot sequences, the logistic regression achieved more than 0.01 point higher macro f1 for all tasks. The largest increase was observed for task "non-strict polyreactivity". Logistic regression achieved an increased score of almost 0.09 above the next-best model (Figure 6 B). The highest performing combination of architecture and encoding on all tasks, logistic-regression trained on one-hot encoding, achieved a macro f1 score of 0.83, 0.89, 0.75 and 0.59 on the different tasks.

While the macro f1 scores were somewhat different on the test dataset than they were with cross-validation (Figure 6) the scoring rank between models were similar. With the exception of models trained on AA-composition for task "coarse multiclass" where the ranking was similar to the other tasks on the test data but not the validation data. The decision trees trained on the one-hot encoded sequences performed the worst.

In general, we observed that the AA composition was able to separate the classes with high accuracy for binary classification and improvements of ~0.30 above random using all models for multiclass classification (Figure 6**).** The shallow methods approached 0.90 macro f1, with the complete balanced dataset (the size of the dataset was not reduced beyond balancing classes), for the highest scoring combination of task, encoding and model (Figure 6). **The highest score was observed for logistic regression, predicting "non-specific" polyreactivity based on one-hot encoded sequences.**

7.1.5.1 Performance of binary classification were comparable between classes and confusion occurs most often between adjacent classes in multiclass classification

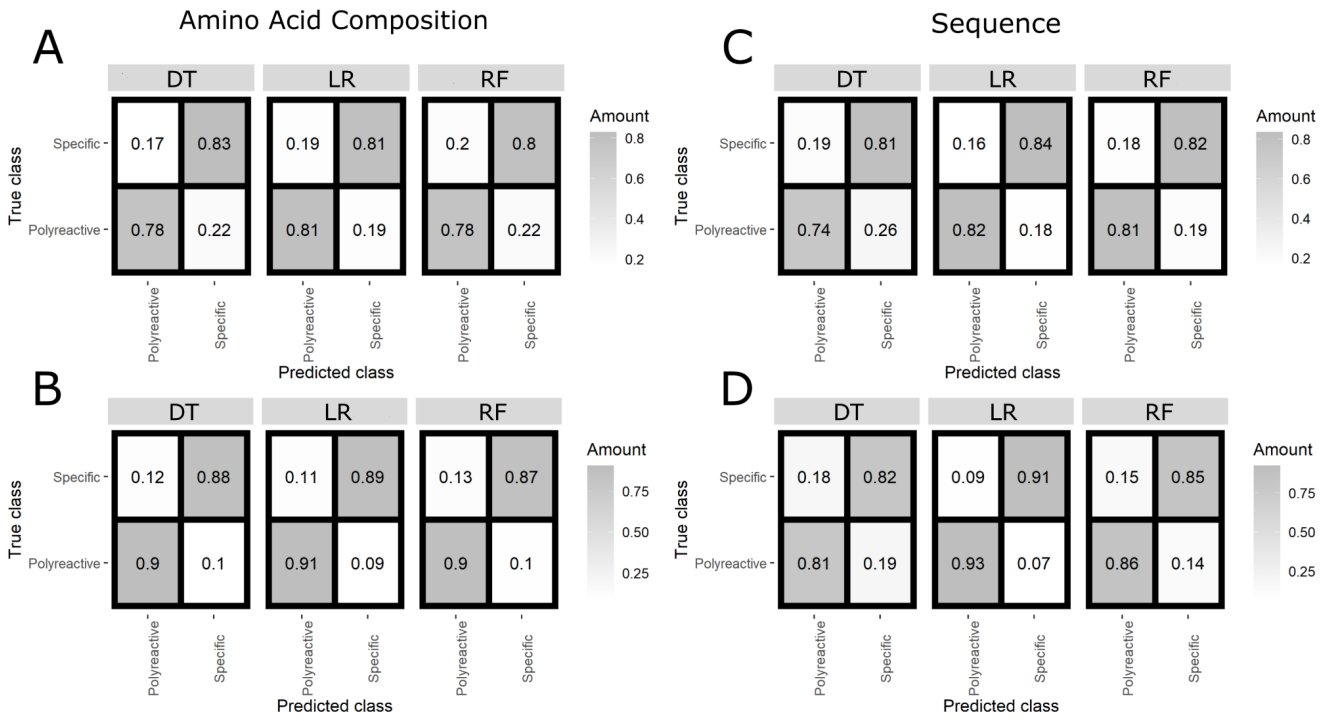


Figure 7: **Shallow Learning models performed comparably well predicting the positive and the negative class.** Confusion matrices showing the number of samples of each class (vertical axis) which were given each predicted label (horizontal axis), divided by the total number of samples in the true class. The numbers reflect the results on the test dataset. A) Confusion matrices for model predicting task "strict polyreactivity" using amino acid composition. B) Confusion matrices for model predicting task "non-strict polyreactivity" using amino acid composition. C) Confusion matrices for model predicting task "strict polyreactivity" using one-hot encoded sequences. D) Confusion matrices for model predicting task "non-strict polyreactivity" using one-hot encoded sequences.

For the binary classification tasks, the ability of all models to identify sequences of both classes was close to equal (Figure 7**).** The random forest, decision tree, and logistic regression (trained on one-hot encoded sequences) were somewhat more successful at predicting specific sequences for datatask "strict polyreactivity" (Figure 7 A and C). The most notable differences were between the fractions of polyreactive sequences that were correctly predicted by the decision trees compared to the specific sequences for the "strict polyreactivity" task. The decision trees were less successful in correctly identifying the polyreactive sequences, with 5% and 7% fewer of the polyreactive sequences being correctly classified when trained on AA composition or one-hot encoding respectively. The logistic regression trained on AA composition was about equally successful at predicting the true labels of both classes. For datatask "non-strict polyreactivity" (Figure 7 B and D) all models except the decision tree trained on one-hot- encoding were better at classifying polyreactive sequences, though the difference was small (differences of ~1-2%).

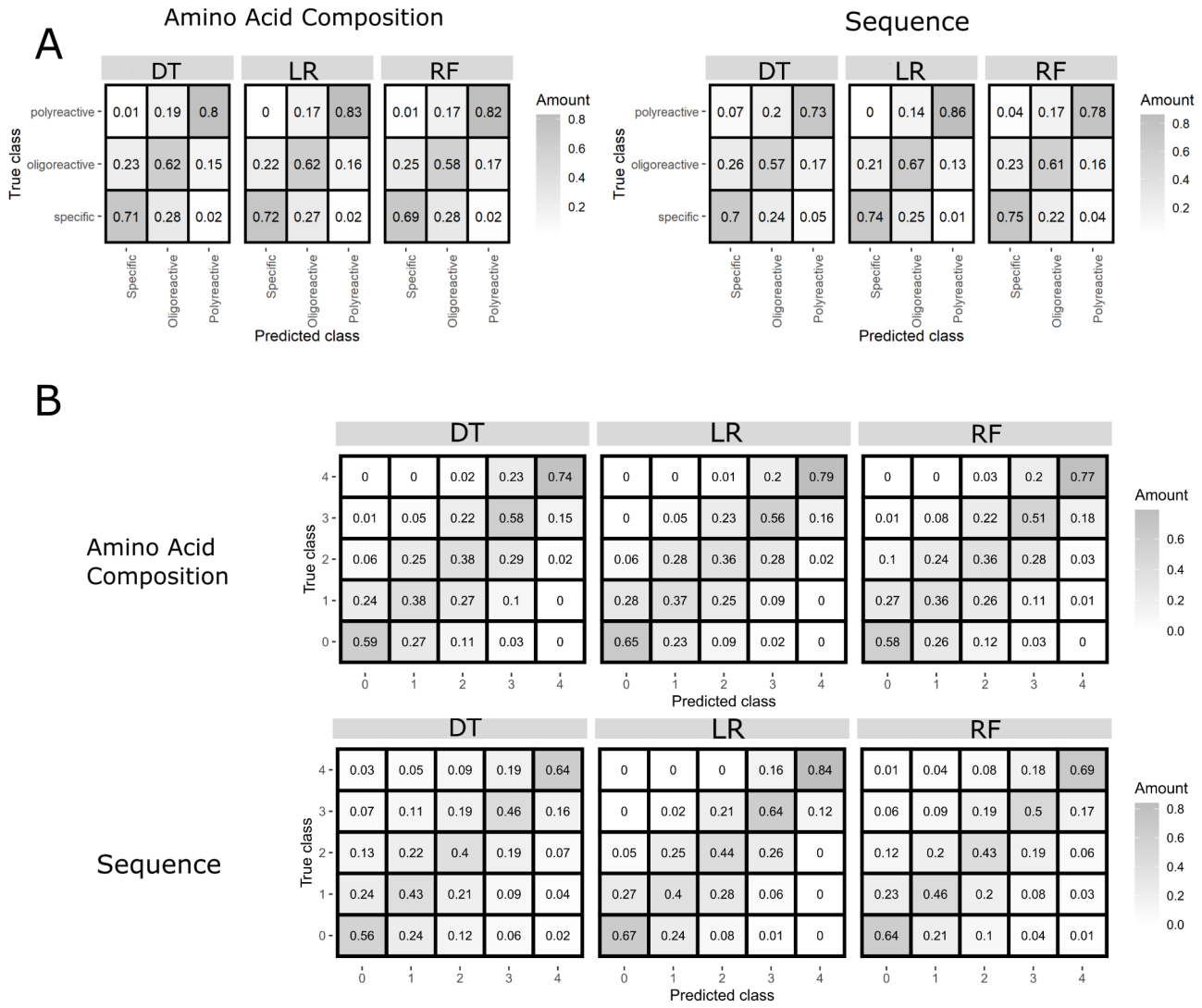


Figure 8: **Trained shallow ML models for multiclass polyreactivity prediction display greater confusion between adjacent classes.** Confusion matrices showing the fraction of samples of each true class (vertical axis) that were given a particular predicted class (horizontal axis) by each model using amino acid composition and sequence. The results reflect prediction of test data labels. A) Confusion matrices for “coarse multiclass”, encoding is specified above. Each tile represents an intersection between the true class of the samples and the predicted label. B) Confusion matrix datatask "finer grained multiclass".

The models trained for the multiclass classification tasks were more successful at separating the classes at each extreme than they were in predicting the sequences belonging to each intermediate class (Figure 8**).** The gray was centered around the diagonal. In other words, most confusion was observed between classes adjacent in terms of the number of antigens bound by the 11-mers, with decreased confusion between classes that were farther apart in terms of the number of antigens bound by the class members. While the logistic regression was better at correctly predicting the sequences belonging to class 1, 3, 4, and 5, the decision tree and the random forest were somewhat better at classifying the sequences belonging to class 2 with one-hot encoded data (Figure 8 B). **For multiclass tasks the classes were easier to separate with increased distance in the number of antigens bound by the sequences within them.**

7.1.6 Using Neural Networks Improves Prediction Score

As high model performance could be reached using traditional shallow models we wanted to know if using a feed-forward artificial neural network (FNN) (1 or more layers) or CNN could further improve macro f1. Improved macro f1 could imply the existence of complex patterns in the sequences that are not captured by the shallow models.

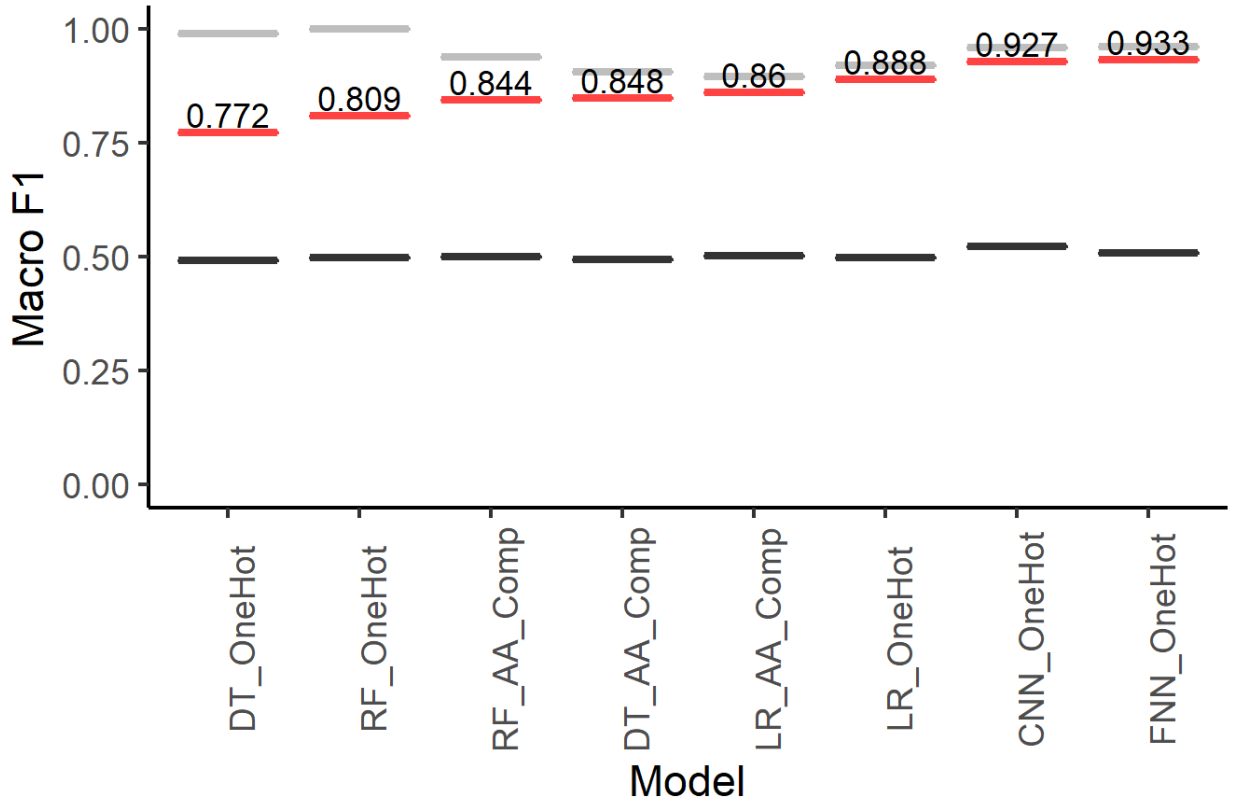


Figure 9: **Neural Networks improve prediction performance** Macro f1 for all models trained and tested on data for "non-strict polyreactivity". The balanced training set was used to train the model and they were tested on the unbalanced independent test set. The gray bars represent macro f1 on the balanced training dataset. The black bars represent mean macrof1 from 10-fold cross-validation on shuffled labels.

Macro f1 score of the FNN and CNN both reached approximately 0.93 on the test dataset (Figure 9**) was higher than logistic regression.** The macro f1 scores represented an improvement in macro f1 score of ~0.04 above logistic regression. Potentially implying the existence of more complex patterns in the data. The single layer neural network (FNN) performed the best with a macro f1 of > 0.93.

7.1.6.1 Neural Networks increased correct prediction of both classes

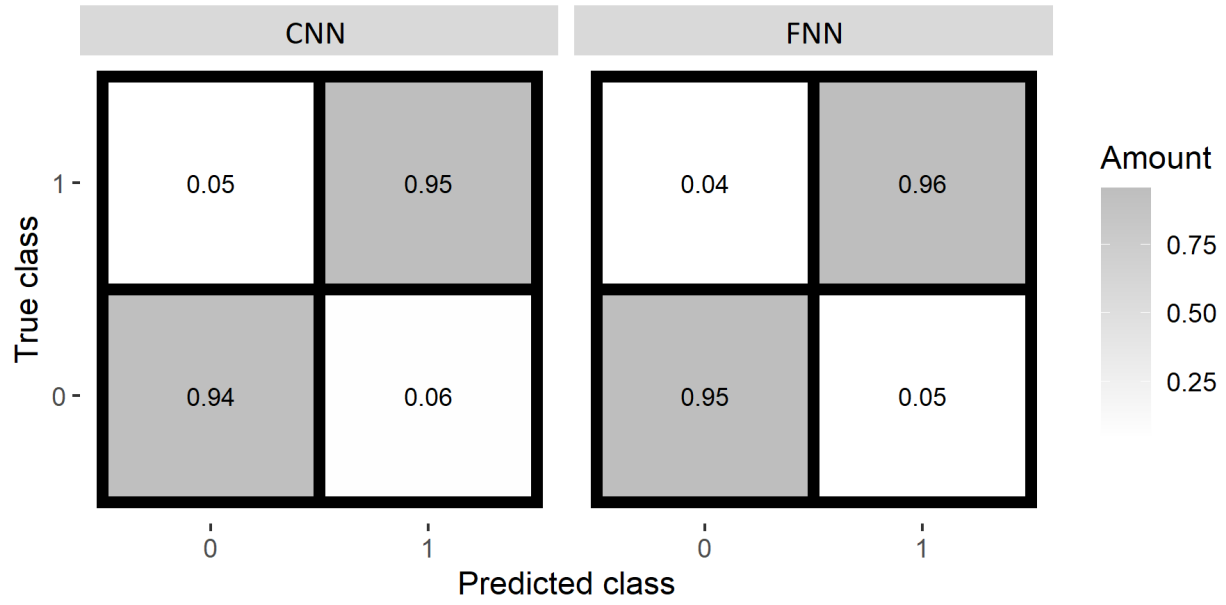


Figure 10: **Neural networks increase correct prediction of both classes defined by "non-strict" polyreactivity.** Confusion matrices showing the number of samples of each class (vertical axis) which were given each predicted label (horizontal axis) by one of the neural networks, divided by the total number of samples in the true class. The numbers reflect the results of prediction on the test dataset.The name of the model is given above the confusion matrices.

Compared to logistic regression, both neural networks increased the fraction of sequences that belonged to each class that were correctly predicted (Figure 7**) (**Figure 10**).** The gain was larger for sequences of the polyreactive class (0.95/0.96 vs 0.91) as opposed to the specific class (0.94/0.95 vs 0.93). Both the CNN and FNN were more successful at identifying polyreactive sequences compared to specific sequences (Figure 10). **The fraction of sequences of each class that were correctly predicted was near equal.**

7.1.6.2 Differences between models were significant

In order to verify that the differences between models were significant, McNemar's test was used, testing all model combinations on the same task. The test specifically looks at whether the marginal differences between two trained models are significant. For two models that are being compared (Model 1 and Model 2) the marginal difference refers to the number of samples that Model 1 got right and Model 2 wrong and vice versa.

$$\text{H0} = \text{The marginal differences are the same/they belong to the same distribution.}$$
$$\text{H1} = \text{There is a real difference in the marginal performance of the models.}$$

For the selected task ("non-strict polyreactivity") all differences were significant at 0.05 (and 0.01) after adjusting for multiple testing. The adjusted p-values were overall low (see Appendix).

For task "strict polyreactivity" all tests were significant except for the comparison between logistic regression trained on AA composition vs random forest trained on one-hot encoded sequences. All models were different from "coarse multiclass", and from "finer grained multiclass" all models differed except for the same models as in "strict polyreactivity" (logistic-regression with AAcomposition and random forest with one-hot encoding).

7.1.7 Interpretation of CNN accuracy with varying kernel sizes reveals that high accuracy can be achieved based on short sequence patterns and accuracy increase with increased pattern length

To understand the type of minimal length patterns that are predictive of polyreactivity, we compared the performance of 1D CNNs with varying numbers of kernels and increasing kernel size (Figure 11**).** To make the results easier to interpret, only one convolutional layer was used. Since only one convolutional layer was used relevant motifs had to be learned using this one layer, which provides a more faithful representation of motif length (Koo and Eddy 2019). The results were generated as part of the parameter search for the CNN model benchmarked above. We looked at the performance when max pooling was used to reduce the filtermap down to one output.

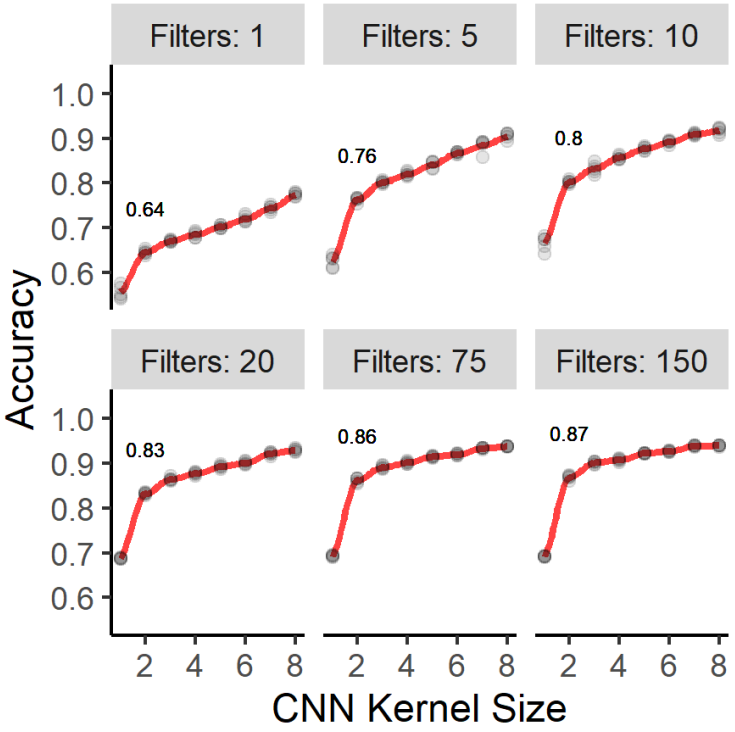


Figure 11: **Increase in CNN kernel size from one to kernel size of two has the most impact on polyreactivity prediction accuracy** Accuracy of the CNN architecture (see methods) depending on the number of filters (facets) and the size of each kernel (x axis), when a max pooling is used after the output of the kernels, to remove positional information.The vertical axis shows the achieved macro f1 score on a validation dataset. The model is here evaluated on five train/validation folds and the results from each are displayed as transparent points whereas the mean point is opaque.

When complete max-pooling was used (i.e. only the largest output from the convolution operation was kept for each combination of filter and sequence), large kernel sizes performed the best (Figure 11). With the largest number of filters (150) a kernel size of 7 reached the highest accuracy. CNN with five filters performed better then with only one, more filters did improve prediction though for large kernels the improvements were marginal. Large sequence patterns can be made up of smaller patterns (f.ex. a filter of size 3 can contain two patterns of size 2), therefore we were particularly interested in which single (+1) increase led to the largest increase in model accuracy. **The large increases were observed when the kernel-size was increased from kernel size one to two (regardless of the number of filters), potentially indicating that the most predictive or most of the predictive patterns are short (size 2).**

7.2 Goal 2: Investigate overestimation of model generalizability due to close similarity between sequences in the training and validation datasets (Data Leakage)

7.2.1 Polyreactivity is predictable from features that are generalizable beyond sequence similarity as measured by Levenshtein Distance

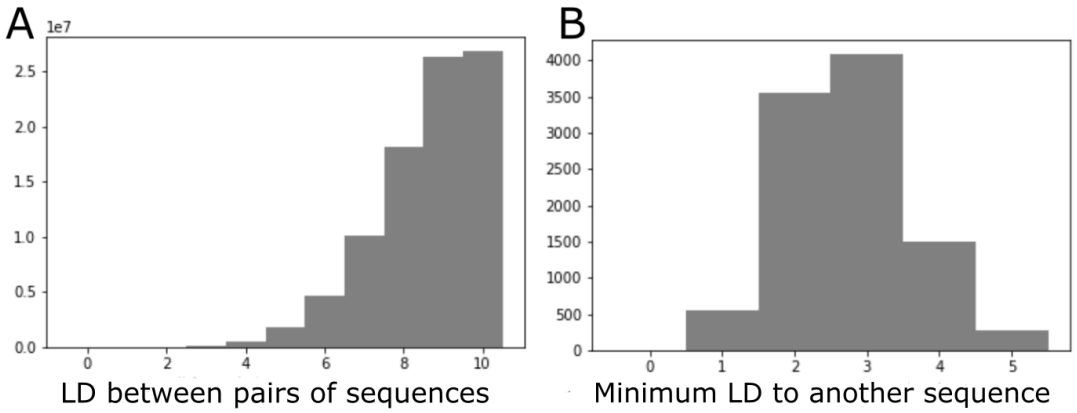


Figure 12**: Minimum distances (measured by LD) from each sequence to another sequence in the 10k dataset tend to be short (<4)** A) Histogram with the distribution of Levenshtein distances (LD) between any pair of 11-mer CDRH3 sequences in a sample of 10K randomly selected sequences. B) Histogram showing the distribution of distance to the closest 11-mer CDRH3 sequence to another sequence in the same sample.

In order to explore the similarity of polyreactive and specific sequences in the data, we calculated the Levenshtein distance (LD) between all pairs of sequences in a selection of 10k sequences. Distributions of LD between all sequences **(**Figure 12 A), displaying the number of sequences as a function of LD, revealed that most distances were LD 8-10 and with few sequences LD <7. Indicating that larger differences are more common with a majority of sequence pairs being separated by LD >8. Most of the sequences had one or more sequences that are similar to them (LD = 2-4) (Figure 12 B). A small subset has a sequence just 1 LD away, in other words they are almost the same sequence, but with one substitution separating them. **Random splitting of sequences between train and test datasets will likely split pairs of sequences of small LD in train and test (i.e. very similar).**

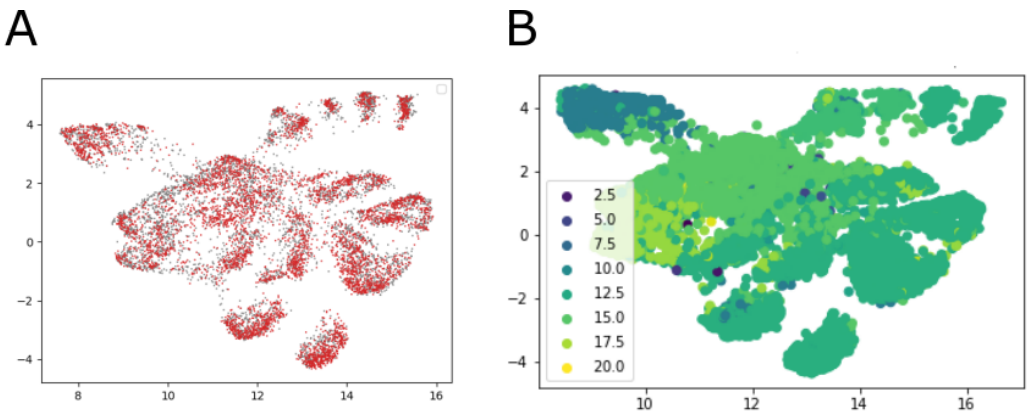


Figure 13: **UMAP clustering of sequences by their LD shows similarity clusters that are not related to the polyreactive class** A) The results of UMAP on Levenshtein distances are colored by class as per datatask "non-strict polyreactivity" (specific/oligo vs polyreactive). Specific sequences are shown in gray and Polyreactive in red. B) Same UMAP colored by clusters achieved through using fclust on hierarchical clustering based on average linkage.

We investigated whether polyreactive sequences can be clustered based on their similarity. UMAP (Uniform Manifold Approximation and Projection) was performed on all the sequences of the reduced dataset using the LD between each pair of sequences as the distance metric (Figure 13). With a selected set of UMAP parameters (see Methods), the UMAP was able to find structure within the data and to visually create clusters of the sequences (Figure 13). Many of the clusters were not completely separated from the rest. The lack of separation was most apparent when looking at the larger cluster/set of clusters in the middle (Figure 13 A). Interestingly, the clusters did not separate the polyreactivity classes, and there were no major clusters where the classes were separated, with samples of both colors represented throughout Figure 13 A. **The UMAP results indicate that similarity between sequences does not necessitate same class, and it is therefore not clear that the performance of trained models on randomly split sequences (**Figure 6**) between train and test could be due to similarity.**

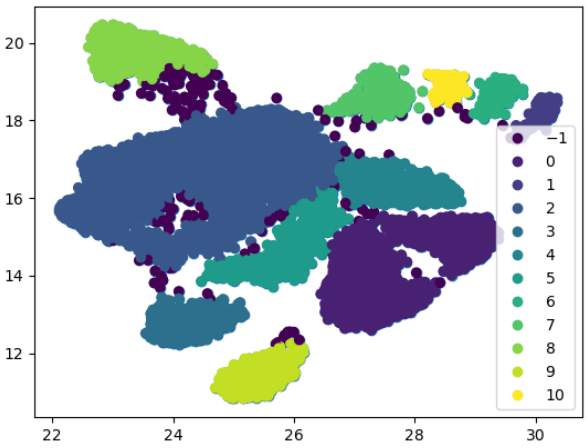


Figure 14**: DBSCAN cluster the data based on UMAP coordinates** The UMAP displayed in Figure 13 where the data points are colored based on clustering with DBSCAN. DBSCAN parameters: Epsilon = .25 Min samples = .35

To test whether high f1 scores were due to data leakage of information between training and test datasets in the form of similarity, we generated training and validation datasets with reduced inter-dataset sequence similarity. We used DBSCAN to annotate the UMAP results into separate clusters (Figure 14), though some of the data points were denoted as noise (class = -1). In order to evaluate the quality of the clustering, we calculated silhouette score for the clustering. Silhouette score is suitable to evaluate clusters when identity is not known *a priori.* It is a measure of intra-cluster tightness and how well separated the clusters are from one another. Silhouette score for the clustering was 0.075 after removal of noise, which is low. A silhouette score can be between -1 and 1, where a score close to 0 would indicate overlapping clusters. However, the main goal was to get train-test separation with increased minimum distance. If that was achieved then overlapping clusters is not a problem per say. **We used the cluster notation to separate clusters into either the train or validation dataset when training a logistic-regression model and compared macro f1 to controls with the same number of sequences in the training and validation data.**

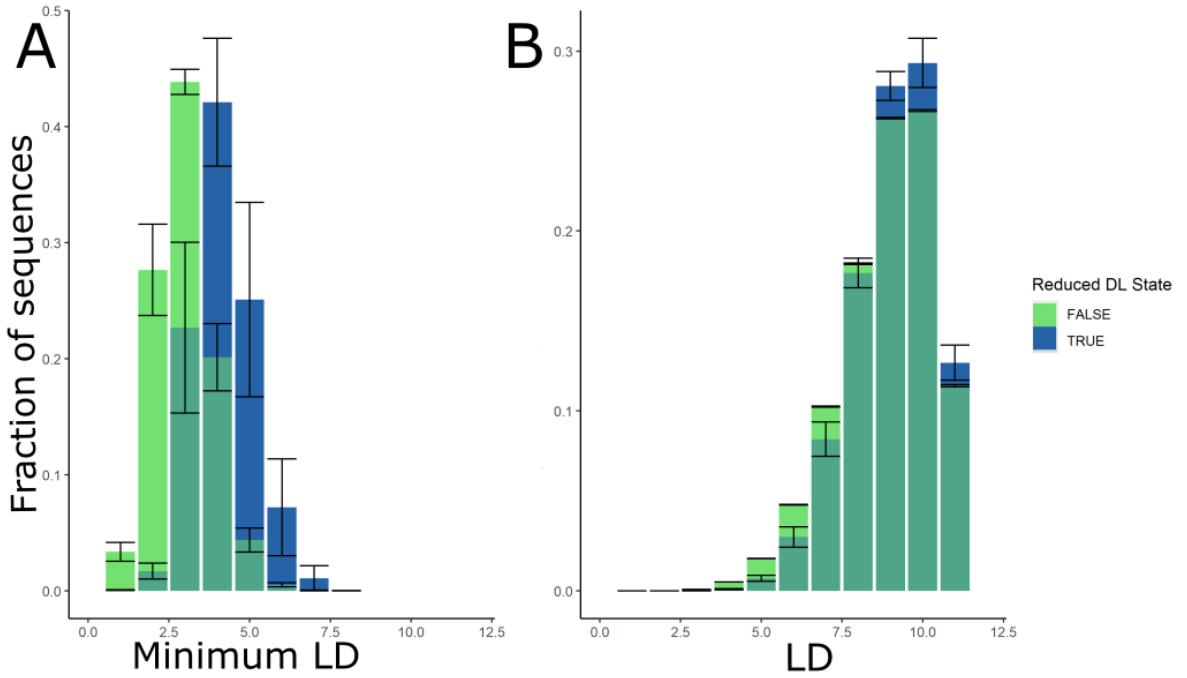


Figure 15: **Separation of sequences based on clusters by LD increased the minimum distance between tre train and validation data.** A) The figure displays the mean fraction of the total number of minimum distances, the given LD (x-axis) represented for the minimum distance distributions with or without separating clusters. When the sequences were divided into training and validation sequences based on clustering it is referred to as the reduced data-leakage (DL) state.B) Show the same as plot A but the distances do not represent minimum LD but rather all LDs between the validation and training data.

Separation of clusters was successful in increasing the distance between train and validation (Figure 15**).** Separating similarity based clusters reduced the proportion of sequences in the validation data which had a minimum LD to the training sequences of 1-3 and increased the proportions that were at least LD = 4 removed from the training sequences. Ensuring that only a small proportion (mean ~ 0.02) of sequences were LD < 3 different from the training sequences and increasing the distance between the train and validation data overall (Figure 15 A and -B). The cluster separation was successful in increasing the minimum distances somewhat with increase in the fraction of sequences 4 and 5 LD apart from ~0.2, but did not lead to large Levenshtein distances (LD >= 5) becoming the majority or even half of the mean minimum distances observed.

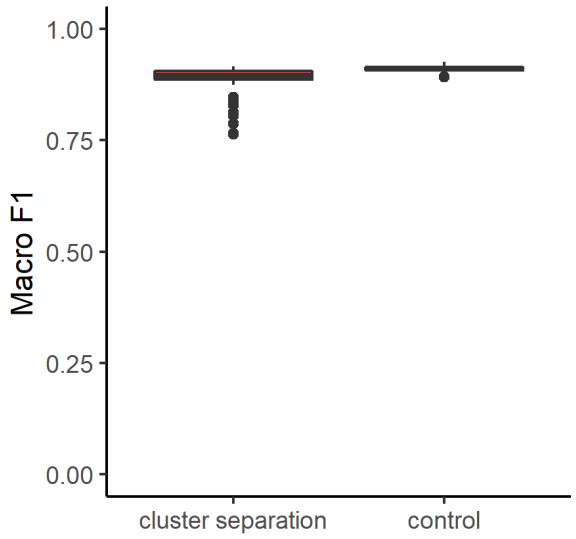


Figure 16**: Logistic Regression models trained on datasets with reduced data leakage keep a high F1 score** Distributions of macro f1 score of logistic regression on "non-strict polyreactivity" binary classification when the train and test data are separated by clusters and control with equal train and test size separated at random.

Clustering the data by similarity and then training and testing a model on different clusters did decrease model macro f1, though it remained above 0.75 (Figure 16**).** The lowest macro f1 was 0.76. In most cases the macro f1 was comparable between the clustered data and the controls. The median macro f1 for the cluster separation was about 0.895 with the median macro f1 being slightly higher for the controls ~0.911 (Figure 16). There is a lower end tail of outlier macro f1 scores from cluster separation. However, even between these the values remain relatively high with some scores being lower.

The above results show the effect of increasing the distances between training and validation sequences so that the overall distance is larger and the amount of validation sequences with a near identical partner in the training data was minimized. It does not directly show how the model behaved on subsets of sequences with various LD from the training sequences.

In order to more directly quantify data leakage, logistic regression and the best performing model overall (FNN) were tested separately on sequences of the validation dataset with a given similarity to any sequence in the training data. Similarity was measured by the LD to the closest sequence in the training dataset, from LD=1 to 6 (Figure 17). As the number of validation sequences with LD > 6 was few, predictions of these were not assessed. A positive control guesser was used to quantify how well a model that predicted class based on similarity would perform in comparison to the trained models.

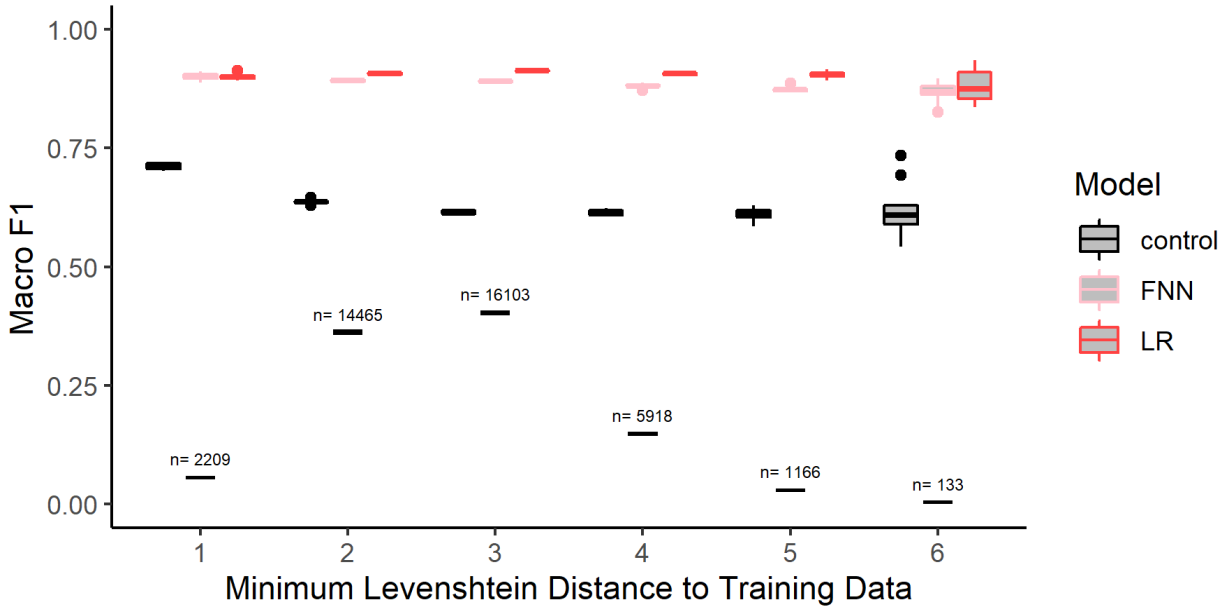


Figure 17**: Trained models perform equally well on similar or dissimilar sequences to sequences in the training dataset** Macro F1 score for validation subgroups defined by minimum Levenshtein distance to a sequence used for training the model. The models used were a logistic regression model (LR) as well as a feed-forward neural net (FNN) trained on data containing one hot encodings for the sequences. The classification is based on "non-strict polyreactivity" (binary classification: specific/oligoreactive vs polyreactive sequences). As a control the macro f1 score when the class of the samples was predicted based on the closest sequences in the training data was included (white). The control shows the performance of a manually defined heuristic "guesser" that outputs the class of the most similar sequence/s in the training dataset. For each subset the guesser took all the sequences in the training data that were closest to the sample in the validation subset it was classifying, and calculated the mean of their class labels. If the mean class was >0.5 the sample was predicted to be polyreactive, if the mean class was <0.5 then it was predicted to be specific. If the mean of the classes was 0.5 however the class was assigned at random.

The prediction accuracy of both models remained high for all minimum distance subsets (Figure 17**).** Logistic regression did not display noticeable decrease in performance as the minimum distance to the training data became larger. We see that there is not much difference in the f1 between validation subsets with short minimum distance to the training data compared to larger distance (LD=6) (Figure 17). We can see that the FNN did perform somewhat worse with increased training distance though the drop was small (0.048). **Trained on the selection of 10k sequences the feed-forward neural network (FNN) underperformed compared to the same model trained on >400k sequences.** Both models outperformed the control. A dip in performance when the distance to train increased from 1 to 2 and from 2 to 3 similar to the control was not observed for either**.** Altogether, our results support that the FNN and LR trained models managed to reach high f1-score by learning patterns that are not relying on close sequence similarity between training and test datasets.

7.3 Goal 3: Look into interpretability of select models

7.3.1 Logistic Regression Coefficients emphasize the role of AA composition

In order to explore interpretability of the logistic regression model, we extracted the logistic regression coefficients to examine how they weighted each AA (AA composition) or combination of position and AA (one-hot). Some interpretability is lost when features are scaled as a feature value of one is not the same as one of said AA or one of said AA at a given position. To get a more direct interpretation of the effect of the features on prediction, the models were trained without scaling the coefficients and the macro f1 was confirmed to be the same.

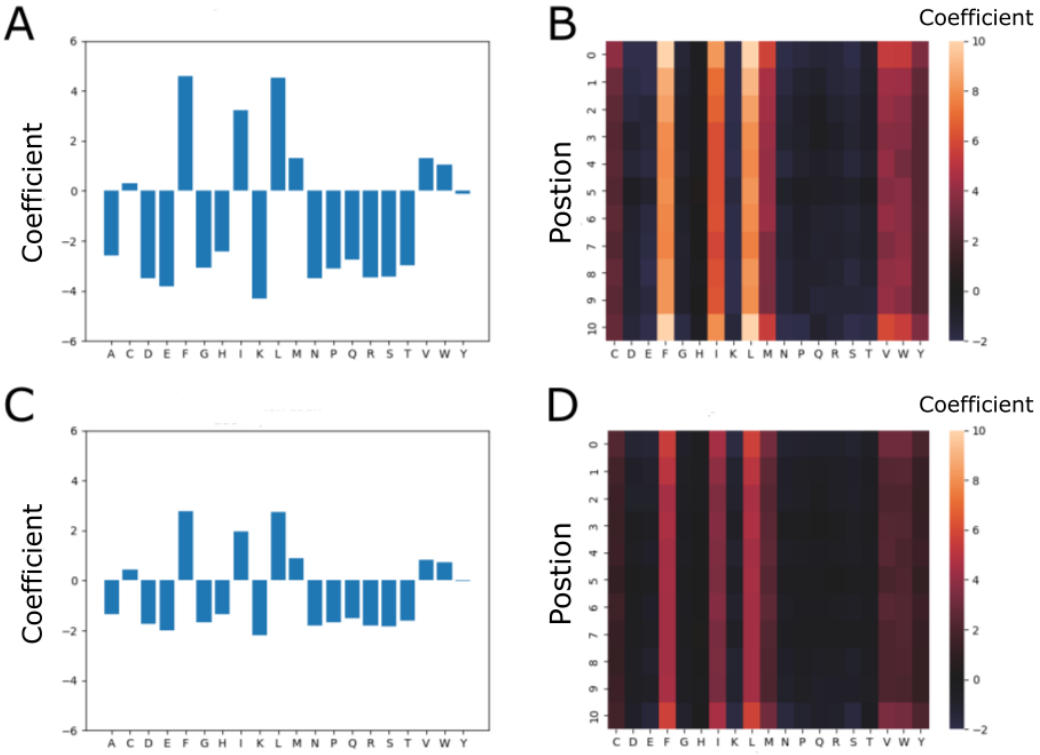


Figure 18: **Logistic Regression coefficients trained on data from task "non-strict polyreactivity"**
A) Model coefficients from logistic regression model trained on amino acid composition when the features are unscaled (task = "non-strict polyreactivity"). Macro f1 was inspected to confirm that it is the same scaled or unscaled . B) Model coefficients for models trained on one-hot encoded sequences. C and D) Similar to A and B but for datatask "strict polyreactivity".

A few AAs contributed positively to the output. The features Cysteine, Phenylalanine, Isoleucine, Leucine, Methionine, Valine and Tryptophan were given positive coefficients. The coefficients of the Cysteine and Tyrosine were the smallest in absolute magnitude and < 0.5 for both tasks (Figure 18 A). In particular the coefficient associated with Phenylalanine is the largest positive coefficient followed by Leucine. Positive coefficients represent positive correlation between that feature and increased probability of making a positive class prediction. With logistic regression the coefficient multiplied by the feature value directly contributes to the logit (or log of the odds). The logit is the sum of weighted features for a particular input prior to transformation into probabilities. The coefficients of the model show what the impact of one of each AA in the sequence or having said AA at a specific position would be on the logit. As an example, for each Valine in the sequence the logit would increase by about 1 ("non-strict" polyreactivity) or by exp(1) to the odds ratio (Christoph Molnar 2022b).

The coefficients of the model trained on one-hot sequences show that most of the variation is between different AAs (Figure 18 B), though there were some differences in the magnitude of the coefficients between positions. Larger variation between different AAs at the same position compared to little variation between different positions for the same AA shows that the effect of the amino is somewhat consistent across positions.

The coefficients of the model trained on "strict polyreactivity" were similar to the ones trained on "non-strict" polyreactivity except when it came to magnitude. The relationship between the magnitude of coefficients for different features was similar. However, the coefficients were smaller in the logistic regression model trained on "strict polyreactivity".

6.3.1 Integrated Gradients Enable Interpretability of FNN

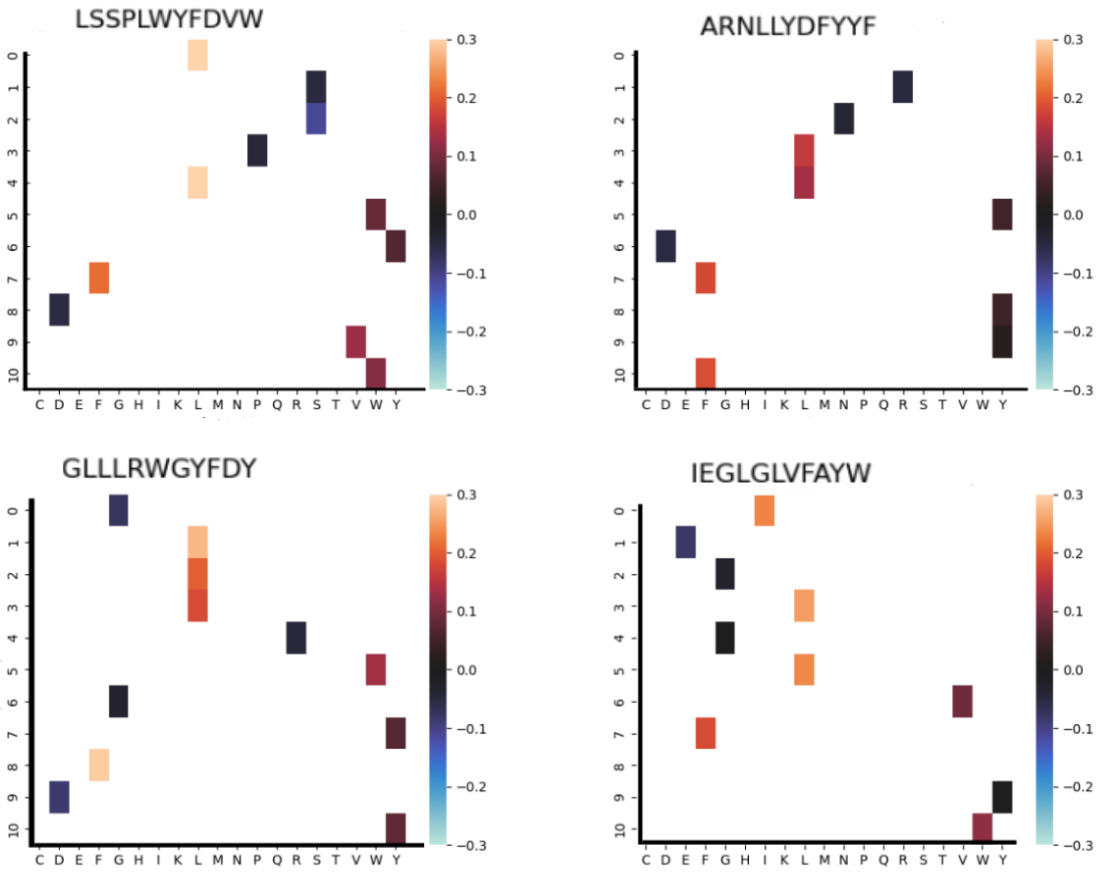


Figure 19**: Integrated gradients from four polyreactive sequences.** One heatmap represents a unique sequence. Red and yellow tones suggest a positive correlation with making a positive classification, blue negatively correlates. The placement of the sequences in the grid is arbitrary and does not reflect relationships between them. The white background signifies features that are not present in the sequence.

The integrated gradients, feature attributions indicating the effect of each feature upon prediction (see Methods), calculated for the polyreactive sequences agree with the logistic regression coefficients (Figure 19**).** The purpose of integrated gradients is to reveal what features the model focuses on and how the feature contribute to the output. For the sequence "LSSPLWYFDVW", the largest attributions were given to the Leucines and then to the Phenylalanine. The Valine, Tryptophan and Tyrosine were also given positive attributions and the Serine at the 3rd position was assigned a negative attribution. **Features that are given positive coefficients from logistic regression were assigned positive attributions, Leucine and Phenylalanine showed larger positive impact as was observed in** Figure 18**.**

In order to assess the correlation between the integrated gradients and logistic regression coefficients 5000 polyreactive sequences were chosen and used to get an average of the integrated gradients across all features.

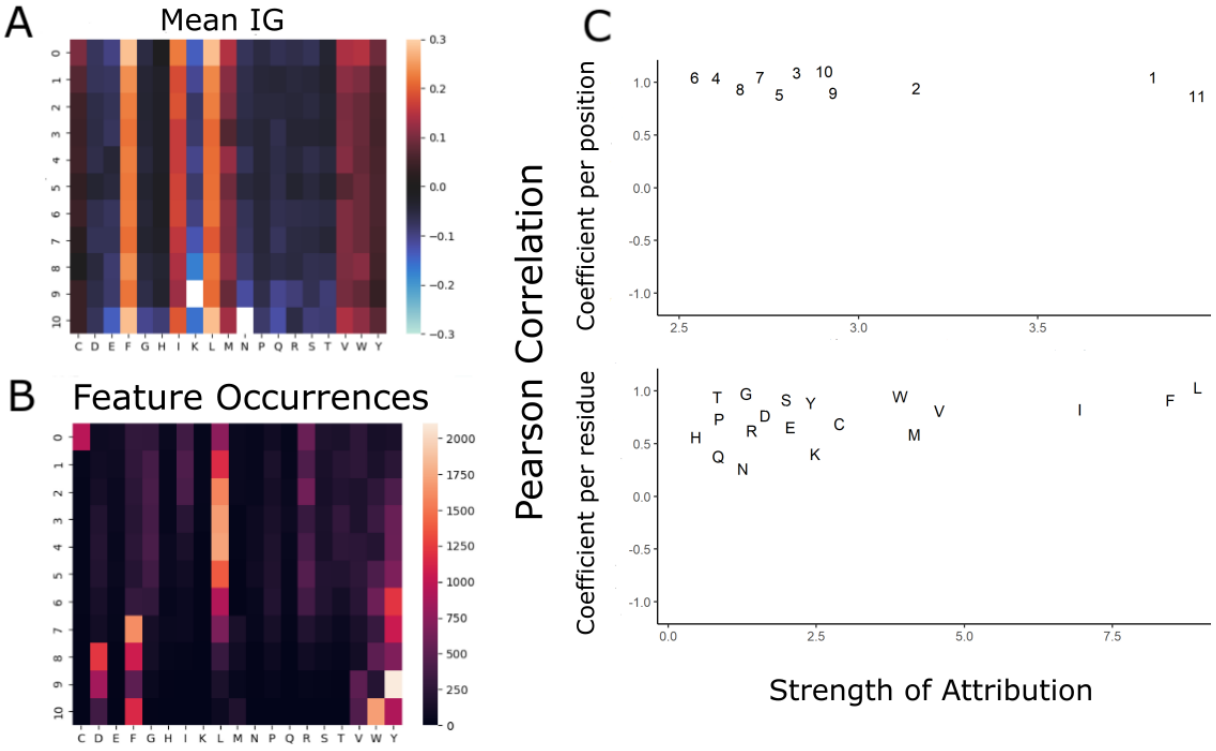


Figure 20: **Average integrated gradients for 5k polyreactive sequences (as defined by "non-strict polyreactivity") resemble logistic regression coefficients.** A) Average of 5k integrated gradients. Warm colors show features contributing positively towards classifying the sequence as polyreactive whereas cool colors contribute negatively. The horizontal axis shows the amino acids and the vertical axis represents each of the 11 positions in the sequences. The range of the color palette was set manually. White squares represent features that were never encountered. B) How often the feature occurred in the 5k sequences. C) Top: Pearson correlation coefficient between amino acids at the same position between integrated gradients and logistic regression coefficients on unscaled data (numbers 1-11 represent positions). Bottom: Correlation between integrated gradients and coefficients at differing positions for the same amino acid. Strength of Attribution: Mean absolute logistic regression coefficients.

The mean attributions reflected in the integrated gradients are in agreement with results from logistic regression. Like with logistic regression there were more observed variation between the AAs then between the same AA at different positions in terms of mean attribution. Integrated gradients corresponded to coefficients, the pearson correlation between the coefficients and the mean of the integrated gradients was close to 1 (minimum pearson correlation ~ 0.977) for each position (Figure 20 C above). The correlation between coefficients and mean gradients of different positions with the same AA were positive (minimum Pearson correlation ~ 0.321) (maximum Pearson correlation ~ 0.980) (Figure 20 C below). The correlation was lower within residues where the strength of the attribution was low. This included amongst others Asparagine and Lysine which were never observed at specific positions when calculating mean of IGs.

The logistic regression model is considered interpretable, and can be interpreted through examination of model coefficients. Integrated gradients assigned feature attributions that were in line with previously observed class differences and correlated with logistic regression coefficients. An interesting observation regarding the attributions given to the AAs is that they look to correspond well with their mean interaction potential according to the inter-residue potentials used in the simulation (Miyazawa and Jernigan 1996).

7.3.2 Conserved Start and End Motifs of CDRH3 did not Bias Model Ranking

The CDRH3 end motifs, the pattern marking the beginning and end of the CDRH3, are not distributed evenly between the classes. This could potentially be problematic because these are conserved motifs where there is little sequence variation. A concern would be whether the uneven distribution of the motifs could bias the ranking of the models. To evaluate to what degree the conserved motifs at the end of the CDR-H3 sequences might bias the ranking of the models, a dataset was created where the end-motifs (CAR and DVW/DYW) were balanced between classes.

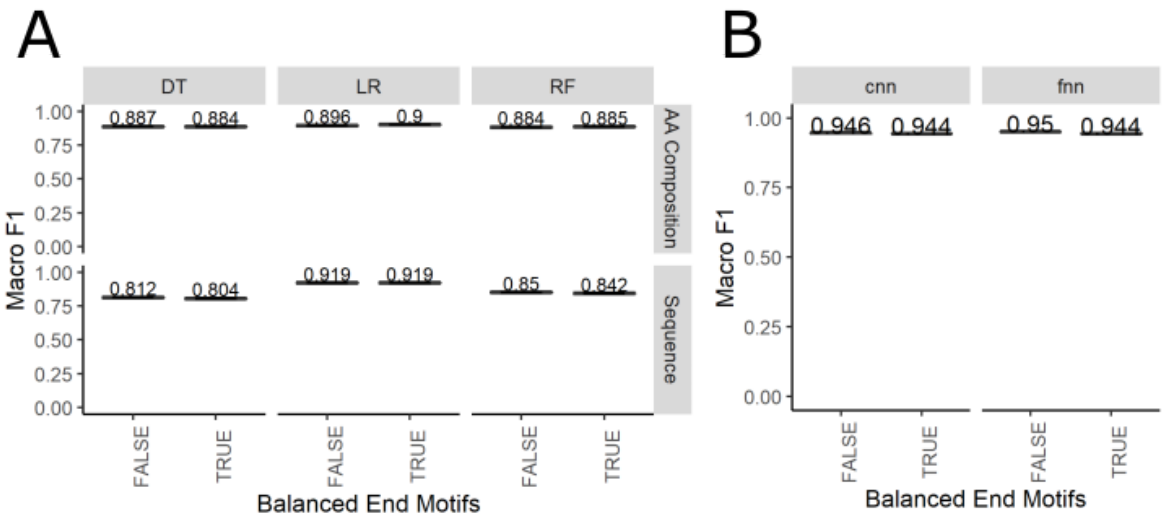


Figure 21**: Balancing end-motifs does not change ranking of methods**
In order to test whether common end motifs for CDRH3 bias the ranking, end motifs were balanced per the classes. So that an equal amount of sequences from both classes would begin with CAR, AR etc. A) Results of using traditional shallow models on the sequences with balanced ends. B) Results of neural networks.

Comparing the accuracies of the model on the full "non-strict" polyreactivity dataset compared to the dataset with ends balanced, the macro f1 scores were similar. There were some differences between the macro f1 on the balanced vs full dataset, however the ranking of the models remained largely the same. The differences were in the ranking of the random forest vs decision trees trained on AA composition and the neural networks performed similarly on the balanced data. The influence of the end motifs on prediction is minimal and did not influence the ranking of the model to a large extent (Figure 21). We also reran part of the CNN kernel analysis with balanced ends (see Appendix).

8 Discussion

8.1 Goal 1: Comparing combinations of model architecture and sequence encoding for creation of supervised ML models to predict polyreactivity

8.1.1 Ability to predict polyreactivity depending on datatask

We classified polyreactivity in four different ways in order to investigate how the models would behave depending on different definitions of polyreactivity. The purpose of defining different scenarios was primarily to see if the classes would be predictable or if it would make a large difference to the ranking of the models. This was motivated by the lack of a universal framework for defining polyreactivity and the uncertainty, pointed out in literature, that binding to few antigens is driven by similar properties as binding to many (Boughter et al. 2020).

In all scenarios polyreactivity was predictable, as seen by the substantial increase in macro f1 compared to randomized controls (Figure 6). Between the various tasks, the rankings of model macro f1 were largely similar. That the binary classification tasks were the easiest to predict indicates that dividing the data up into more classes did not help the models learn but rather made the task more difficult due to the increased number of possible labels. This was further demonstrated by the multiclass confusion matrices (Figure 8). While we did not conduct further investigation whether there were antibodies binding few antigens in a distinct fashion (which could further elucidate to what extent such could be the case). In general, comparing a class with higher polyreactivity compared to another class with lower polyreactivity shows the same trends in sequence enrichments (Figure 5) and coefficients from logistic regression (Figure 18). Which seem to suggest that the classes are not entirely distinct in character with completely different binding rules, but rather appear to divide a continuum going from “less” to “more” polyreactive.

Here, we formalized polyreactivity as classification inspired by previous studies into polyreactivity. In the future, within the range of the class definitions that we tested, degree of polyreactivity could potentially be described as a continuum of more classes of degrees, or alternatively as a regression problem. For instance, Harvey et al. predicted a quantitative polyreactivity measurement, called the “polyspecific reagent binding” (PSR) (Harvey et al. 2022), which is a fluorescent measurement of the amount of different denatured antigens bound by a single antibody, without knowing exactly which antigen variants they bound to. Binary prediction of polyreactivity however can still provide insight into how to predict polyreactivity and help create a test for polyreactivity.

Viewing polyreactivity as a continuum, defining a threshold for testing whether an antibody is polyreactive would be more of a question of what level of polyreactivity is acceptable for the intended use. However, this task can be answered using a classification approach. While there is no universal framework describing the degree of polyreactivity, by comparing the tested antibodies to an antibody with known specificity using the same essay could set a threshold for acceptable levels of polyreactivity. However, some potential limitations to this approach have been highlighted by a study showing that for issues of developability, clusters of traits might be more meaningful than using hard cutoff levels for individual developability parameters (Jain et al. 2017). Still, a test for acceptable levels of polyreactivity could potentially be used to warn off a potential problem alongside other parameters (Jain et al. 2017).

8.1.2 Between a selected list of model architectures, how do these models compare in their ability to correctly classify antibody CDRH3 sequences based on observed levels of "polyreactivity"?

Here, several supervised ML approaches were benchmarked for predicting polyreactivity based on macro f1 score (Figure 6 and Figure 9). Three selected shallow learning models were tested using two types of sequence encodings (AA composition and one-hot encoding of the CDRH3 11-mer sequences) (Figure 6). We also benchmarked two neural networks for predicting "non-strict polyreactivity" based on one-hot encoding (Figure 9).

Of the traditional shallow models, the logistic regression reached the highest macro f1-scores (Figure 6) on all tasks. The increase in macro-f1 score indicates that between the tested architectures (decision tree, logistic regression and random forest), with the selected parameters, the logistic regression models best captured relationships between classes. The results imply that within the range of values that were used to separate the classes, logistic regression was the best model for capturing the relationships between the sequence features and polyreactivity.

The tree-based architectures (decision tree and random forest) displayed high variance in performance between the training and test sequences when trained on one-hot encoded sequences, indicative of overfitting (Figure 9). Decision tree models are often prone to overfitting, as was observed with the high variance between the training f1 and test f1 for the decision trees (and random forests) trained with positional information. When the decision tree was trained on AA composition, tuning the length of the tree was sufficient to avoid overfitting. When positional information was included, no decision tree model could be identified without either high bias or high variance. Random forests are often more robust to overfitting as they are ensemble models that rely on majority voting from several long decision trees. However, we see that they displayed high variation between training and test macro f1. For task "non-strict" polyreactivity the test macro f1 was lower when positional information was provided while the test accuracy was high, which might be indicative of overfitting (Figure 9). It is possible that further hyperparameter tuning could improve the performances of the models. Tuning parameters that would introduce more variation between the individual trees or reduce their size might help improve the ability of the model to generalize to unseen data. However, from the results it looks like the method by which the tree-based architectures fit to the data was not optimal.

Our results support using relatively complex models, in particular feed-forward neural networks (with one hidden layer), for prediction of polyreactivity when a substantial amount of data is available for training. When trained on training splits from the complete "non-strict" polyreactivity the neural networks achieved increased macro f1 score on the test data compared to logistic regression (Figure 9). However, the feed-forward neural network (FNN) performed worse than the logistic regression model when trained on the smaller subset of 10k sequences (for assessment of data leakage) (Figure 17). The low performance of the network when trained on fewer samples may indicate that if the available training dataset is smaller (not larger than 5k sequences of each class), training a model based on an architecture with limited complexity such as logistic regression could yield a more accurate model. That more data is needed to train neural networks than what is needed to train a logistic regression model is expected as neural networks are generally considered more difficult to train. Since `Absolut!` data reflect a simplification of antibody-antigen binding, we would expect that our data is less complex than experimental data and thus even more experimental data might be necessary to successfully train neural networks with the same performance as in simulated datasets.

In future studies the finding that neural networks could potentially outperform logistic regression will become more relevant as more experimental data become available. Most research looking at polyreactivity to date has been conducted on smaller datasets of 1000 sequences or less (Boughter et al. 2020). However, the recent advances in experimentally assessing polyreactivity could enable creation of larger polyreactivity datasets in the future suitable for training neural networks. Indeed, in a recent preprint Harvey et al. (Harvey et al. 2022) already report creating such a dataset from their newly invented approach.

The McNemar's test showed that most of the observed differences between models were significant, however that does not necessarily mean that they are interesting. At a large sample size (>100k) the power of the test is such that any difference is likely to lead to a rejection of H0. That is, we can assume that the power of the model will be high. The power is described by $1-\beta$, where β is the probability of not rejecting H0 when there is an effect. Or rather, failing to report that there is an effect when such is the case. Therefore, some of the results, while significant, are not so interesting in practice (Breur 2016). The differences between the decision tree and random forest trained on amino acid composition or the two neural networks (difference < 1%) would perhaps not be very interesting.

The test directly compares pretrained models. For the results to be generalized further we must assume that the variation in performance due to training has to be small (Dietterich 1998). The results produced by cross-validation demonstrated low variation between splits (Supplementary Table5), compared to differences in macro f1. However, the variation was higher for the neural networks and as there was little difference in performance we may not be certain that there is a difference between the neural network architectures. However, as the difference was so small to begin with, the question of whether they are significantly different is likely not of much interest.

8.1.3 Does providing explicit positional information improve the models ability to predict levels of polyreactivity?

During this project we have trained ML models on different encodings with varying levels of complexity and using architectures that are able to capture different types of features. We have compared the performance of logistic regression, decision tree and random forests on both AA composition and one-hot encoded sequence in terms of macro f1 scores. The testing of these models has been conducted in part for the purpose of investigating what features are predictive of polyreactivity in our data.

Explicit positional information did give the logistic regression model a performance boost on all tasks and lead to increased overall scores (as the logistic regression models trained on one-hot encoded sequence performed better than the other models on each task). Thus it could be reasonable to assume that positional information does increase prediction performance. The logistic regression models trained on one-hot encoding were significantly different to the ones trained on AA composition in terms of the size of marginal differences ($\alpha = 0.05$). Low variance, similar to the logistic regression models trained on AA composition, indicates that adding positional information did not lead to an overly complicated function overfitting to the training data (Figure 9). Our results indicate that the effect of each AA on prediction of polyreactivity can depend on the position in the CDRH3 sequence as the sequence encoding leads to better prediction than just AA composition.

8.1.4 Is polyreactivity prediction improved by using models that capture nonlinear relationships between features?

In evaluating the performance of the neural networks compared to logistic regression we also evaluated whether there could be any nonlinear interactions between individual sequence features determining polyreactivity (as per the task “non-strict polyreactivity). A limitation of logistic regression is that it assumes that, if there are any interactions between the features, logistic regression requires that the relationships are manually provided as independent features (interaction terms) (Christoph Molnar 2022c). Architectures like neural networks on the other hand are capable of capturing interactions without manual help. We also observed that without introducing non-linearity to the CNN model the macro f1 on the validation data was noticeably reduced, and more comparable with logistic regression (Supplementary Figure 4). This could further support that the increase in performance was due to nonlinearity. The

increased performance of the neural networks compared to logistic regression (and the CNN without nonlinearity) supports that the rules underlying polyreactivity (in the `Absolut!` dataset) involve interactions between features in the one-hot encoded sequences.

The tree-based architectures (decision tree and random forest) are also able to capture non-linear relationships between the features. However the way that they model these relationships are different. They rely on a top-down approach that progressively separates the data into smaller groups. As previously mentioned, the tree based models displayed a tendency towards high variance when trained on one-hot encoding indicative of learning the bias of the training data. Thus, even though the tree-based models can account for feature interactions, the way they model these relationships was likely not optimal for the task (at least with the parameters used).

8.1.5 In the interest of identifying minimal pattern length predictive of polyreactivity, how does prediction accuracy change in response to increase in the possible length of recognized patterns?

Accuracy of CNNs with one convolutional layer can reveal presence of predictive patterns with a given max length depending on kernel size. To get an idea of the minimal length of sequence patterns predictive of polyreactivity we examined how the accuracy of CNNs with a single convolutional layer changed with increased kernel size. The CNNs included a max pooling layer that reduced the resulting feature map from each filter of the previous convolutional layer to a single output representing one sequence pattern. Thus, all positional information not included in this output was removed, and it guaranteed that each feature map represented a pattern no longer than the kernel size. As mentioned in the introduction, convolutional layers can be seen as feature extractors (Raschka and Mirjalili 2019), detecting patterns in the original input which then become the input features of the subsequent layers. The output of the max pooling layer is fed to the output layer which weights each pattern and transforms the output into class probabilities similar to a logistic regression model. We could imagine the model as similar to a logistic regression model predicting polyreactivity with patterns of max size equivalent to the kernel size as input features.

High accuracy with longer kernel size is not necessarily due to longer patterns. While a large kernel size enables capturing longer patterns, the model could still potentially be making predictions based on shorter patterns. Longer patterns might also be made up of combinations of shorter patterns, such as an observed pattern of size 3 being made up of 2 patterns of size 2 or even longer non-contiguous patterns being made up of several short patterns. Without further investigations into the kernel weights we cannot be sure what each filter in the CNN represents. Due to the uncertainty in what the kernels represent and the possibility of longer kernels finding shorter patterns we are mainly interested in accuracy increase with each increase in kernel size.

Short sequence patterns were predictive of polyreactivity. We observed large jumps in accuracy with increase in kernel size from one to two, even for models with 1 - 5 filters (Figure 11). These jumps in accuracy demonstrate that, even when the number of detectable patterns were reduced, patterns of size 2 were predictive of polyreactivity. As the number of filters increased, the increase in accuracy with larger kernels were reduced, potentially indicating that most predictive patterns in the data were short or could be expressed as short patterns. Though, as the length of the sequences is limited to 11 there could be tradeoffs where the presence of some patterns informs the model of the absence of another pattern. When the number of detectable patterns are reduced the risk of such an effect is low. Since larger increases in accuracy were detected with few filters it seems likely that the observed effect was due to short patterns which are themselves predictive and that there are short patterns predictive of polyreactivity.

We have not looked into what patterns are predictive. The results suggest that there are short patterns but confirmation and identification of specific predictive k-mers (sequence patterns of size k) require further investigation. To further evaluate to what degree short patterns are predictive of polyreactivity the weights of the CNNs with kernel size 2 could be extracted and studied further. Alternatively, a list of k-mers could be selected with max k of 2 and used to train a model on encoding specifying whether a sequence contains said k-mer. Predictive patterns could be identified through feature selection. Such an analysis could be particularly interesting for comparing the patterns with AA composition to investigate to what extent they reflect positional dependencies.

CNN models predicting polyreactivity based on patterns of max length 2 reached high accuracy. With the largest number of filters the accuracy reached 87% (Figure 11) for the model with kernel size of 2. We might expect a model making predictions based on explicit 2-mers would reach comparable accuracy on our data or higher. Longer kernel size led to higher accuracy regardless of filter number. This could be potentially indicative of long range dependencies in the sequences (Robert et al. 2022). However, a weakness with the analysis of the CNN architectures with max pooling, is that the models cannot know how many times the same pattern appears in a sequence. A model basing prediction not only on the presence of 2-mers, but also frequency within the sequence could potentially further break down longer patterns and reach even higher accuracy. To be certain of this however, would require testing. Using short k-mers is desirable as using long k-mers lead to large numbers of potential motifs rarely seen in the data, causing trouble due to high computational cost, and risking certain motifs very rarely or never being observed during training (Ostrovsky-Berman et al. 2021).

8.2 Goal 2: Investigate overestimation of model generalizability due to close similarity between sequences in the training and validation datasets (Data Leakage)

8.2.1 Is the predictive power of the models dependent on close similarity to already observed sequences in terms of LD, and could the models learn patterns that are not based on close sequence similarity?

As for whether the performance of the models is due to data leakage, in terms of similar sequences in the train and test data, our results do not seem to support that. When training and validation sequences were separated using clustering, the number of sequences in the validation which closely resembled the training sequences was reduced. The cluster separation also caused the minimum and total LD between the training and validation data to increase (Figure 15). Still, the macro f1 of logistic regression remained high (Figure 16). Thus, the logistic regression model was able to separate classes in the validation set without having seen highly similar sequences in the training dataset. Though the change in distance was mainly a reduction of sequences that were 1-3 LD apart from the nearest training sequences.

The analysis of the performance of both the logistic regression and FNN on subsets of sequences with varying LD gave more insight into the generalizability of the models. Evaluation of model performances on subsets of variable distance to the training sequences showed that both tested models reached high macro f1 scores on all subsets and outcompeted the control (Figure 17) indicating that the models are not making predictions based on close sequence similarity (LD). That the models outperformed the control in classifying sequences that resemble the ones seen prior (LD = 1), indicates further that they did not get confused due to similarity (Mason et al. 2021; Robert et al. 2022). Coupled with the high macro f1 on sequences that were dissimilar from the training sequences (LD = 5-6) this result implies that the models were able to learn generalizable features not depending on close sequence similarity (LD).

The training conditions were different in the minimum distance control compared to the conditions the final models were tested on, as only 10k training sequences were used. When a smaller number of sequences are used there will be fewer sequences that are similar to one another in the training data which could potentially force the model to generalize better. However, when the training sequences cover a broader range of possible values it can make it easier for the models to actually learn relevant patterns since the model can more easily distinguish between the effects of features. The performance of the neural network underperformed as compared to when the same architecture was trained on the full dataset, potentially indicating difficulties learning on only the 10k sequences in the training data.

8.3 Goal 3: Look into interpretability of select models

8.3.1 What type of sequence features does the logistic regression model emphasize when predicting polyreactivity?

Logistic regression emphasized the effect of AA composition. The logistic regression coefficients (Figure 18B and -D) exhibited larger variation between amino acids and little variation between positions. The coefficients from task “strict polyreactivity” and “non-strict polyreactivity” did not differ in whether they were positive or negative depending on position (Figure 18). We can surmise that the effects of position on residue influence on model prediction is one of differing magnitude. Possibly due to different probability of the residue being involved in binding to the epitope. The emphasis on AA composition is in line with the results showing that the macro f1 score was high when the logistic regression was trained on

AA composition. Our results agree with previous observations that AA composition is predictive of polyreactivity. While no other study has looked directly at predicting polyreactivity based on the total AA composition, high performance has been reported using features based on AA composition (Rabia et al. 2018). It also aligns with previous findings emphasizing the correlation between AA composition or physicochemical properties (tied to AA composition) and polyreactivity (Lecerf et al. 2019; Boughter et al. 2020; Kelly et al. 2018; Birtalan et al. 2008).

The emphasis of the logistic regression on AA composition is not necessarily surprising. The logistic regression has to find coefficient values that predicts the label of all sequences, without taking into account feature interactions. The number of possible 11-mer sequences is very large, and capturing the effects of each position is therefore difficult for a model that assumes the features are independent as this can be due to interaction between positions. And as mentioned AA composition can predict polyreactivity well.

The effects of position could be underestimated due to biases of the simulation. There is a possibility of some bias due to the simulation not taking into account the conformational rigidity in the CDRH3, as it has been proposed that differences between position is due to stabilization of the antibody conformation (Harvey et al. 2022). This can have led to some underestimation of the effect of position on the influence of each residue. The limitations of the data are described further in the limitations section.

8.3.2 Can attribution methods, like integrated gradients, capture sequence features which impact the prediction of the best neural network?

In order to interpret how the sequence features influence the predictions of the neural networks and potentially diagnose problematic behavior, attribution methods such as integrated gradients could be used. To evaluate the potential of the integrated gradients for interpretation of the neural network decision-making, we correlated the mean integrated gradients of 5000 sequences to the coefficients of the logistic regression model trained on the same data.

A limitation with integrated gradients as an attribution method is that it is developed to investigate the feature attributions for each individual input and not at the scale of multiple inputs (Sundararajan, Taly, and Yan 2017). However, such a method has been used to find out globally what the model focuses on (Sarasua, Pölsterl, and Wachinger, n.d.). IGs are often used on images or inputs where each individual feature in the raw data is not necessarily meaningful in and of themselves. For the purposes of this analysis we assume each feature to be the same between inputs. Each position in the input array represents the same AA at the same position between inputs, similar to how the logistic regression views them and we do not care that positional dependencies may alter the meaning of the position. The average of the IGs across multiple sequences could still be informative when we are interested in how the IGs align with previous results.

The method by which integrated gradients are computed fulfills a completeness axiom. That is, the total of the integrated gradients add up to the difference in model output for the baseline and for the actual input (Sundararajan, Taly, and Yan 2017). However, IGs have been known to produce noisy attributions (Kapishnikov et al. 2021) and gradients from the points along the interpolation where the output changes minimally can bias the attributions (Miglani et al. 2020). Therefore, they should not be considered the absolute truth in terms of the overall feature attributions of the model (as would also be the case with feature attributions of individual sequences). The main purpose of this analysis was to get an overview of whether attributions align previous results to see if the integrated gradients can extract reasonable attributions. As the mean of the IGs resembled the logistic regression coefficients, this supported that the IGs were able to extract somewhat reasonable attributions.

The correlation between coefficients and integrated gradients (Figure 20) convey that the mean integrated gradients reflect similar overall attributions as was observed with logistic regression coefficients. The logistic regression model is assigning weights that best model a linear function where each variable/feature can only have one universal weight/coefficient. Therefore, it seems logical that the mean of the integrated gradients sampled from many sequences resembles the coefficients. The gradients do not appear random and reflect features that have been observed through other analysis, both in terms of logistic regression (Figure 18) and observation of class sequences (Figure 5). That the attributions make sense in terms of previously observed results shows that the integrated gradients are detecting relevant features (i.e., that they are working somewhat as intended).

One of the purposes of extracting feature attributions generally is to identify signs that the model is making decisions based on irrelevant features or noise. Both the IGs and logistic regression coefficients displayed attributions based on the AA residue type that are similar and aligns to their overall binding energy with other residues. Thus displaying a level of explainability as it is reasonable that the sequences are more likely to be considered binders for several antigens as the AA composition promotes optimal binding energies. The model appears to have learned relevant features for classification.

8.4 Limitations

The benchmarking of ML models and other analysis was performed on data from the software `Absolut!`. The purpose of the software is to benchmark ML approaches for prediction of different antibody-antigen binding prediction problems. For this purpose `Absolut!` is built to replicate many levels of biological complexity that are present in experimental datasets and are important for evaluating ML approaches. A major limitation is that we cannot directly use train models on `Absolut!` datasets to predict real-world antibodies. Nor can we directly conclude what features are predictive of polyreactivity (Leucine, specific k-mers etc.) in real-world datasets. This thesis informs on the expected types of features that are most predictive of polyreactivity and architectures which capture their relationship with polyreactivity. Since we have only worked on `Absolut!` data so far we cannot be certain as to the extent our conclusions are applicable to experimental data.

Limitations of the software include that it is a simplification of antibody-antigen binding (these limitations are also highlighted in the revised version of the paper which describe `Absolut!` (Robert et al. 2022), of note the author of this thesis is co-author in the revised manuscript):

- In the real world proteins do not fold according to a euclidean grid with only 90 degree angles. And the distances between AAs can vary in real life complexes.
- The framework can only compare CDR sequences/k-mers of fixed length. Thus not all the 11-mers included in this study represent full length CDRH3 sequences.
- The framework allows the antibodies to take any possible binding conformation, and therefore does not take into account the effect of differing flexibility.

- The effects of solvent interactions are not implemented, thus we were not able to account for the effects of solvents on polyreactivity.
- 3D distribution of charge is not implemented.
- `Absolut!` has not implemented features to estimate binding energy to non-peptide antigens. Thus, all the antigens in this study are proteins/peptides made up of AA chains. Polyreactivity is often measured against several types of macromolecules including complex sugars; we cannot be certain that the results would be transferable to non-peptide antigens.

That conclusions from `Absolut!` are generalizable to experimental settings has been validated for certain use cases (Robert et al. 2022). Correspondence between results on `Absolut`! data and experimental data lends credence to the limited levels of complexity that is maintained in the data being sufficient to draw conclusions on the comparative success of ML methods. However, this is the first time that `Absolut!` is used to benchmark ML methods for prediction of cross-reactivity and it has yet to be experimentally validated for this use. Still, if the level of complexity is enough for benchmarking models to predict binding affinity to an antigen it may have the complexity necessary to benchmark methods for cross-reactivity prediction as well.

There are some levels of complexity which may have a larger impact on prediction of polyreactivity in particular. The CDRH3 is flexible and has been shown to take several conformations (Fernández-Quintero et al. 2020), however in some antibodies the CDRH3 may be more flexible than in others. It has been widely suggested that polyreactive antibodies have more flexible ABSs (Dimitrov et al. 2013; Notkins 2004; Guthmiller et al. 2020; Fernández-Quintero et al. 2019). Though interestingly a recent study found that the polyreactive antibodies they studied were unusually rigid (Borowska, Boughter, and Adams 2020). In the `Absolut!` framework each CDRH3 is equally flexible since any possible binding conformation is evaluated and no mechanism has been implemented to account for differential flexibility. Therefore, the data may not capture the full complexity of antibody-antigen polyreactivity due to flexibility. Since all sequences are flexible, we could imagine that they all have a potential to be polyreactive if they only achieve high enough affinity to several antibodies, and that the models would not fully capture the complexity of specific binding. However, we do not know how not including variable flexibility of antibodies influences the prediction performance of the classifiers.

In order to test the effect of variable flexibility the models could be ranked on experimental data accounting for flexibility. Ranking the models on experimental data alone would not tell us directly whether flexibility was an important factor. However, if we had sufficient numbers of polyreactive and specific antibodies with comparable flexibility we could compare prediction on those to prediction of polyreactivity of antibodies with highly flexible CDRH3 in comparison to the specific antibodies. The feasibility of conducting such an analysis would depend on whether it would be possible to find such data and the practicality in attaining data regarding flexibility from such sequences.

The limitation that the sequences are 11-mers of the same size could impact how the performance of the models generalizes to experimental datasets where the sequences are of different lengths. It could give a leg up to methods that cannot deal with input of unequal sizes. It may be desirable then to align 11-mers back to their original CDRH3 sequence and develop a method for evaluating the binding energy of the full-length CDRH3 sequence. However, that would possibly be challenging due to not being able to account for structural constraints.

Since conclusions from `Absolut!` data regarding the ranking of models that predict specific affinity have been shown to be generalizable to experimental data, we could imagine that such would be the case also for non-specific polyreactive binding despite the simplifications. Since we may expect some of the same complexity of specific binding to translate to non-specific binding.

8.5 Outlook and Future Perspectives

Our results represent early findings, benchmarking approaches to guide further development of ML models for prediction of antibody polyreactivity. At the beginning of this project several features of polyreactive antibodies had been elucidated, however few studies had looked into directly predicting polyreactivity *in silico* beyond using net charge of the CDRs or hydrophobicity as predictors. With one relatively recent study being the only study to our knowledge comprehensively investigating predicting polyreactivity using ML approaches (Boughter et al. 2020). During this project we have benchmarked several ML models for predicting polyreactivity based on sequence information where polyreactivity was estimated against 142 antigens, four different ways using data from `Absolut!`. We showed that in all cases polyreactivity was predictable and provided a ranking of select models. We have investigated predictive power based on k-mer length using basic CNN models, which indicated that short patterns can be predictive of polyreactivity. Additionally, we validated that selected models did not rely on sequence similarity (LD) for prediction and looked into interpretability of said models.

Still there are questions remaining. We have not yet benchmarked methods according to how well they predict polyreactivity defined through binding on a separate list of antigens. It has been noted that the observed polyreactivity of an antibody can differ based on the assay used (Lecerf et al. 2019), and shown that estimations of polyreactivity do not necessarily correlate based on the antigens polyreactivity is tested against (Harvey et al. 2022). And as the binding of the antibody to the antigen is dependent on the antigen as well as the antibody it would be of interest to know how well the models can predict polyreactivity against different antigens. For this purpose `Absolut!` data can be used. Polyreactivity can be estimated based on different antigens for the antibody sequences in the training and test data.

The nature of potential feature interactions and the makeup of potential motifs predictive of polyreactivity were not further investigated in this work. Investigation into how the neural networks make predictions as well as possible short motifs could help further elucidate what levels of feature complexity are predictive of polyreactivity. As the integrated gradients were able to identify relevant attributions we could potentially use these for further investigation.

We would need some sort of experimental validation to evaluate the generalization of our observations. Ideally, the experimental validation should be done on a dataset which closely mirrors the limitation of `Absolut`! data. Moreover, our study only includes proteins. Therefore, the data from protein-arrays, PSR and potentially ELISA would be beneficial, where several protein antigens are used to evaluate polyreactivity. Our dataset includes only naive antibodies to avoid bias due to different levels of affinity maturation. So further validation on our findings would be more relevant on naive antibody sequences.

The traditional *in-vitro* polyreactivity datasets are relatively small, which also limits the validation of our findings. As we observed, on small datasets the logistic regression outperformed the neural network. However, with recent advancements in assay development (Harvey et al. 2022) it may become feasible to conduct more comprehensive evaluations.

9 Methods

9.1 In Silico Polyreactivity Dataset

We use an *in silico* dataset, generated using the simulation software `Absolut!` that simulated the binding between 7.3 million CDR-H3 sequences and 142 nonredundant antigens (Robert et al. 2022). The antigens were selected from the AbDb database (Ferdous and Martin 2018), and originated from 41 different species and relevant for 45 different disease classifications/ adverse health effects (example cancer, allergy, infection etc.). The antibody sequences used were taken from murine BL6 FACS sorted naive B cells (Greiff et al. 2017).

`Absolut!` simulates the binding of peptides of 11 AAs (AAs) to coarse-grained antigen structures described in a 3D lattice where one lattice position can only contain one AA and covalently bound AAs are neighbors in the lattice. The 3D lattice representations of the selected antigens were generated from their PDB structure as described in (Robert et al. 2022). The root-mean-square distance (RMSD) is minimized to find an antigen lattice representation that best reflects its original PDB structure.

The energy of the CDR-antigen complex is calculated through interaction potentials between interacting partners in the complex. Residues adjacent to one another which are not covalently linked are considered interaction partners. The affinity between interaction partners are determined by experimentally inferred interaction potentials (Miyazawa and Jernigan 1996). Both the binding energy between antibody sequence and antigen as well as self-folding energy is taken into account. The self-folding energy refers to the sum of interaction potentials between interacting partners in the same protein. The binding energy is measured as the sum of interaction potentials between partners in the antigen and antibody sequence. The energy of the complex does not equate to the affinity as the affinity depends on information from the rest of the antibody which is not available, however it does reflect a tendency for the CDRH3 to bind to the antigen.

For each antibody CDRH-3 sequence, every possible contiguous 11-mer was assessed for binding to each of the 142 antigen lattice structures, returning a binding energy and the best antibody-antigen structure. For each antigen, the CDRH3 sequences with the lowest 1% binding energy were used to set a threshold, all 11-mers whose binding energy was estimated as lower than the threshold were classified as binders (further information can be found in the original paper (Robert et al. 2022)). The information regarding the binders were stored in a table containing binary information of whether the sequence was bound to each antigen (Figure 22). Only sequences that were considered a binder for at least one antigen were kept.

	CDR3.seq	n.bind	antigen
	<chr>	<dbl>	<chr>
1	CARVGWLLGYW	5	00000000000000000001001000000000000000000000000000000000000000000~
2	LPYYYDYGYYF	1	00000000000010000000000000000000000000000000000000000000000000000~
3	ARARLLPFDYW	2	00000000000000000000000000000000000000000000000000000000000000001~
4	CARIARLFYAM	3	00000000000000000000001000100000000000000000000000000000000000000~
5	AIFYGNVSWFA	1	00000000000000000000000000000000000000000000000000000000000000000~
6	KDLLWLRRGYF	1	00000000000000000001000000000000000000000000000000000000000000000~

Figure 22**: Description of the polyreactivity dataset** with the eleven amino acids long substrings of the CDR-H3 sequences that were lower than the 1% energy threshold which are defined as binders. It also contains a column which conveniently shows how many antigens are bound among the 142. The third column contains a binary number for each antigen, describing the target antigens. Only sequences with at least one target antigen are kept. Only 6 sequences are shown.

Of the 142 antigens, most are from different proteins with a couple of exceptions. There are 2 versions of Interleukin 2, one from *Homo sapiens* and the other *Mus musculus*. There are also 2 separate entries for Interleukin 13. Comparison of the sequences considered binders for both Il-2s showed ~10% overlap and between the Il-13s there was only a 2% overlap (Figure 23). This was not particularly high (even on the lower end) in terms of the binder overlap between the included antigens. Thus, we considered them as separate antigens.

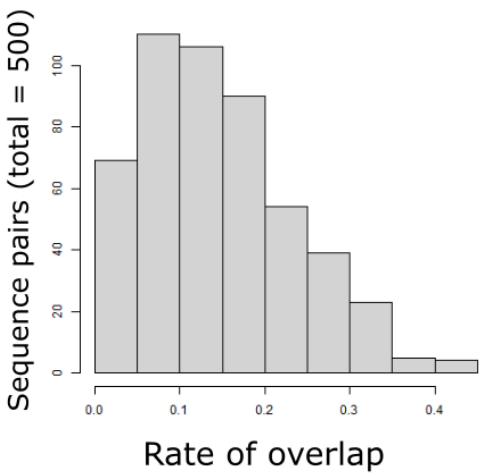


Figure 23: **Rate of overlap in the antibodies bound by 500 randomly selected pairs on antigens.**
The rate of overlap was calculated by taking the number of antibodies considered a binder to the first antigen that was also considered a binder to the second antigen and dividing that by the number of antibodies total considered a binder for each adding the results together (for each antigen in the pair) and dividing by 2.

9.2 Formulation of the Problem and Data Processing

We aim to formalize the problem of polyreactivity prediction as a classification problem where each sequence is annotated with a class describing how many antigens it binds. To determine strategy for dividing classes the distribution of the number of antibodies that were considered a binder to each number of antigens was inspected. This revealed a smooth distribution where 445.500 (~43%) antibodies recognizing only one antigen, while 146.997 (~14%) recognized 2, 79.573 (~8%) recognized 3 antigens, and the remaining 357.234 (~35%) recognized 4 or more antigens. Since the distribution was smooth the polyreactivity classes are defined based on the amount of antibodies that would be observed in a class rather than dividing classes based on observed gaps in the distribution (If we for example imagined the distribution as two peaks with few intermediate sequences).

In order to detect potential outliers (binding to distinctly large numbers of antigens) that were not observed in the analysis of the distribution via histogram, a combined box and scatter plot was made to observe outliers directly.

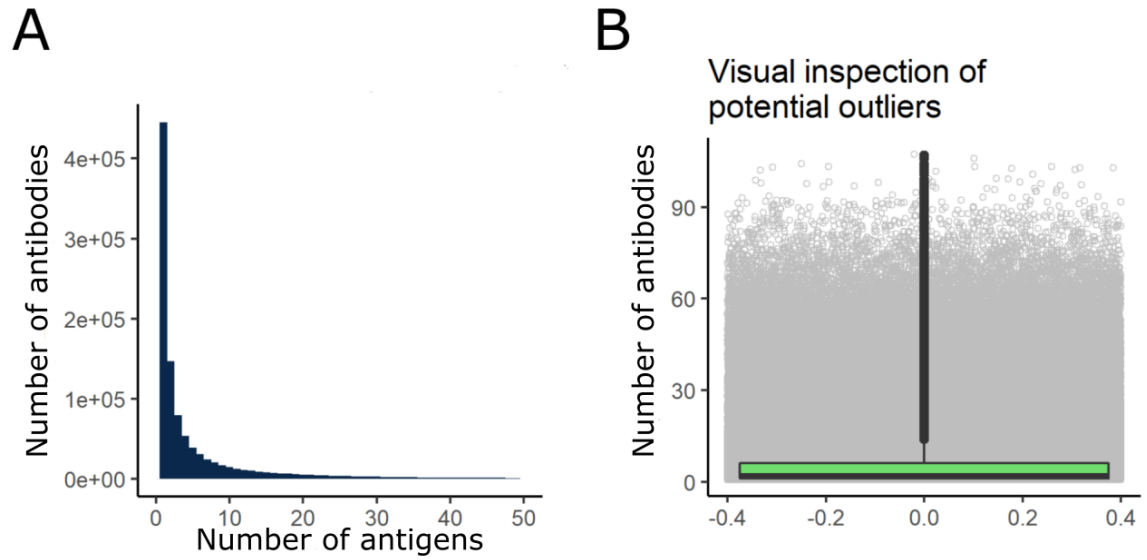


Figure 24**: The number of antigen each antibody is estimated to bind follows a continuous distribution.** A: Represents the number of CDR3 sequences (vertical-axis) which binds to a given number of antigens (horizontal-axis) as a histogram. B: Box-plot showing the medium, quartiles and suggested outlier candidates. The ticker black line towards the end of the plot is made up of consecutive dots. The data is based on the training data.

We did not identify obvious outliers in Figure 24B. Considering all the data points created an almost continuous line, and the visualization of individual data points in gray did not reveal points markedly separated from the others, no sequences were considered outliers as they were not markedly different from any other antibody in the data.

9.3 Test Exclusion

The data was separated into "training + validation" data (80%) and "test" data (20%) in a stratified manner that ensures there is roughly the same proportion of CDR3s binding a given number of antigens in both the test and training datasets. The sequences were separated into "test" before classes were assigned. The test dataset is excluded from all further analyses and never provided to the ML until single final testing. To ensure smooth stratification (and to avoid single sample "classes"), the sequences binding to 51 or more antigens were grouped together as the "51+" class. The train and test data was then stored separately.

9.4 List of Tasks, what is Input and Output

The sequences were grouped into classes based on the number of bound antigens to fit multiple ML formalisms (Table 2). For task "strict polyreactivity", the “specific” class was defined strictly as those that are only bound to one antigen, a fraction which makes up about 40% of the data. For tasks “non-strict polyreactivity" and "coarse multiclass" the polyreactive sequences were separated from the specific/oligoreactive sequences by the 3rd quartile (about 6). Separating the classes by the 3rd quartile was done so that there would be less data loss and because it provided a relative measure of what it would mean to bind “many" antigens. The 5 classes of "finer grained multiclass" were defined by first assigning those sequences which bound 1 antigen to class 1, since they could not be divided up, and then dividing the remaining sequences about equally (quartiles).

Table 2: **List of ML tasks and the borders defining what class the CDRH3 sequences are assigned to based on the number of antigens they bind in the dataset.**

Task	Class 1	Class 2	Class 3	Class 4	Class 5
"strict polyreactivity"	1	>1			
"non-strict polyreactivity"	1-6	>= 7			
"coarse multiclass"	1	2-5	>=6		
"finer grained multiclass"	1	2	3-5	5-11	>11

9.5 Balancing

The data was balanced using undersampling (Table 3) with a random number of sequences from the majority class (or the classes which were not the minority). Undersampling was deemed appropriate since the smallest class was always containing large amounts(>100k) of sequences for training models.

Table 3: **The size of the balanced dataset and number of samples in the unbalanced classes for all tasks**

task	Size of balanced data	class 1	class 2	class 3	class 4	class 5
"strict polyreactivity"	891.000	445.500	583.804			
"non-strict polyreactivity"	468.516	795.046	234.258			
"coarse multiclass"	794.607	445.500	318.935	264.869		
"finer grained multiclass"	595.855	445.500	146.997	171.938	119.171	145.698

9.5.1 Sequence Encoding

ML approaches were tested on both the AA composition and one-hot encoding. These are features taken directly from the sequence requiring minimal amounts of engineering. AA composition has been implicated in polyreactivity and one-hot encoding is a popular method of encoding biological sequence information (Trabelsi, Chaabane, and Ben-Hur 2019; Jing et al. 2020). AA composition also serves as a control to ascertain whether positional information improves model performance. For AA composition, each of the 20 common AA side chains were represented as one feature where the rows reported how many times said AA occurred in the CDR3. For one-hot encoding the CDR3 sequences were first divided into single AAs where each position is one feature in R and then exported to a csv file. The features were then converted to one hot encoding in python using pandas `get_dummies()` ("Pandas.get_dummies — Pandas 1.4.3 Documentation" n.d.). For each position the first column was removed to remove data redundancy, since the sum of the values along the 20 AAs per position is always 1, and therefore prevents giving the model the same data twice.

	AA_1_C	AA_1_D	AA_1_E	AA_1_F	...	AA_11_T	AA_11_V	AA_11_W	AA_11_Y
0	0	0	0	1	...	0	0	1	0
1	0	0	0	0	...	0	0	1	0
2	1	0	0	0	...	0	0	0	1
3	0	0	0	0	...	0	0	0	0
4	0	0	1	0	...	0	0	1	0
...	...	...	...	...	...	...	...	...	...
468511	0	0	0	0	...	0	0	0	1
468512	1	0	0	0	...	0	0	0	1
468513	0	0	0	0	...	0	0	0	0
468514	0	0	0	0	...	0	0	1	0
468515	1	0	0	0	...	0	0	0	1

Figure 25: **Example of one-hot encoding of sequence data.** Displayed is an output showing one-hot encoded data where the first 4 amino acids at the first position and last 4 at the 11th position can be seen. Where the amino acid is present at that position the data frame contains a 1 and where not 0.

9.5.2 K-fold validation

The models were evaluated using stratified 10-fold cross-validation. As a control the models were also trained and tested on a dataset with shuffled labels to inspect that the accuracy was the same as random guessing (basal accuracy to be expected at random).

9.5.3 Performance metrics

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Macro Precision} = \frac{\sum_{i=1}^{k\,classes} Precision_i}{k\,classes}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{Macro Recall} = \frac{\sum_{i=1}^{k\,classes} Recall_i}{k\,classes}$$

$$\text{Macro F1-score} = 2\frac{MacroPrecision \times \text{Macro Recall}}{\text{Macro Precision} + \text{Macro Recall}}$$

$$\text{Accuracy} = \frac{\text{True Predictions}}{TruePredictions + FalsePredictions}$$

9.5.4 Leucine

A logistic regression model (sklearn LogisticRegression) was fitted to the data and evaluated using 10-fold cross-validation. The hyperparameters were not tuned. As logistic is typically less sensitive to hyperparameters, and the purpose of the experiment was to see whether leucine was a good predictor in itself and not directly focusing on optimizing the model, hyperparameter optimization was not deemed necessary.

9.5.5 Architectures tested with AA Composition and One-Hot Sequences

Decision Tree (`scikit-learn`)

Logistic Regression (`scikit-learn`)

Random Forest (`scikit-learn`)

Feed Forward Neural Network (`Tensorflow`)

Convolutional Neural Network (`Tensorflow`)

9.6 Hyperparameter Strategy

In order to find decent hyperparameter values to optimize the models a selection of hyperparameters which we believed could potentially affect the model performance was chosen (Table 4). Then for each hyperparameter the effect of a broad range of values on macro f1 (both train and validation), recall and precision (validation) was inspected. Based on this inspection a second range of values were selected to test for the effect of different combinations of parameter values using grid-search. The best performing hyperparameters (maximum macro f1 on validation data across 5 folds) were thus used to test the performance of the model.

When validating/testing all shallow models and for tuning logistic regression the data was scaled using `StandardScaler`($Z = \frac{x - \overline{x}}{\sigma}$), however when tuning decision trees and random forests it was not used as these architectures do not require scaled data. It should not impact the performance.

9.6.1 Hyperparameter Boundaries

Decision tree: max_features, max_depth and min_impurity_decrease

Random forest: max_features, n_estimators

Logistic Regression: C, tol

Table 4: **Range of hyperparameters tested for each task for shallow models, depending on the architecture.**

model	encoding	parameter grid
decision tree	amino acid composition	[{'max_depth': [10, 13, 15, 17, 20, 25], 'min_impurity_decrease': [0., 0.0001, 0.001], 'max_features': [5, 10, 15, 20]}]
decision tree	one-hot	[{'max_depth': [25, 30, 33, 35, 37, 40, 45, 50], 'min_impurity_decrease': [0., 0.0001, 0.001], 'max_features': [100, 125, 150, 175, 209]}]
logistic regression	both	[{'logisticregression__C': [0.25, 0.5, 0.75], 'logisticregression__tol': [1e-5, 1e-3, 1e-2]}]
random forest	amino acid composition	[{'max_features': [5, 10, 12, 16, 20], 'n_estimators': [10, 25, 50, 75, 100, 150, 300]}]
random forest	one-hot	[{'max_features': [20, 30, 50, 100, 209], 'n_estimators': [30, 50, 75, 100, 150, 300]}]

In order to tune hyperparameters for the feed forward - and convolutional neural networks, a selection of parameters were chosen to test against arrays of different values. Each parameter was tested separately to other parameters according to the values in Table 5.

Table 5: **Parameters tested individually for both neural networks.** Batch denominator is the number of batches the data is cut into. Layers denote the number of hidden layers in the network.

Model	Parameter	Values
Both	Batch Denominator	[5, 10, 25, 50, 75, 100, 200, 500]
Both	Learning Rate	[0.001, 0.002, 0.01, 0.02, 0.1, 0.2]
FNN	Layers	[1, 2]
FNN	Neurons	[5, 10, 25, 50, 100, 150, 200]
CNN	Filters	[1, 10, 50, 100, 200, 400]
CNN	Kernel Size	[1, 2, 3, 4, 5, 6, 7, 8, 9]
CNN	Max Pooling	[1, 2, 3, 4]

Afterwards, a grid search was performed on the architecture design parameters (Table 2 and Table 3), using the best hyperparameters for learning rate and batch denominator. These values were chosen based on inspecting the results of tuning each parameter separately. Certain parameters were deemed to be more important. The metric emphasized was macro f1, and the best selected parameters are shown in Table 8.

Table 6: **The parameter values that were tested using grid-search for the feed-forward neural net**.

Parameter	Tested Values
n Neurons	[1, 5, 10, 20, 30, 50, 75, 100, 125, 150, 175]
Layers	[1, 2]

Table 7: **The parameter values that were tested using grid-search for the convolutional neural net.**

Parameter	Tested Values
Filters	[1, 2, 3, 4, 5, 6, 7, 8]
Kernel Size	[1, 5, 10, 20, 30, 50, 75, 100, 150]
Max Pool	[1, 2, 'kernel size max']

$$a = \text{sequence length}$$
$$b = \text{kernel size}$$
$$\text{'kernel size max}' = a - b + 1$$

To perform one max pooling operation across the entire output of the convolutional layer the max value of the max pool parameter had to be determined ("kernel size max"). The maximum possible value for max pool parameter based on the kernel size was one more than the length of the sequences minus the kernel size. Example: With kernel size 8 and stride of 1 the output of the convolution for that filter is of size 4 (11-8 +1).

9.6.2 Hyperparameters Shallow Learning

Table 6: **Best hyperparameter values for shallow models trained on amino acid composition.**

model	task	parameter 1		parameter 2		parameter 3	
decision tree	"strict polyreactivity"	max_depth	15	max_features	20	min impurity decrease	0
logistic regression	"strict polyreactivity"	C	0.25	tol	1e-05	none	
random forest	"strict polyreactivity"	n estimators	300	max_features	12	none	
decision tree	"non-strict polyreactivity"	max_depth	15	max_features	20	min impurity decrease	0
logistic regression	"non-strict polyreactivity"	C	0.25	tol	1e-05	none	
random forest	"non-strict polyreactivity"	n estimators	150	max_features	16	none	
decision tree	"coarse multiclass"	max_depth	17	max_features	20	min impurity decrease	0
logistic regression	"coarse multiclass"	C	0.5	tol	1e-05	none	
random forest	"coarse multiclass"	n estimators	300	max_features	16	none	
decision tree	"finer grained multiclass"	max_depth	17	max_features	20	min impurity decrease	0
logistic regression	"finer grained multiclass"	C	0.75	tol	1e-05	none	
random forest	"finer grained multiclass"	n estimators	300	max_features	16	none	

Table 7: **Best hyperparameter values for shallow models trained on one-hot encoded sequences**.

model	task	parameter 1		parameter 2		parameter 3	
decision tree	"strict polyreactivity"	max_depth		max_features		min impurity decrease	0
logistic regression	"strict polyreactivity"	C	0.5	tol	0.01	none	
random forest	"strict polyreactivity"	n estimators	300	max_features	100	none	
decision tree	"non-strict polyreactivity"	max_depth	35	max_features	209	min impurity decrease	0
logistic regression	"non-strict polyreactivity"	C	0.25	tol	1e-05	none	
random forest	"non-strict polyreactivity"	n estimators	300	max_features	100	none	
decision tree	"coarse multiclass"	max_depth	40	max_features	209	min impurity decrease	0
logistic regression	"coarse multiclass"	C	0.5	tol	1e-05	none	

random forest	"coarse multiclass"	n estimators	300	max_features	100	none	
decision tree	"finer grained multiclass"	max_depth	40	max_features	209	min impurity decrease	0
logistic regression	"finer grained multiclass"	C	0.75	tol	1e-05	none	
random forest	"finer grained multiclass"	n estimators	300	max_features	100	none	

9.6.3 Neural Networks

A

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 209)               0
_________________________________________________________________
dense (Dense)                (None, 100)               21000
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 21,101
Trainable params: 21,101
Non-trainable params: 0
```

B

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 4, 100)            15300
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 2, 100)            0
_________________________________________________________________
flatten (Flatten)            (None, 200)               0
_________________________________________________________________
dense (Dense)                (None, 1)                 201
=================================================================
Total params: 15,501
Trainable params: 15,501
Non-trainable params: 0
```

Figure 26**: Architecture of the best performingFNN (a) and CNN (b).** (a) FNN dimensions: One-hot encoded 11-mers (220 dimensions) are preprocessed for correlated (dummy) features by removing the first amino acid of each position, that would be 1 - sum (other amino acids at this position), therefore 209 dimensions. The best architecture contained one hidden layer of 100 neurons (ReLu activation). (b) CNN dimensions: 1D convolutional neural network with one convolutional layer of 100 filters that span over 8 positions. The best pooling was achieved by taking the maximum of every block of two positions. The activation function used in the final layer of each model was Sigmoid.

In order to complement the previously tested shallow architectures, we consider a feed-forward neural network architecture (FNN) with one or more layers, and a convolutional network (CNN). For the CNN we selected a basic architecture used on 2D input and altered it to become a 1D CNN (Zhou 2020) which was fitted to our data. The models were created and trained using `Tensorflow`. To train the networks the Adam optimizer was used, and the loss function was binary cross-entropy. The model with the best hyperparameters was then evaluated using 10-fold cross validation. The models were additionally validated on data with shuffled labels as a control in the same manner as the other models.

9.6.4 Hyperparameters Neural Network

Table 8: **Best parameter values for both FNN and CNN**

Model	Parameter 1	Value	Parameter 2	Value	Parameter 3	Value
FNN	n Neurons	100	Layers	1	None	
CNN	Filters	100	Kernel Size	8	Max Pooling	2

9.7 Version of Libraries Used

```
R version = 4.0.4
Python = 3.7.6 (private computer) and 3.7.4 (immunohub)
R:
ggplot2 3.3.5
tidyverse 1.3.1
Python:
Tensorflow 2.6.0
scikit-learn 0.24.2
numpy 01.20.3
matplotlib 3.4.2
scipy 1.7.1
```

9.8 Calculation of Levenshtein Distances

Levenshtein distance was chosen as the metric to represent distance between the sequences. Levenshtein distance is the minimum number of alterations that separate two character strings. In terms of protein sequences the Levenshtein distance is the minimum number of substitutions, deletions and/or insertions needed to transform one sequence into another.

To calculate the Levenshtein distances between sequences the `distance()` attribute from the package `python-Levenshtein` was used and the results stored in a numpy matrix as integer values. Since we used sequences of the same length (11 AAs), normalization of LD based on sequence length was not necessary.

The algorithm to calculate LD can be expressed as such:

$$\mathrm{LD(a,b)} = \begin{cases} \text{length a}, & \text{if length b} = 0, \\ \text{length b}, & \text{if length a} = 0, \\ \mathrm{LD(rest\ of\ a,\ rest\ of\ b\ )}, & \text{if a[0]} = \text{b[0]} \\ 1 + \min \begin{cases} \mathrm{LD(rest\ of\ a\ ,b)} \\ \mathrm{LD(a,\ rest\ of\ b)}, \\ \mathrm{LD(rest\ of\ a,\ rest\ of\ b)} \end{cases} & \text{otherwise} \end{cases}$$

(Contributors to Wikimedia projects 2003)

Tokens a and b represent two strings (or substrings) being compared. The statement above specifies how the LD is calculated for each position of the strings. If the length of either string is zero, then the LD is equal to the length of the other string. Since it is unlikely that one wants to calculate the LD between a string and nothing, this step is relevant to later positions when the algorithm has run through all positions of one string. If the first position of both strings contain the same character, then nothing is added to the LD and the LD is calculated for the rest of the string positions. Else, 1 is added to the LD and the algorithm is repeated on either the remainder of string a vs string b, the remainder of string b vs string a or the remainder of both. The first two options correspond to insertions/deletions and the last correspond to a substitution.

9.9 Investigating Distributions, UMAP and Clustering

A sample of 10K sequences was randomly selected, as a basis, to generate dissimilar training/validation datasets by clustering (the size of the dataset was reduced to improve time efficiency). 5K of the selected sequences were from each class as defined by task "non-strict polyreactivity" (specific/oligoreactive vs polyreactive).

To investigate how often the sequences have any given minimum distance to another sequence, a histogram showing this distribution was created (Figure 12). To get the distribution of minimal distance the values from the matrix diagonal (0) were removed. The matrix was turned into a long form dataframe, which was then grouped by each individual sequence. The smallest values in each group were selected to be used for display in a histogram.

9.9.1 UMAP

In order to visualize how the sequences relate to each other in terms of Levenshtein distance, the UMAP dimensionality reduction algorithm was used (for results see Figure 2). To find decent values for the parameters `n_neighbors` and `min_dist` *("Basic UMAP Parameters — Umap 0.5 Documentation" n.d.)* a range of possible values were chosen for each to evaluate in combination using a nested loop. The results were then saved as figures and an optimal combination selected by visual inspection. Parameters were selected based on the ability to separate clusters.

Table 9**: Range of values and the parameters that were tested for optimization of umap dimensionality reduction.**

Parameter	Value Range
min_dist	[0.1, 0.3, 0.5, 0.7, 0.9]
n_neighbors	[5, 7, 10, 30, 50, 100, 200, 500, 1000, 5000]

The distance matrix was given to UMAP and it was specified that the distances were precomputed:

```
lv_umap = umap.UMAP(metric='precomputed', min_dist=0.1, n_neighbors=100)
coordinates = lv_umap.fit_transform(mat_levenshtein)
```

9.9.2 Hierarchical Clustering and Fcluster

Previously described methods for reducing similarity between train and test data has involved separation of clusters (Petti and Eddy 2022). We first attempted to cluster the sequences by LD using hierarchical clustering. Average linkage was able to find some clusters in the data, however they were of very uneven size. Attempts to use the `fcluster` attribute from the `scipy` library to get cluster labels resulted in only a few larger clusters and several small clusters (Figure 13B). For the threshold parameter t, we set the value to 9 ("Scipy.cluster.hierarchy.fcluster — SciPy v1.8.1 Manual" n.d.). The criteria used was "distance", for this criteria the t parameter determines the maximum cophenetic distance between any data points in a cluster. Several values of t were tested. 9 was chosen as 8 led to large numbers of small clusters and 10 combined all samples into one large cluster.

9.9.3 DBSCAN

DBSCAN was used to cluster the result of UMAP in order to get cluster labels for subsequent analysis of model generalization. Parameter values for DBSCAN ("sklearn.cluster.DBSCAN" n.d.) were selected in a similar fashion to UMAP. The criterias that were prioritized was that there should be enough clusters to separate them in a balanced manner and that the LD was not too large between training and validation, but not too many as to generate a lot of overlapping clusters.

Table 10: **The range of values of and the parameters that were tested for optimization of dbscan clustering.** Lists containing the values compared for hyperparameters epsilon and minimum samples.

Parameter	Value Range
eps	[0.25, 0.5, 0.75, 1]
min_samples	[5, 10, 20, 24, 25, 27, 30, 35, 40, 45, 50, 55, 75, 100, 150, 500]

9.9.4 Silhouette Score as to Quantify Clustering Efficiency

The quality of the clusters that were identified using UMAP and DBSCAN were evaluated using Silhouette score.

$$\mathrm{a} = \mathrm{mean}(\text{intra-cluster distance})$$
$$\mathrm{b} = \mathrm{mean}(\text{distance from each point in a to the points of nearest cluster})$$

$$\text{Silhouette coefficient}(\mathrm{x}) = \frac{a - b}{max(a, b)}$$

("Sklearn.metrics.silhouette_score" n.d.)

The silhouette score is the mean silhouette coefficient for all clustered samples. The silhouette score for the clustering is the average score across all data points, where each individual score is based on the average of the distances to all other points in the cluster compared to the distances to points in the nearest cluster.

9.10 Estimating Macro f1 when the model is trained and tested on different Clusters defined by Similarity

UMAP and DBSCAN were used to get cluster labels. Then the sequences that were labeled by DBSCAN as noise were removed. The same 10k sequences that were used for UMAP and DBSCAN were used for this analysis. Assignment of each cluster to the training or validation data was decided at random with 6 being assigned to train and the rest (n=4) being assigned to validation. That there would not be too few training or validation sequences were prioritized.

As a control, sequences of the same 10k subset were randomly split into training and validation data with the same number of sequences in each as with the clustered split. This was done as the number of sequences in each clustered split varies and for fair comparison it was desirable to test a random split with the same number of sequences. This process was repeated 100 times.

9.11 Estimating Macro f1 depending on Minimum Distance to Training Data

To get the macro f1 of the model on test sequences with different minimum distances (LD) to any training sequence, the minimum distance was used to split the validation dataset into subsets. These are defined by having the same minimum distance to any given training sequence.

Only subsets with minimum distance from 1 to 6 were generated as there were not enough sequences with higher minimum distance to get a good estimation of accuracy. The simulation was repeated 10 times. Each round 50k sequences were used where 10k was used for training and the rest would be split into the 6 validation subsets. The size of the validation data had to be sufficiently large in order to get a decent subset size (about 100 sequences or more). However, with increased training size the probability of there being closely related sequences increases. Therefore the size of the training dataset was not increased above 10k.

The macro f1 score was calculated for each subset and stored in a dataframe along with the number of sequences in the subset. The model used to predict the classes was logistic regression and the classification scheme used was that of task "non-strict polyreactivity" (specific/oligoreactive vs polyreactive). In order to see whether the FNN score was based on similarity the quantification analysis performed on logistic regression was repeated on the FNN.

9.11.1 Positive control for data leakage

A positive control for the data leakage quantification was made so it would be possible to compare model performance to an instance where classification was based on similarity. For each sample in the validation data the minimum LD to training sequences was calculated and the most similar sequences identified. A heuristic based guesser would take the class of the similar sequences and make a prediction using the average class. The class would be positive (polyreactive) if the mean was above 0.5 and negative if lower. If the mean was 0.5 then class was randomly assigned.

Table 11: **Mean and standard deviation in sample size for minimum distance subsets.** Results found in Figure 17. Numbers reflect the mean and standard deviation (SD) of the size of the subsets used to evaluate macro f1 based on minimum Levenshtein distance to training sequences.

Minimum Levenshtein Distance	Sample Size Mean	Sample Size SD
1	2209.3	31.3
2	114464.7	105.8
3	16102.6	81.2
4	5918.2	54.7
5	1166.3	26.6
6	133	9.4

9.12 Integrated Gradients

In order to examine the contributions of each feature to the model predictions we calculated integrated gradients (Sundararajan, Taly, and Yan 2017). They were calculated using interpolation between one real case input data and a baseline tensor containing only zeros. A zero baseline was taken as an example of a negative input instance. We used 100 interpolation steps, same as has been used to cluster binder vs non-binding 11-mer sequences (Robert et al. 2022). The interpolation between the input and a baseline is crucial for the sensitivity of the method, as it searches the values in input space between negative (for the baseline) and positive (for the output), where the output increases (has not become saturated).

The gradients for all interpolation steps were summed up for each feature (positions in the tensors) for the same input and divided by the number of interpolation steps. The yield of this process is the mean of the local gradient for each feature. To get an idea of the average effect of each feature the integrated gradients were summed up over a collection of 5000 polyreactive sequences before being divided on how often the feature was observed. The results are shown in Figure 20.

The choice of an all zero baseline was based on easy interpretability and making sure that the prediction for the baseline would be negative (negative class) and close to zero. Per recommendation (Sundararajan, Taly, and Yan 2017) we checked the baseline prediction, the prediction for the baseline values was 7.656236e-34. This works well for the positive class (though not necessarily so well for the negative class).

9.13 Coefficients

In order to gain insight into how logistic regression predicted polyreactivity we trained the logistic regression model on both data containing AA composition and one-hot encoding for datatask "strict polyreactivity" and "non-strict polyreactivity" and then extracted the coefficients after training (results displayed in Figure 18). 80/20 train-validation split was used to verify that the performance was similar to what has been seen previously and that the performance was similar when the features were not scaled.

9.14 Balancing End-Motifs

In order to inspect whether the end motifs were biasing ranking, the sequences were balanced based on both end motif (CAR and DYW/DVW) and class. Balancing was performed so that each version of the beginning (CAR, AR, R) and stop end motifs (D, DV, DY, DYW) occured with the same frequency within each class as defined by datatask "non-strict polyreactivity". The ends of the CDRH3 sequence were thus not removed and present in the data but since these motifs were equally dispersed between classes it would not be easy for the models to make classifications based on these. The sequences were divided into subsets which consisted of sequences that contained both ends of the CDRH3 (complete sequences), those that contained either end and 11-mers from the middle of the CDRH3 which contained none of the motifs. These subsets were combined.

9.15 Preprocessing and Model evaluation on Test-Data

Test data for each task was treated similar to the training data in terms of preprocessing features and classification. The test-data was not balanced. Models were trained using the entire balanced training data for each task. The test was performed once for each model investigated using kfold validation (task, architecture, encoding), except the model using only Leucine as indicator.

9.16 Hypothesis testing

In order to test whether observed differences in performance between two classifiers were statistically significant or due to chance, McNemar's test was used. The McNemar's test is used to compare paired nominal data. It has been shown to be suitable for comparing classifiers per low type 1 error rate (Dietterich 1998), and was recommended when the models were only measured on one test set. It directly tests whether two trained classifiers are different, however it was found to work well for comparing classifiers regardless of training with some constraints (Dietterich 1998). The limitations of the test when drawing conclusions about whether two models (untrained) are significantly different include that it requires the assumption that variation in classifier performance due to sampling and the randomness of the learning algorithm is low. As testing was performed once we do not know how much the performances would vary on the test-data. However, the standard deviations of the macro f1 produced during cross-validation were low (Supplementary Table5).

9.16.1 Execution

The test was carried out on each pair of classifiers for each task, classifiers for different tasks were not compared as the variation in accuracy between tasks was large. The Bonferroni method was used to correct for multiple testing (Jafari and Ansari-Pour 2019). Adjusted p-values were calculated using the number of tests for said task multiplied by the p-value. Adjusted p-values are p-values adjusted for multiple testing by multiplication by the number of tests performed.

9.17 Hardware used

Computations were run on an internal e-infrastructure “Immunohub”, which is funded by the University of Oslo (UiO) and operated by the hosting lab (GreiffLab: https://greifflab.org/) and SandveLab (https://sandvelab.org/) together with the University Center for Information Technology, University of Oslo, IT-Department (USIT). As well as a private Windows computer by Asus with Intel(R)Core(TM) i7-10510U CPU.

10 References

Aguilera, I., J. Melero, A. Nuñez-Roldan, and B. Sanchez. 2001. "Molecular Structure of Eight Human Autoreactive Monoclonal Antibodies." *Immunology* 102 (3): 273–80.

Ahmad, Muhammad Aurangzeb, Carly Eckert, and Ankur Teredesai. 2018. "Interpretable Machine Learning in Healthcare." *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. https://doi.org/10.1145/3233547.3233667.

Akbar, Rahmad, Habib Bashour, Puneet Rawat, Philippe A. Robert, Eva Smorodina, Tudor-Stefan Cotet, Karine Flem-Karlsen, et al. 2022. "Progress and Challenges for the Machine Learning-Based Design of Fit-for-Purpose Monoclonal Antibodies." *mAbs* 14 (1): 2008790.

Akbar, Rahmad, Philippe A. Robert, Milena Pavlović, Jeliazko R. Jeliazkov, Igor Snapkov, Andrei Slabodkin, Cédric R. Weber, et al. 2021. "A Compact Vocabulary of Paratope-Epitope Interactions Enables Predictability of Antibody-Antigen Binding." *Cell Reports* 34 (11): 108856.

Alipanahi, Babak, Andrew Delong, Matthew T. Weirauch, and Brendan J. Frey. 2015. "Predicting the Sequence Specificities of DNA- and RNA-Binding Proteins by Deep Learning." *Nature Biotechnology* 33 (8): 831–38.

Bailly, Marc, Carl Mieczkowski, Veronica Juan, Essam Metwally, Daniela Tomazela, Jeanne Baker, Makiko Uchida, et al. 2020. "Predicting Antibody Developability Profiles Through Early Stage Discovery Screening." *mAbs* 12 (1): 1743053.

"Basic UMAP Parameters — Umap 0.5 Documentation." n.d. Accessed July 28, 2022. https://umap-learn.readthedocs.io/en/latest/parameters.html.

Birtalan, Sara, Yingnan Zhang, Frederic A. Fellouse, Lihua Shao, Gabriele Schaefer, and Sachdev S. Sidhu. 2008. "The Intrinsic Contributions of Tyrosine, Serine, Glycine and Arginine to the Affinity and Specificity of Antibodies." *Journal of Molecular Biology*. https://doi.org/10.1016/j.jmb.2008.01.093.

Borowska, Marta T., Christopher T. Boughter, and Erin J. Adams. 2020. "Structural and Dynamic Elucidation of Natural Polyreactivity in Antibodies." *Biophysical Journal*. https://doi.org/10.1016/j.bpj.2019.11.2784.

Boughter, Christopher T., Marta T. Borowska, Jenna J. Guthmiller, Albert Bendelac, Patrick C. Wilson, Benoit Roux, and Erin J. Adams. 2020. "Biochemical Patterns of Antibody Polyreactivity Revealed through a Bioinformatics-Based Analysis of CDR Loops." *eLife* 9 (November). https://doi.org/10.7554/eLife.61393.

Breur, Tom. 2016. "Statistical Power Analysis and the Contemporary 'crisis' in Social Sciences." *Journal of Marketing Analytics* 4 (2): 61–65.

Bunker, Jeffrey J., Steven A. Erickson, Theodore M. Flynn, Carole Henry, Jason C. Koval, Marlies Meisel, Bana Jabri, Dionysios A. Antonopoulos, Patrick C. Wilson, and Albert Bendelac. 2017. "Natural Polyreactive IgA Antibodies Coat the Intestinal Microbiota." *Science* 358 (6361). https://doi.org/10.1126/science.aan6619.

Chiu, Mark L., Dennis R. Goulet, Alexey Teplyakov, and Gary L. Gilliland. 2019. "Antibody Structure and Function: The Basis for Engineering Therapeutics." *Antibodies (Basel, Switzerland)* 8 (4). https://doi.org/10.3390/antib8040055.

Contributors to Wikimedia projects. 2003. "Levenshtein Distance." Wikimedia Foundation, Inc. December 18, 2003. https://en.wikipedia.org/wiki/Levenshtein_distance.

Cunningham, Orla, Martin Scott, Zhaohui Sunny Zhou, and William J. J. Finlay. 2021. "Polyreactivity and Polyspecificity in Therapeutic Antibody Development: Risk Factors for Failure in Preclinical and Clinical Development Campaigns." *mAbs* 13 (1): 1999195.

Dietterich, T. G. 1998. "Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms." *Neural Computation* 10 (7): 1895–1923.

Dimitrov, Jordan D., Cyril Planchais, Lubka T. Roumenina, Tchavdar L. Vassilev, Srinivas V. Kaveri, and Sebastien Lacroix-Desmazes. 2013. "Antibody Polyreactivity in Health and Disease: Statu Variabilis." *The Journal of Immunology*. https://doi.org/10.4049/jimmunol.1300880.

Ditzel, H. J., K. Itoh, and D. R. Burton. 1996. "Determinants of Polyreactivity in a Large Panel of Recombinant Human Antibodies from HIV-1 Infection." *The Journal of Immunology* 157 (2): 739–49.

Dreiseitl, Stephan, and Lucila Ohno-Machado. 2002. "Logistic Regression and Artificial Neural Network Classification Models: A Methodology Review." *Journal of Biomedical Informatics* 35 (5-6): 352–59.

Earl, Dent, Ngan Nguyen, Glenn Hickey, Robert S. Harris, Stephen Fitzgerald, Kathryn Beal, Igor Seledtsov, et al. 2014. "Alignathon: A Competitive Assessment of Whole-Genome Alignment Methods." *Genome Research* 24 (12): 2077–89.

Eisen, Herman N. 2014. "Affinity Enhancement of Antibodies: How Low-Affinity Antibodies Produced Early in Immune Responses Are Followed by High-Affinity Antibodies Later and in Memory B-Cell Responses." *Cancer Immunology Research* 2 (5): 381–92.

Elgundi, Zehra, Mouhamad Reslan, Esteban Cruz, Vicki Sifniotis, and Veysel Kayser. 2017. "The State-of-Play and Future of Antibody Therapeutics." *Advanced Drug Delivery Reviews* 122 (December): 2–19.

Ferdous, S., and A. C. R. Martin. 2018. "AbDb: Antibody Structure Database-a Database of PDB-Derived Antibody Structures." *Database: The Journal of Biological Databases and Curation* 2018 (January). https://doi.org/10.1093/database/bay040.

Fernández-Quintero, Monica L., Johannes R. Loeffler, Johannes Kraml, Ursula Kahler, Anna S. Kamenik, and Klaus R. Liedl. 2019. "Characterizing the Diversity of the CDR-H3 Loop Conformational Ensembles in Relationship to Antibody Binding Properties." *Frontiers in Immunology* 0. https://doi.org/10.3389/fimmu.2018.03065.

Fernández-Quintero, Monica L., Nancy D. Pomarici, Barbara A. Math, Katharina B. Kroell, Franz Waibl, Alexander Bujotzek, Guy Georges, and Klaus R. Liedl. 2020. "Antibodies Exhibit Multiple Paratope States Influencing VH–VL Domain Orientations." *Communications Biology*. https://doi.org/10.1038/s42003-020-01319-z.

Frese, Katrin, Meike Eisenmann, Ralf Ostendorp, Bodo Brocks, and Stefan Pabst. 2013. "An Automated Immunoassay for Early Specificity Profiling of Antibodies." *mAbs* 5 (2): 279–87.

Fukushima, K. 1980. "Neocognitron: A Self Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position." *Biological Cybernetics* 36 (4): 193–202.

Gao, Linyi Alex, Max E. Wilkinson, Jonathan Strecker, Kira S. Makarova, Rhiannon K. Macrae, Eugene V. Koonin, and Feng Zhang. 2022. "Prokaryotic Innate Immunity through Pattern Recognition of Conserved Viral Proteins." *Science* 377 (6607): eabm4096.

Gilpin, Leilani H., David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. "Explaining Explanations: An Overview of Interpretability of Machine Learning." *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. https://doi.org/10.1109/dsaa.2018.00018.

Greiff, Victor, Ulrike Menzel, Enkelejda Miho, Cédric Weber, René Riedel, Skylar Cook, Atijeh Valai, et al. 2017. "Systems Analysis Reveals High Genetic and Antigen-Driven Predetermination of Antibody Repertoires throughout B Cell Development." *Cell Reports* 19 (7): 1467–78.

Gulcehre, Caglar, Kyunghyun Cho, Razvan Pascanu, and Yoshua Bengio. 2014. "Learned-Norm Pooling for Deep Feedforward and Recurrent Neural Networks." *Machine Learning and Knowledge Discovery in Databases*. https://doi.org/10.1007/978-3-662-44848-9_34.

Guthmiller, Jenna J., Linda Yu-Ling Lan, Monica L. Fernández-Quintero, Julianna Han, Henry

A. Utset, Dalia J. Bitar, Natalie J. Hamel, et al. 2020. "Polyreactive Broadly Neutralizing B Cells Are Selected to Provide Defense against Pandemic Threat Influenza Viruses." *Immunity* 53 (6): 1230–44.e5.
Harvey, Edward P., Jung-Eun Shin, Meredith A. Skiba, Genevieve R. Nemeth, Joseph D. Hurley, Alon Wellner, Ada Y. Shaw, et al. 2022. "An in Silico Method to Assess Antibody Fragment Polyreactivity." *bioRxiv*. https://doi.org/10.1101/2022.01.12.476085.
Hashimoto, G., P. F. Wright, and D. T. Karzon. 1983. "Antibody-Dependent Cell-Mediated Cytotoxicity against Influenza Virus-Infected Cells." *The Journal of Infectious Diseases* 148 (5): 785–94.
Jabbar, Haider Khalaf, and Rafiqul Zaman Khan. 2014. "Methods to Avoid Over-Fitting and Under-Fitting in Supervised Machine Learning (Comparative Study)." *Computer Science, Communication and Instrumentation Devices*. https://doi.org/10.3850/978-981-09-5247-1_017.
Jafari, Mohieddin, and Naser Ansari-Pour. 2019. "Why, When and How to Adjust Your P Values?" *Cell Journal* 20 (4): 604.
Jain, Tushar, Tingwan Sun, Stéphanie Durand, Amy Hall, Nga Rewa Houston, Juergen H. Nett, Beth Sharkey, et al. 2017. "Biophysical Properties of the Clinical-Stage Antibody Landscape." *Proceedings of the National Academy of Sciences of the United States of America* 114 (5): 944–49.
Janeway, Charles A., Jr, Paul Travers, Mark Walport, and Mark J. Shlomchik. 2001. "The Course of the Adaptive Response to Infection." In *Immunobiology: The Immune System in Health and Disease. 5th Edition*. Garland Science.
Jing, Xiaoyang, Qiwen Dong, Daocheng Hong, and Ruqian Lu. 2020. "Amino Acid Encoding Methods for Protein Sequences: A Comprehensive Review and Assessment." *IEEE/ACM Transactions on Computational Biology and Bioinformatics / IEEE, ACM* 17 (6): 1918–31.
Kapishnikov, Andrei, Subhashini Venugopalan, Besim Avci, Ben Wedin, Michael Terry, and Tolga Bolukbasi. 2021. "Guided Integrated Gradients: An Adaptive Path Method for Removing Noise." *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr46437.2021.00501.
Kaplon, Hélène, and Janice M. Reichert. 2018. "Antibodies to Watch in 2018." *mAbs* 10 (2): 183–203.
Kaur, Bani Preet, and Elizabeth Secord. 2019. "Innate Immunity." *Pediatric Clinics of North America* 66 (5): 905–11.
Kelly, Ryan L., Doris Le, Jessie Zhao, and K. Dane Wittrup. 2018. "Reduction of Nonspecificity Motifs in Synthetic Antibody Libraries." *Journal of Molecular Biology* 430 (1): 119–30.
Kelly, Ryan L., Tingwan Sun, Tushar Jain, Isabelle Caffry, Yao Yu, Yuan Cao, Heather Lynaugh, et al. 2015. "High Throughput Cross-Interaction Measures for Human IgG1 Antibodies Correlate with Clearance Rates in Mice." *mAbs* 7 (4): 770–77.
Kingsford, Carl, and Steven L. Salzberg. 2008. "What Are Decision Trees?" *Nature Biotechnology* 26 (9): 1011–13.
Klasse, P. J. 2014. "Neutralization of Virus Infectivity by Antibodies: Old Problems in New Perspectives." *Advances in Biology* 2014 (September). https://doi.org/10.1155/2014/157895.
Koide, Shohei, and Sachdev S. Sidhu. 2009. "The Importance of Being Tyrosine: Lessons in Molecular Recognition from Minimalist Synthetic Binding Proteins." *ACS Chemical Biology* 4 (5): 325–34.
Koo, Peter K., and Sean R. Eddy. 2019. "Representation Learning of Genomic Sequence Motifs with Convolutional Neural Networks." *PLoS Computational Biology* 15 (12): e1007560.
Labrousse, H., M. Adib-Conquy, and S. Avrameas. 1997. "Effect of Temperature on the Reactivities of Polyreactive and Monospecific Monoclonal IgG Antibodies." *Research in*

Immunology 148 (4): 267–76.
Lecerf, Maxime, Alexia Kanyavuz, Sébastien Lacroix-Desmazes, and Jordan D. Dimitrov. 2019. "Sequence Features of Variable Region Determining Physicochemical Properties and Polyreactivity of Therapeutic Antibodies." *Molecular Immunology* 112 (August): 338–46.
Libbrecht, Maxwell W., and William Stafford Noble. 2015. "Machine Learning Applications in Genetics and Genomics." *Nature Reviews Genetics*. https://doi.org/10.1038/nrg3920.
Libretexts. 2015. "Dissociation Constant." Chemistry LibreTexts. Libretexts. August 9, 2015. https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Supplemental_Modules_(Physical_and_Theoretical_Chemistry)/Equilibria/Chemical_Equilibria/Dissociation_Constant.
Lipton, Zachary C. 2018. "The Mythos of Model Interpretability." *Queue*. https://doi.org/10.1145/3236386.3241340.
Lu, Ruei-Min, Yu-Chyi Hwang, I-Ju Liu, Chi-Chiu Lee, Han-Zen Tsai, Hsin-Jung Li, and Han-Chung Wu. 2020. "Development of Therapeutic Antibodies for the Treatment of Diseases." *Journal of Biomedical Science* 27 (1): 1–30.
"Machine Learning in Bioinformatics: A Brief Survey and Recommendations for Practitioners." 2006. *Computers in Biology and Medicine* 36 (10): 1104–25.
Makowski, Emily K., Lina Wu, Alec A. Desai, and Peter M. Tessier. 2021. "Highly Sensitive Detection of Antibody Nonspecific Interactions Using Flow Cytometry." *mAbs* 13 (1): 1951426.
Mason, Derek M., Simon Friedensohn, Cédric R. Weber, Christian Jordi, Bastian Wagner, Simon M. Meng, Roy A. Ehling, et al. 2021. "Optimization of Therapeutic Antibodies by Predicting Antigen Specificity from Antibody Sequence via Deep Learning." *Nature Biomedical Engineering* 5 (6): 600–612.
Mietzner, Brun, Makoto Tsuiji, Johannes Scheid, Klara Velinzon, Thomas Tiller, Klaus Abraham, Jose B. Gonzalez, et al. 2008. "Autoreactive IgG Memory Antibodies in Patients with Systemic Lupus Erythematosus Arise from Nonreactive and Polyreactive Precursors." *Proceedings of the National Academy of Sciences*. https://doi.org/10.1073/pnas.0803644105.
Miglani, Vivek, Narine Kokhlikyan, Bilal Alsallakh, Miguel Martin, and Orion Reblitz-Richardson. 2020. "Investigating Saturation Effects in Integrated Gradients," October. https://doi.org/10.48550/arXiv.2010.12697.
Miyazawa, Sanzo, and Robert L. Jernigan. 1996. "Residue – Residue Potentials with a Favorable Contact Pair Term and an Unfavorable High Packing Density Term, for Simulation and Threading." *Journal of Molecular Biology*. https://doi.org/10.1006/jmbi.1996.0114.
Molnar, Charles, and Jane Gair. 2015. "23.2. Adaptive Immune Response." In *Concepts of Biology - 1st Canadian Edition*. BCcampus.
Molnar, Christoph. 2022a. "3.2 Taxonomy of Interpretability Methods." March 29, 2022. https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html.
———. 2022b. "5.2 Logistic Regression." March 29, 2022. https://christophm.github.io/interpretable-ml-book/logistic.html.
———. 2022c. "Chapter 5 Interpretable Models." In *Interpretable Machine Learning*.
Mouquet, Hugo, Johannes F. Scheid, Markus J. Zoller, Michelle Krogsgaard, Rene G. Ott, Shetha Shukair, Maxim N. Artyomov, et al. 2010. "Polyreactivity Increases the Apparent Affinity of Anti-HIV Antibodies by Heteroligation." *Nature* 467 (7315): 591–95.
Naqa, El, and Martin J. Murphy. 2015. "What Is Machine Learning?" *Machine Learning in Radiation Oncology*, 3–11.
"normalizedDifferenceOfProbabilities: Normalized Probability Differences in DiffLogo: DiffLogo: A Comparative Visualisation of Biooligomer Motifs." 2020. October 27, 2020.

https://rdrr.io/bioc/DiffLogo/man/normalizedDifferenceOfProbabilities.html.
Norman, Richard A., Francesco Ambrosetti, Alexandre M. J. J. Bonvin, Lucy J. Colwell, Sebastian Kelm, Sandeep Kumar, and Konrad Krawczyk. 2020. "Computational Approaches to Therapeutic Antibody Design: Established Methods and Emerging Trends." *Briefings in Bioinformatics* 21 (5): 1549–67.
Notkins, Abner Louis. 2004. "Polyreactivity of Antibody Molecules." *Trends in Immunology* 25 (4): 174–79.
Olson, Randal S., William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. 2017. "PMLB: A Large Benchmark Suite for Machine Learning Evaluation and Comparison." *BioData Mining* 10 (December): 36.
Ostrovsky-Berman, Miri, Boaz Frankel, Pazit Polak, and Gur Yaari. 2021. "Immune2vec: Embedding B/T Cell Receptor Sequences in ℝN Using Natural Language Processing." *Frontiers in Immunology* 0. https://doi.org/10.3389/fimmu.2021.680687.
"Pandas.get_dummies — Pandas 1.4.3 Documentation." n.d. Accessed August 2, 2022. https://pandas.pydata.org/docs/reference/api/pandas.get_dummies.html.
Parham, Peter. 2014. *The Immune System*. Garland Pub.
Pecetta, Simone, Oretta Finco, and Anja Seubert. 2020. "Quantum Leap of Monoclonal Antibody (mAb) Discovery and Development in the COVID-19 Era." *Seminars in Immunology* 50 (August): 101427.
Petti, Samantha, and Sean R. Eddy. 2022. "Constructing Benchmark Test Sets for Biological Sequence Analysis Using Independent Set Algorithms." *PLoS Computational Biology* 18 (3): e1009492.
Pittala, Srivamshi, and Chris Bailey-Kellogg. 2020. "Learning Context-Aware Structural Representations to Predict Antigen and Antibody Binding Interfaces." *Bioinformatics* 36 (13): 3996–4003.
Planchais, Cyril, Ayrin Kök, Alexia Kanyavuz, Valérie Lorin, Timothée Bruel, Florence Guivel-Benhassine, Tim Rollenske, et al. 2019. "HIV-1 Envelope Recognition by Polyreactive and Cross-Reactive Intestinal B Cells." *Cell Reports* 27 (2): 572–85.e7.
Prakash, Eva I., Avanti Shrikumar, and Anshul Kundaje. 2022. "Towards More Realistic Simulated Datasets for Benchmarking Deep Learning Models in Regulatory Genomics." In *Machine Learning in Computational Biology*, 58–77. PMLR.
Quang, Daniel, and Xiaohui Xie. 2016. "DanQ: A Hybrid Convolutional and Recurrent Deep Neural Network for Quantifying the Function of DNA Sequences." *Nucleic Acids Research* 44 (11): e107.
Rabia, Lilia A., Yulei Zhang, Seth D. Ludwig, Mark C. Julian, and Peter M. Tessier. 2018. "Net Charge of Antibody Complementarity-Determining Regions Is a Key Predictor of Specificity." *Protein Engineering, Design & Selection: PEDS* 31 (11): 409–18.
Raschka, Sebastian, and Vahid Mirjalili. 2019. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2, 3rd Edition*. Packt Publishing Ltd.
Ravetch, Jeffrey V., and Silvia Bolland. 2001. "IgG Fc Receptors." *Annual Review of Immunology*. https://doi.org/10.1146/annurev.immunol.19.1.275.
Raybould, Matthew I. J., Claire Marks, Konrad Krawczyk, Bruck Taddese, Jaroslaw Nowak, Alan P. Lewis, Alexander Bujotzek, Jiye Shi, and Charlotte M. Deane. 2019. "Five Computational Developability Guidelines for Therapeutic Antibody Profiling." *Proceedings of the National Academy of Sciences of the United States of America* 116 (10): 4025–30.
Reichenbach, Stephen E., Claudia A. Zini, Karine P. Nicolli, Juliane E. Welke, Chiara Cordero, and Qingping Tao. 2019. "Benchmarking Machine Learning Methods for Comprehensive Chemical Fingerprinting and Pattern Recognition." *Journal of Chromatography. A* 1595 (June): 158–67.
Robert, Philippe A., Rahmad Akbar, Robert Frank, Milena Pavlović, Michael Widrich, Igor

Snapkov, Maria Chernigovskaya, et al. 2022. "Unconstrained Generation of Synthetic Antibody-Antigen Structures to Guide Machine Learning Methodology for Real-World Antibody Specificity Prediction." *bioRxiv*. bioRxiv. https://doi.org/10.1101/2021.07.06.451258.

Robert, Philippe A., Theinmozhi Arulraj, and Michael Meyer-Hermann. 2021. ": A 3D Structural Affinity Model for Multi-Epitope Vaccine Simulations." *iScience* 24 (9): 102979.

Robert, Philippe A., Andrea L. J. Marschall, and Michael Meyer-Hermann. 2018. "Induction of Broadly Neutralizing Antibodies in Germinal Centre Simulations." *Current Opinion in Biotechnology*. https://doi.org/10.1016/j.copbio.2018.01.006.

Sarasua, Ignacio, Sebastian Pölsterl, and Christian Wachinger. n.d. "Hippocampal Representations for Deep Learning on Alzheimer's Disease." https://doi.org/10.21203/rs.3.rs-1124968/v1.

"Scipy.cluster.hierarchy.fcluster — SciPy v1.8.1 Manual." n.d. Accessed July 28, 2022. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.fcluster.html.

Shakib, Farouk. 2016. *The Human IgG Subclasses: Molecular Analysis of Structure, Function and Regulation*. Elsevier.

Sharma, Siddharth, Simone Sharma, and Anidhya Athaiya. 2020. "ACTIVATION FUNCTIONS IN NEURAL NETWORKS." *International Journal of Engineering Applied Sciences and Technology*. https://doi.org/10.33564/ijeast.2020.v04i12.054.

Sharma, Vikas K., Thomas W. Patapoff, Bruce Kabakoff, Satyan Pai, Eric Hilario, Boyan Zhang, Charlene Li, et al. 2014. "In Silico Selection of Therapeutic Antibodies for Development: Viscosity, Clearance, and Chemical Stability." *Proceedings of the National Academy of Sciences of the United States of America* 111 (52): 18601–6.

"sklearn.cluster.DBSCAN." n.d. Scikit-Learn. Accessed August 13, 2022. https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

"Sklearn.metrics.silhouette_score." n.d. Scikit-Learn. Accessed May 2, 2022. https://scikit-learn/stable/modules/generated/sklearn.metrics.silhouette_score.html.

Starr, Charles G., and Peter M. Tessier. 2019. "Selecting and Engineering Monoclonal Antibodies with Drug-like Specificity." *Current Opinion in Biotechnology* 60 (December): 119–27.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. 2017. "Axiomatic Attribution for Deep Networks." http://arxiv.org/abs/1703.01365.

Tang, Cheng, Damien Garreau, and Ulrike von Luxburg. 2018. "When Do Random Forests Fail?" *Advances in Neural Information Processing Systems* 31. https://proceedings.neurips.cc/paper/2018/file/204da255aea2cd4a75ace6018fad6b4d-Paper.pdf.

Techopedia. 2017. "Ground Truth." Techopedia.com. Techopedia. May 9, 2017. http://www.techopedia.com/definition/32514/ground-truth.

Thiyagalingam, Jeyan, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. 2022. "Scientific Machine Learning Benchmarks." *Nature Reviews Physics*. https://doi.org/10.1038/s42254-022-00441-7.

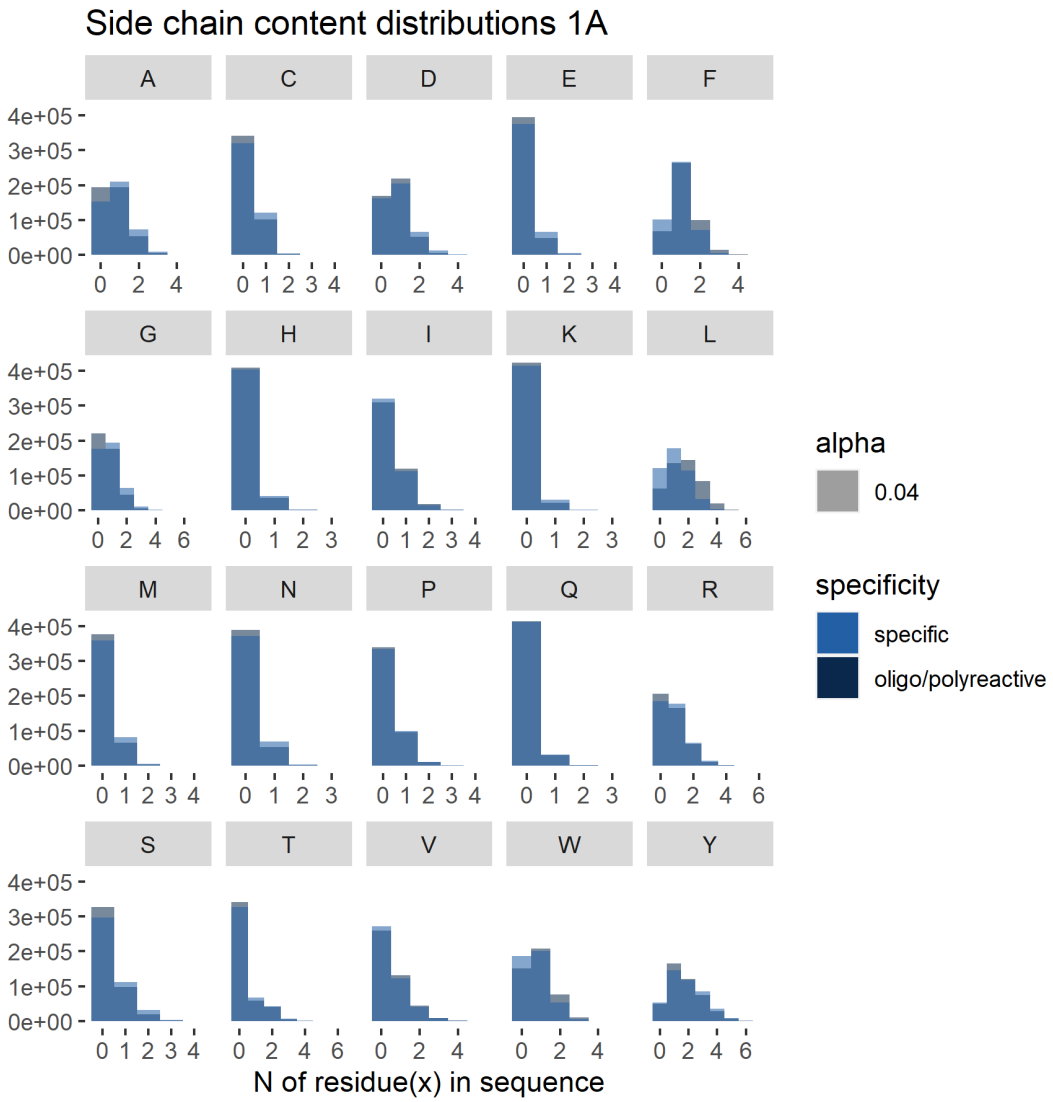
Thompson, Rebecca S., Noor M. Khaskhely, Kristin R. Malhotra, David J. Leggat, Jason Mosakowski, Sadik Khuder, Gary R. McLean, and M. A. Julie Westerink. 2012. "Isolation and Characterization of Human Polyreactive Pneumococcal Polysaccharide Antibodies." *Open Journal of Immunology*. https://doi.org/10.4236/oji.2012.23012.

Tiller, Kathryn E., Lijuan Li, Sandeep Kumar, Mark C. Julian, Shekhar Garde, and Peter M. Tessier. 2017. "Arginine Mutations in Antibody Complementarity-Determining Regions Display Context-Dependent Affinity/specificity Trade-Offs." *The Journal of Biological Chemistry* 292 (40): 16638–52.

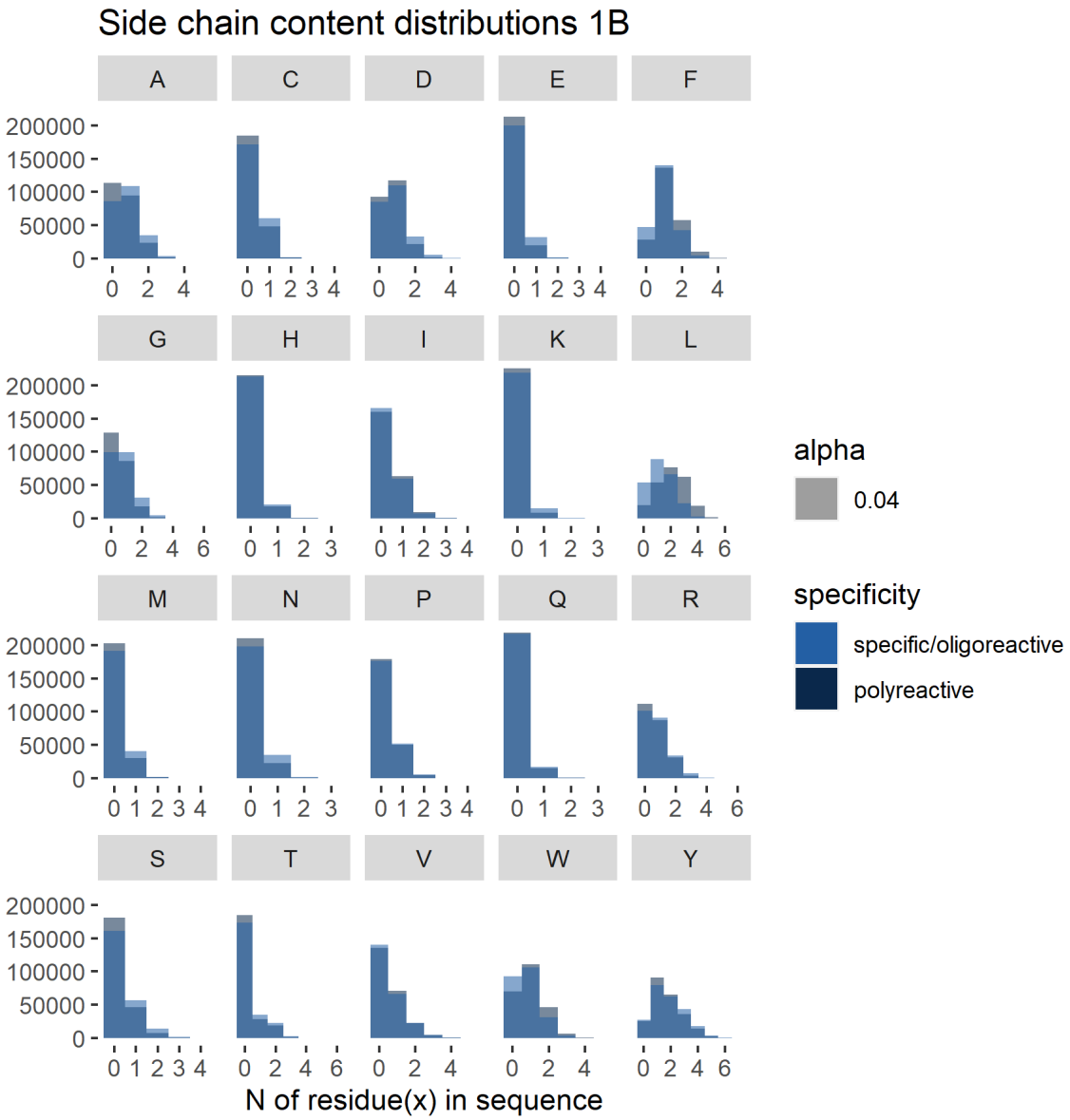
Trabelsi, Ameni, Mohamed Chaabane, and Asa Ben-Hur. 2019. "Comprehensive Evaluation

of Deep Learning Architectures for Prediction of DNA/RNA Sequence Binding Specificities." *Bioinformatics* 35 (14): i269–77.
Tsang, Michael, Dehua Cheng, and Yan Liu. 2017. "Detecting Statistical Interactions from Neural Network Weights," May. https://doi.org/10.48550/arXiv.1705.04977.
Tu, J. V. 1996. "Advantages and Disadvantages of Using Artificial Neural Networks versus Logistic Regression for Predicting Medical Outcomes." *Journal of Clinical Epidemiology* 49 (11): 1225–31.
Urquhart, Lisa. 2019. "Top Drugs and Companies by Sales in 2018." *Nature Reviews. Drug Discovery*, March. https://doi.org/10.1038/d41573-019-00049-0.
Victora, Gabriel D., and Michel C. Nussenzweig. 2022. "Germinal Centers." *Annual Review of Immunology*. https://doi.org/10.1146/annurev-immunol-120419-022408.
Wardemann, Hedda, and Christian E. Busse. 2017. "Novel Approaches to Analyze Immunoglobulin Repertoires." *Trends in Immunology* 38 (7): 471–82.
Wardemann, Hedda, Sergey Yurasov, Anne Schaefer, James W. Young, Eric Meffre, and Michel C. Nussenzweig. 2003. "Predominant Autoantibody Production by Early Human B Cell Precursors." *Science*. https://doi.org/10.1126/science.1086907.
Weber, Cédric R., Rahmad Akbar, Alexander Yermanos, Milena Pavlović, Igor Snapkov, Geir K. Sandve, Sai T. Reddy, and Victor Greiff. 2020. "immuneSIM: Tunable Multi-Feature Simulation of B- and T-Cell Receptor Repertoires for Immunoinformatics Benchmarking." *Bioinformatics* 36 (11): 3594–96.
Wei, Zhen, Jingyi Zhang, Li Liu, Fan Zhu, Fumin Shen, Yi Zhou, Si Liu, Yao Sun, and Ling Shao. 2019. "Building Detail-Sensitive Semantic Segmentation Networks With Polynomial Pooling." *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/cvpr.2019.00728.
Xu, Y., W. Roach, T. Sun, T. Jain, B. Prinz, T-Y Yu, J. Torrey, et al. 2013. "Addressing Polyspecificity of Antibodies Selected from an in Vitro Yeast Presentation System: A FACS-Based, High-Throughput Selection and Analytical Tool." *Protein Engineering Design and Selection*. https://doi.org/10.1093/protein/gzt047.
Yin, Shi, Chao Liu, Zhiyong Zhang, Yiye Lin, Dong Wang, Javier Tejedor, Thomas Fang Zheng, and Yinguo Li. 2015. "Noisy Training for Deep Neural Networks in Speech Recognition." *EURASIP Journal on Audio, Speech, and Music Processing*. https://doi.org/10.1186/s13636-014-0047-0.
Zhang, Xiuling, Kailun Wei, Xuenan Kang, and Jinxiang Li. 2021. "Hybrid Nonlinear Convolution Filters for Image Recognition." *Applied Intelligence*. https://doi.org/10.1007/s10489-020-01845-7.
Zhou, Victor. 2020. "Keras for Beginners: Implementing a Convolutional Neural Network." Victor Zhou. November 10, 2020. https://victorzhou.com/blog/keras-cnn-tutorial/.
Zorn, Emmanuel, and Sarah B. See. 2017. "Polyreactive Natural Antibodies in Transplantation." *Current Opinion in Organ Transplantation*. https://doi.org/10.1097/mot.0000000000000376.

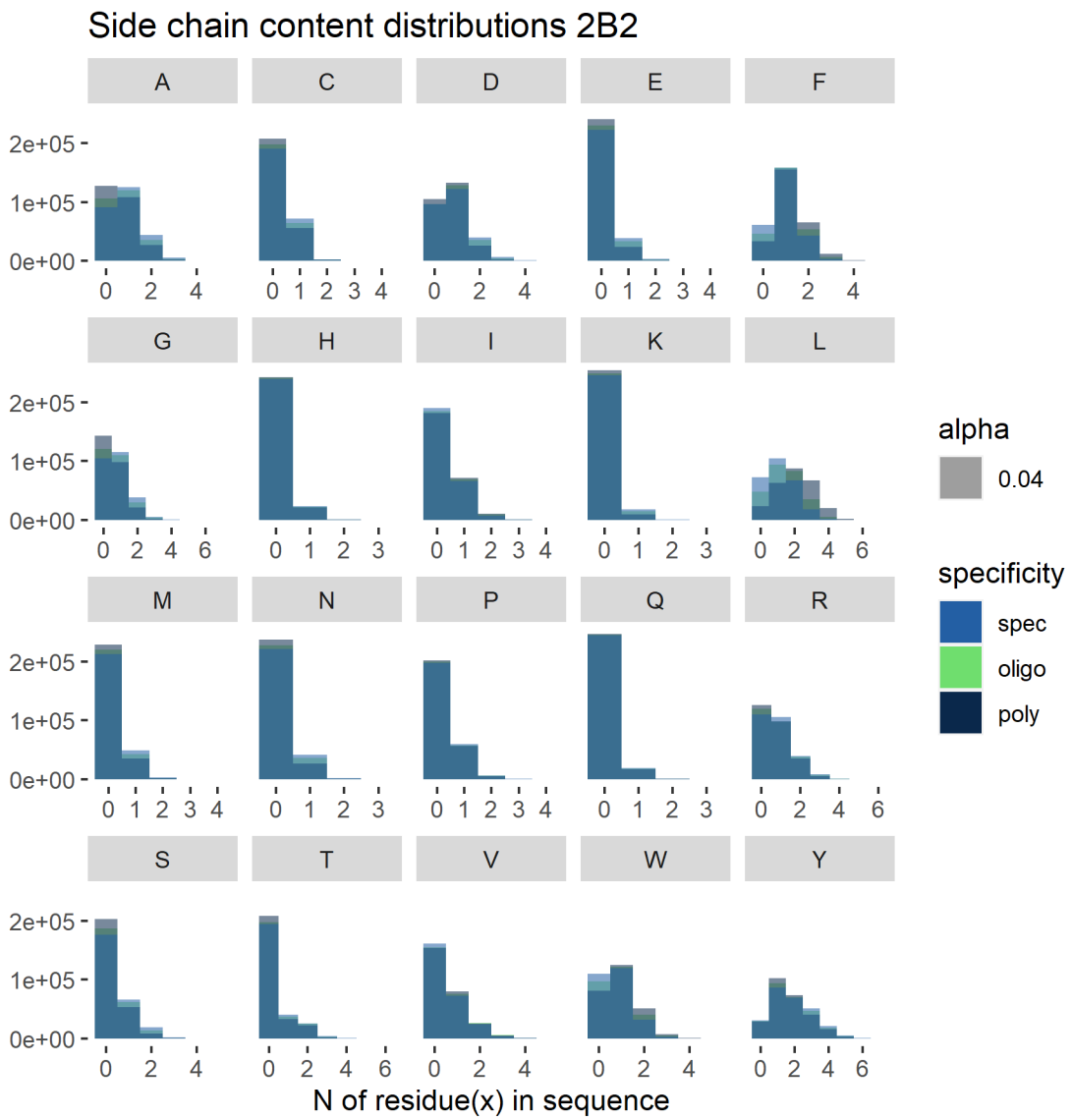
11 Appendix



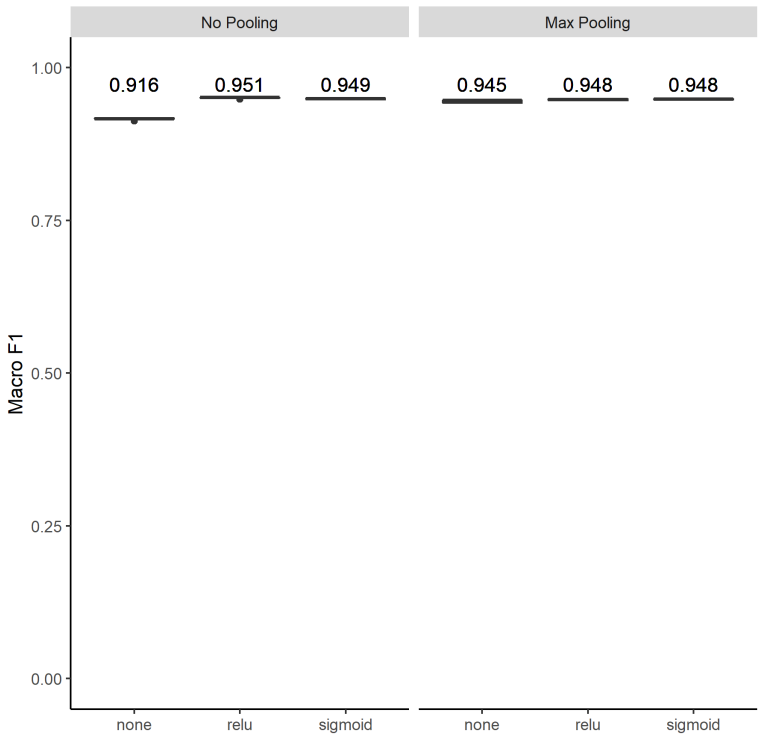
Supplementary Figure 1: **Distribution of Amino Acids occurrences within sequences of each class per task "strict" polyreactivity (1A).**



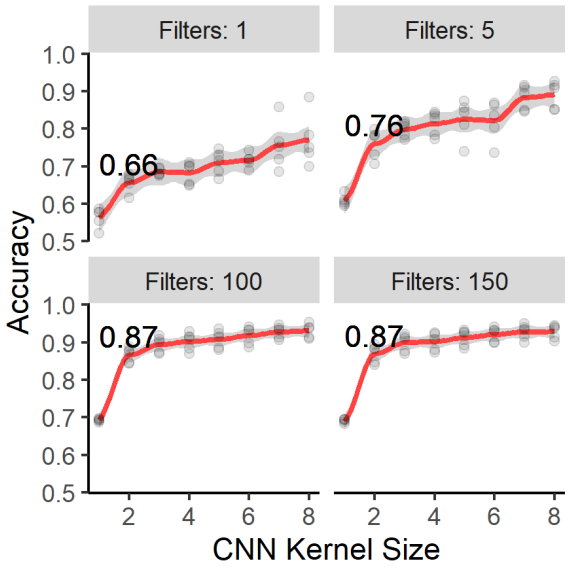
Supplementary Figure 2: **Distribution of Amino Acids occurrences within sequences of each class per task "non-strict" polyreactivity (1B).**



Supplementary Figure 3: **Distribution of Amino Acids occurrences within sequences of each class defined by task "coarse grained multiclass" (2B2).**



Supplementary Figure 4: **Without max pooling or activation function the CNN model performance decreased and was comparable to logistic regression.** Results of 5-fold cross validation of a CNN model with one hidden layer. The models differed in whether max pooling was deployed or the activation function used in the convolutional layer.



Supplementary Figure 5: **Accuracy depending on CNN kernel max length on data with balanced sequence ends.**Accuracy based on 5 fold cross validation evaluating accuracy on data where the sequence end motifs where balanced. Performed in a similar fashion as previous results included under Results, ReLu activation function was used for the convolutional layer.

Supplementary Table 1: **Expected occurrence of each amino acid in sequences of the given class per task "strict polyreactivity.**

Amino Acid	poly reactive	specific	Task
A	0,706	0,865	strict polyreactivity
C	0,242	0,288	strict polyreactivity
D	0,766	0,843	strict polyreactivity
E	0,119	0,171	strict polyreactivity
F	1,141	0,958	strict polyreactivity
G	0,626	0,797	strict polyreactivity
H	0,086	0,096	strict polyreactivity
I	0,351	0,313	strict polyreactivity
K	0,049	0,072	strict polyreactivity
L	1,711	1,139	strict polyreactivity
M	0,161	0,205	strict polyreactivity
N	0,127	0,171	strict polyreactivity
P	0,262	0,278	strict polyreactivity
Q	0,071	0,078	strict polyreactivity
R	0,729	0,809	strict polyreactivity
S	0,316	0,423	strict polyreactivity
T	0,345	0,393	strict polyreactivity
V	0,564	0,522	strict polyreactivity
W	0,882	0,729	strict polyreactivity
Y	1,748	1,847	strict polyreactivity

Supplementary Table 2: **Expected occurrence of each amino acid in sequences of the given class per task "non_strict polyreactivity"**

Amino Acid	polyreactive	specific	Task
A	0,631	0,815	non-strict polyreactivity
C	0,218	0,274	non-strict polyreactivity
D	0,710	0,829	non-strict polyreactivity
E	0,092	0,156	non-strict polyreactivity
F	1,224	1,014	non-strict polyreactivity
G	0,542	0,747	non-strict polyreactivity
H	0,082	0,092	non-strict polyreactivity
I	0,363	0,328	non-strict polyreactivity
K	0,035	0,065	non-strict polyreactivity
L	2,058	1,287	non-strict polyreactivity
M	0,141	0,192	non-strict polyreactivity
N	0,102	0,160	non-strict polyreactivity
P	0,255	0,272	non-strict polyreactivity
Q	0,065	0,076	non-strict polyreactivity
R	0,689	0,786	non-strict polyreactivity
S	0,263	0,391	non-strict polyreactivity
T	0,305	0,384	non-strict polyreactivity

V	0,562	0,541	non-strict polyreactivity
W	0,956	0,773	non-strict polyreactivity
Y	1,707	1,817	non-strict polyreactivity

Supplementary Table 3: **Expected occurrence of each amino acid in sequences of the given class per task "coarse grained multiclass"**

Amino Acid	oligo	poly	spec
A	0,760	0,639	0,866
C	0,259	0,221	0,288
D	0,808	0,716	0,843
E	0,138	0,096	0,170
F	1,081	1,215	0,957
G	0,687	0,553	0,798
H	0,088	0,083	0,096
I	0,343	0,362	0,313
K	0,058	0,037	0,072
L	1,457	2,016	1,140
M	0,176	0,143	0,206
N	0,146	0,105	0,171
P	0,267	0,256	0,278
Q	0,075	0,066	0,078
R	0,758	0,695	0,810
S	0,353	0,270	0,423
T	0,374	0,311	0,393
V	0,566	0,562	0,522
W	0,827	0,945	0,730
Y	1,778	1,710	1,848

Supplementary Table 4: **Expected occurrence of each amino acid in sequences of the given class per task "finer grained multiclass"**

Amino Acid	class 1	class 2	class 3	class 4	class 5
A	0,863	0,786	0,739	0,690	0,597
C	0,289	0,265	0,255	0,241	0,205
D	0,843	0,827	0,792	0,754	0,685
E	0,170	0,145	0,133	0,114	0,082
F	0,958	1,051	1,108	1,163	1,258
G	0,794	0,711	0,668	0,618	0,501
H	0,097	0,089	0,088	0,084	0,081
I	0,312	0,331	0,353	0,359	0,365
K	0,070	0,063	0,055	0,046	0,030
L	1,141	1,360	1,538	1,771	2,216
M	0,204	0,184	0,169	0,153	0,135
N	0,173	0,155	0,138	0,121	0,093
P	0,279	0,269	0,265	0,265	0,249

Q	0,079	0,077	0,073	0,073	0,061
R	0,814	0,768	0,749	0,730	0,665
S	0,423	0,371	0,339	0,306	0,239
T	0,390	0,376	0,372	0,346	0,281
V	0,522	0,556	0,576	0,571	0,553
W	0,727	0,814	0,840	0,880	0,998
Y	1,852	1,803	1,752	1,716	1,707

Supplementary Table 5: **Mean and standard deviation of Macro f1 scores for each model evaluated by 10-fold cross-validation.** Model Abreviations: LR = Logistic Regression, DT = Decision Tree, RF = Random Forest, CNN = Convolutional Neural Network, FNN = Feed Forward Neural Network.

Model	Mean Macro F1	Standard Deviation
FNN	0.950	0.00202
CNN	0.946	0.00309
LR onehot non-strict	0.919	0.00115
DT onehot non-strict	0.812	0.00152
RF onehot non-strict	0.850	0.00162
LR AA composition non-strict	0.896	0.00128
DT AA composition non-strict	0.887	0.00174
RF AA composition non-strict	0.884	0.00123
LR onehot strict	0.828	0.00213
DT onehot strict	0.773	0.00172
RF onehot strict	0.812	0.00117
LR AA composition strict	0.809	0.00209
DT AA composition strict	0.802	0.00186
RF AA composition strict	0.790	0.00134
LR onehot coarse grained multiclass	0.757	0.00191
DT onehot coarse grained multiclass	0.665	0.00226
RF onehot coarse grained multiclass	0.710	0.00188
LR AA composition coarse grained multiclass	0.690	0.00165
DT AA composition coarse grained multiclass	0.711	0.00109
RF AA composition coarse grained multiclass	0.693	0.00174

LR onehot finer grained multiclass	0.597	0.00177
DT onehot finer grained multiclass	0.492	0.00134
RF onehot finer grained multiclass	0.506	0.00223
LR AA composition finer grained multiclass	0.546	0.00209
DT AA composition finer grained multiclass	0.535	0.00217
RF AA composition finer grained multiclass	0.512	0.00191

Supplementary Table 6: **Results of Mcnemar's test of significance between models for task strict polyreactivity.**
The Architecture 1 and Architecture 2 columns denote the architecture used to train the tested model for a Mcneamar test comparing pairs of classifiers. The Encoding X columns denote the encoding of the sequences which the model was trained on.

Architecture 1	Encoding 1	Architecture 2	Encoding 2	Chi squared	Raw p-values	Adjusted p-value
DecisionTreeClassifier	aacomp	DecisionTreeClassifier	onehot	813,579	6,02E−179	9,03E−178
DecisionTreeClassifier	aacomp	LogisticRegression	aacomp	390,323	7,04E−87	1,06E−85
DecisionTreeClassifier	aacomp	LogisticRegression	onehot	1928,509	0	0
DecisionTreeClassifier	aacomp	RandomForestClassifier	aacomp	206,805	6,84E−47	1,03E−45
DecisionTreeClassifier	aacomp	RandomForestClassifier	onehot	196,422	1,26E−44	1,89E−43
DecisionTreeClassifier	onehot	LogisticRegression	aacomp	1542,857	0	0
DecisionTreeClassifier	onehot	LogisticRegression	onehot	3474,071	0	0
DecisionTreeClassifier	onehot	RandomForestClassifier	aacomp	359,133	4,35E−80	6,52E−79
DecisionTreeClassifier	onehot	RandomForestClassifier	onehot	3210,532	0	0
LogisticRegression	aacomp	LogisticRegression	onehot	965,554	5,52E−212	8,28E−211
LogisticRegression	aacomp	RandomForestClassifier	aacomp	929,546	3,71E−204	5,56E−203
LogisticRegression	aacomp	RandomForestClassifier	onehot	6,288	0,01215531825	0,1823297737
LogisticRegres	onehot	RandomForestClas	aacomp	2777,306	0	0

sion		sifier				
LogisticRegression	onehot	RandomForestClassifier	onehot	303,859	4,75E−68	7,13E−67
RandomForestClassifier	aacomp	RandomForestClassifier	onehot	542,025	6,84E−120	1,03E−118

Supplementary Table 7: **Results of Mcnemar's test of significance between models for task “non-strict” polyreactivity.**
The Architecture 1 and Architecture 2 columns denote the architecture used to train the tested model for a Mcneamar test comparing pairs of classifiers. The Encoding X columns denote the encoding of the sequences which the model was trained on.

Architecture 1	Encoding 1	Architecture 2	Encoding 2	Chi squared	Raw p-value	Adjusted p-value
CNN	onehot	DecisionTreeClassifier	aacomp	10139,633	0	0
CNN	onehot	DecisionTreeClassifier	onehot	23520,048	0	0
CNN	onehot	FNN	onehot	136,064	1,93E−31	5,41E−30
CNN	onehot	LogisticRegression	aacomp	7933,443	0	0
CNN	onehot	LogisticRegression	onehot	3501,552	0	0
CNN	onehot	RandomForestClassifier	aacomp	10868,404	0	0
CNN	onehot	RandomForestClassifier	onehot	16807,886	0	0
DecisionTreeClassifier_aacomp	aacomp	DecisionTreeClassifier	onehot	5024,209	0	0
DecisionTreeClassifier_aacomp	aacomp	FNN	onehot	11452,606	0	0
DecisionTreeClassifier	aacomp	LogisticRegression	aacomp	492,690	3,70E−109	1,04E−107
DecisionTreeClassifier	aacomp	LogisticRegression	onehot	3577,668	0	0
DecisionTreeClassifier	aacomp	RandomForestClassifier	aacomp	59,972	9,62E−15	2,69E−13
DecisionTreeClassifier	aacomp	RandomForestClassifier	onehot	1571,169	0	0
DecisionTreeClassifier	onehot	FNN	onehot	24900,089	0	0
DecisionTreeClassifier	onehot	LogisticRegression	aacomp	6821,102	0	0

DecisionTreeClassifier	onehot	LogisticRegression	onehot	12791,464	0	0
DecisionTreeClassifier	onehot	RandomForestClassifier	aacomp	4357,845	0	0
DecisionTreeClassifier	onehot	RandomForestClassifier	onehot	2171,819	0	0
FNN	onehot	LogisticRegression	aacomp	9268,778	0	0
FNN	onehot	LogisticRegression	onehot	4560,276	0	0
FNN	onehot	RandomForestClassifier	aacomp	12290,098	0	0
FNN	onehot	RandomForestClassifier	onehot	17985,734	0	0
LogisticRegression	aacomp	LogisticRegression	onehot	2184,440	0	0
LogisticRegression	aacomp	RandomForestClassifie	aacomp	748,499	8,51E−165	2,38E−163
LogisticRegression	aacomp	RandomForestClassifier	onehot	2731,735	0	0
LogisticRegression	onehot	RandomForestClassifier	aacomp	4068,617	0	0
LogisticRegression	onehot	RandomForestClassifier	onehot	7307,576	0	0
RandomForestClassifier	aacomp	RandomForestClassifier	onehot	1203,926	8,55E−264	2,39E−262

Supplementary Table 8: **Results of Mcnemar's test of significance between models for task “coarse” multiclass.**
The Architecture 1 and Architecture 2 columns denote the architecture used to train the tested model for a Mcneamar test comparing pairs of classifiers. The Encoding X columns denote the encoding of the sequences which the model was trained on.

Architecture 1	Encoding 1	Architecture 2	Encoding 2	Chi squared	Raw p-value	Adjusted p-value
DecisionTreeClassifier	aacomp	DecisionTreeClassifier	onehot	1091,023861	2,95E−239	4,42E−238
DecisionTreeClassifier	aacomp	LogisticRegression	aacomp	294,4213111	5,41E−66	8,12E−65
DecisionTreeClassifier	aacomp	LogisticRegression	onehot	2719,710161	0	0
DecisionTreeClassifier	aacomp	RandomForestClassifier	aacomp	481,2830269	1,12E−106	1,69E−105
DecisionTreeClassifier	aacomp	RandomForestClassifier	onehot	41,01088074	1,51E−10	2,27E−09

DecisionTreeClassifier	onehot	LogisticRegression	aacomp	1812,482246	0	0
DecisionTreeClassifier	onehot	LogisticRegression	onehot	4976,406069	0	0
DecisionTreeClassifier	onehot	RandomForestClassifier	aacomp	348,0882523	1,11E−77	1,66E−76
DecisionTreeClassifier	onehot	RandomForestClassifier	onehot	2980,106766	0	0
LogisticRegression	aacomp	LogisticRegression	onehot	1837,921468	0	0
LogisticRegression	aacomp	RandomForestClassifier	aacomp.csv	1177,382424	5,02E−258	7,53E−257
LogisticRegression	aacomp	RandomForestClassifier	onehot	14,00608033	0,000182220425 8	0,002733306387
LogisticRegression	onehot	RandomForestClassifier	aacomp	4274,113006	0	0
LogisticRegression	onehot	RandomForestClassifier	onehot	1050,65239	1,76E−230	2,63E−229
RandomForestClassifier	aacomp	RandomForestClassifier	onehot	426,7186	8,42E−95	1,26E−93

Supplementary Table 9: **Results of Mcnemar's test of significance between models for task ”finer-grained” multiclass.**
The Architecture 1 and Architecture 2 columns denote the architecture used to train the tested model for a Mcneamar test comparing pairs of classifiers. The Encoding columns denote the encoding of the sequences which the model was trained on.

Architecture 1	Encoding 1	Architecture 2	Encoding 2	Chi squared	Raw p-value	Adjusted p-value
DecisionTreeClassifier	aacomp	DecisionTreeClassifier	onehot	651,673	9,67E−144	1,45E−142
DecisionTreeClassifier	aacomp	LogisticRegression	aacomp	1038,184	9,01E−228	1,35E−226
DecisionTreeClassifier	aacomp	LogisticRegression	onehot	4619,440	0,00E+00	0
DecisionTreeClassifier	aacomp	RandomForestClassifier	aacomp	290,583	3,71E−65	5,57E−64
DecisionTreeClassifier	aacomp	RandomForestClassifier	onehot	390,503	6,43E−87	9,65E−86
DecisionTreeClassifier	onehot	LogisticRegression	aacomp	2213,181	0,00E+00	0
DecisionTreeClassifier	onehot	LogisticRegression	onehot	6233,839	0,00E+00	0
DecisionTreeClassifier	onehot	RandomForestClassifier	aacomp	175,758	4,09E−40	6,13E−39

DecisionTreeClassifier	onehot	RandomForestClassifier	onehot	3851,062	0,00E+00	0
LogisticRegression	aacomp	LogisticRegression	onehot	2093,235	0,00E+00	0
LogisticRegression	aacomp	RandomForestClassifier	aacomp	1910,595	0,00E+00	0
LogisticRegression	aacomp	RandomForestClassifier	onehot	7,830	5,14E−03	0,07708313192
LogisticRegression	onehot	RandomForestClassifier	aacomp	6051,104	0,00E+00	0
LogisticRegression	onehot	RandomForestClassifier	onehot	1262,561	1,55E−276	2,32E−275
RandomForestClassifier	aacomp	RandomForestClassifier	onehot	1007,989	3,29E−221	4,94E−220

Norges miljø- og biovitenskapelige universitet
Noregs miljø- og biovitskapelege universitet
Norwegian University of Life Sciences

Postboks 5003
NO-1432 Ås
Norway