Utilising Machine Learning Techniques on Simulated Viral Evolution Datasets to Improve Viral Recombinant Identification

By

Joshua Cullinan

CLLJOS001

Submitted to the **University of Cape Town**

In fulfilment of the requirements for the degree

MSc Bioinformatics

Faculty of Health Sciences

UNIVERSITY OF CAPE TOWN

Date of Submission: 29/01/2025

Supervisor: Professor Darren Martin (Department of Computational biology, UCT)

Declaration

I, Joshua Cullinan, hereby declare that the work on which this dissertation/thesis is based is my original work (except where acknowledgements indicate otherwise) and that neither the whole work nor any part of it has been, is being, or is to be submitted for another degree in this or any other university.
I empower the university to reproduce for the purpose of research either the whole or any portion of the contents in any manner whatsoever.

Signature: Signed by candidate **…………………………………….**

Original Submission Date: 29/01/2025

Library Submission Date: 24/04/2025

Table of Contents

Abstract

This thesis explores machine learning applications for enhancing viral recombination detection. Using SANTA-SIM-generated viral evolution data, multiple computational approaches were developed and evaluated against existing methods in the Recombination Detection Program (RDP5). The study trained and tested several models, including logistic regression, gradient boosting, random forests and neural networks, on a dataset of 491 124 sequences. A novel neural network architecture employing position selection achieved the highest performance with a weighted Area Under Curve (AUC) of 0.784, surpassing RDP5's baseline AUC of 0.739. The gradient boosting classifier demonstrated strong results with an AUC of 0.765, whilst the binary neural network achieved 0.764.

Performance evaluation focused on precision, recall and F1-scores to address the inherent class imbalance between recombinant and parental sequences. The models demonstrated modest performance in detecting recombinants (precision 0.627-0.687, recall 0.652-0.686). These improvements, though incremental, represent progress in automated recombination detection. The successful preliminary integration of the logistic regression model into RDP5 demonstrates the practical applicability of these approaches. This work provides a foundation for enhancing viral recombination detection through machine learning, whilst highlighting areas requiring further development to achieve more substantial improvements in detection accuracy.

Introduction and Literature Review

Viruses

In the systematic classification of life, organisms are typically grouped into the domains of Bacteria, Archaea, and Eukarya. However, viruses occupy a distinct position, challenging traditional biological categorisation. While there is debate among biologists regarding their classification as "living" due to their dependence on host cells for replication, the significant impact of viruses on various life forms is evident. Characterised by a fundamental structure, primarily a protein coat surrounding their genetic material, viruses have developed specialised strategies for replication and propagation.

Within this distinct category, the diversity of viral genomic structures stands out. Contrary to the relatively consistent genomes of cellular organisms, viruses exhibit a wide array of genetic configurations. The majority of known viruses have RNA genomes, but many utilise DNA or, in some cases, have genomes consisting of both RNA and DNA. Virus genomes can be single stranded (ss) or double stranded (ds), and, whilst rare, viruses in some families have genomes that are predominantly double stranded but with single stranded regions (e.g., Hepadnaviridae). Furthermore, viruses with single stranded RNA genomes may also be classified according to whether their genomic RNAs can be directly translated; referred to as positive (+) sense, or whether proteins can only be translated from a complementary RNA strand, referred to as negative sense (-). Some ssRNA viruses have ambisense genomes with proteins being translated from both the genome strand and its complement, referred to as +/- sense.

In certain cases viruses have adopted complex replication mechanisms beyond the classical mechanisms. Retroviruses, for instance, have RNA genomes but utilise reverse transcriptase enzymes to reverse-transcribe genomic RNA molecules into a DNA intermediate during host infection, allowing integration into the host genome and facilitating replication via host-mediated transcription during cell division.

The extensive genomic diversity of viruses plays a pivotal role in their ability to infect a diverse variety of hosts, circumvent immune defences, and adapt to varied environments. These properties are used to classify viruses into seven groups defined by David Baltimore in 1971 (1) (*table 1.1*).

Table 1.1 The Baltimore Classification of Viruses (1)		
Group	**Classification**	**Example**
Group 1	dsDNA	Herpesviruses
Group 2	ssDNA	Parvoviruses
Group 3	dsRNA	Reoviruses
Group 4	(+)ssRNA	Coronaviruses
Group 5	(-)ssRNA	Orthomyxovirus
Group 6	ssRNA-RT (Reverse Transcriptase)	Human Immunodeficiency Virus
Group 7	dsDNA-RT	Hepadnaviridae

Mutations and Genetic Diversity

The diverse structural variation of viruses is further complemented by the diversity of mechanisms by which viruses undergo genetic change. Mutation is a primary source of genetic variation in all organisms, and viruses are no exception. Due to the lack of proof-reading mechanisms in viral replication processes, especially in RNA viruses, mutation rates can be exceptionally high. Point mutations: deletions and insertions are common types of mutations affecting single base pairs.

Many DNA viruses (particularly those with dsDNA genomes) have a far lower mutation rate than RNA viruses, due to the higher fidelity of their replicative enzymes (which are generally viral or host encoded high fidelity DNA polymerases) (2). This accounts for one explanation as to why DNA virus genomes tend to be longer than those of RNA viruses, in that the higher fidelity of DNA viruses replication reduces the likelihood of multiple lethal/sublethal deleterious mutations arising in a single replication cycle.

RNA viruses are thus subject to a size cap where the risk of accumulating multiple mutations per replication cycle increases with the length of the genome. There is a point where the advantage of having more encoded information is outweighed by the risk of acquiring so many mutations per replication cycle that this information cannot be reliably retained per replication cycle (3). Put simply, the retention of genetic information requires that, with each replication cycle, a substantial proportion of progeny genomes be identical to the parental genomes from which they were copied.

Point mutations affecting a single base pair are not the only type of mutation; entire genome segments can undergo duplication and deletion contributing to genetic diversity. Duplication events copy genome segments, potentially leading to evolutionary diversification by providing opportunities for variations to occur in one copy whilst retaining an original unaltered segment. Deletion events remove entire segments, from one to a few hundred nucleotides in length, potentially streamlining the genome for benefits like faster replication or immune evasion.

Viral Recombination

Viral recombination is a common method by which viruses may diversify and undergo significant genetic change. Recombination is a process initiated when two related viral genomes, concurrently infect a host cell. During the replication phase, an exchange of genetic material between these genomes takes place, leading to the creation of a progeny genome that carries a combination of genetic material from both parental viruses. If the parent genomes were identical there would be no evidence of any recombination having occurred. If the parental genomes were not identical, however, recombination can introduce significant genetic variation, especially if the parental viruses are highly diverged (4).

Typically, recombination is expected to occur most frequently between viruses of the same species strain or type since these will share highly similar/identical host-ranges and/or cellular tropisms. It can and does also commonly occur between viruses in different species, genera or even families: the main constraint being that these diverse species all meet, at least occasionally, within some shared host and cell type. Interestingly, the hybrid progeny of recombination between such diverse viruses can even possess new genes, which were not present in either of the parent viruses (2).

Viral recombination is mechanistically classified according to where the crossover site is located on the parent strands; the two distinct classes are homologous recombination and non-homologous recombination both of which can occur at the same time (5,6). Homologous recombination is the most prevalent form (7). In homologous recombination the co-infecting viral genomes are recombined using information from the same or nearly the same points on the parent genomes.

In contrast to the sequence-similarity dependent nature of homologous recombination, non-homologous or illegitimate recombination operates without the need for extensive sequence similarity between the recombining genomes (7). This form of recombination can introduce significant alterations to the viral genome, manifesting as deletions, insertions, or even more complex rearrangements. Such changes have no respect for the prior position of genes or sequences and can have profound implications on the virus's functionality and its interactions with host organisms. The detectable rate of non-homologous recombination is far lower than homologous recombination. One reason for this is that information compression in virus genomes is so intense that almost every nucleotide site has some functional significance such that the the random chimeric nucleic acid chains produced by illegitimate recombination are likely to be so defective that they are unable to replicate and transmit (8).

Recombination also occurs between the nucleic acid sequences of host cells (either DNA or RNA) and the infecting virus; a process that may occur in both directions: either from host to virus or virus to host (9). Retroviruses, by nature, after reverse transcribing their RNA into DNA integrate this DNA into the host cell's genome. Therefore it is no surprise that up to eight percent of the human genome is evidently derived from retrovirus insertions (10). There is extensive evidence of dsDNA virus genes present in cellular dsDNA genomes; almost all known genomes contain such "proviruses" (11). Recombinational transfers of DNA from host genomes to viral genomes has also been shown (12,13).

Mechanisms of Viral Recombination

Recombination in RNA Viruses

Recombination rates in RNA viruses vary significantly. RNA viruses with negative-sense single-stranded genomes can display such low recombination rates that they can effectively be considered clonal. In contrast, certain positive-sense single-stranded RNA viruses and retroviruses, such as HIV, demonstrate recombination rates that surpass mutation rates when assessed per nucleotide.

Homologous Recombination in Non-segmented RNA Viruses

In principle, RNA recombination can occur in all RNA viruses irrespective of whether their genomes are segmented or consist of a single strand (8). Copy-choice recombination is the leading hypothesis to explain the recombination process in RNA viruses (14). Copy-choice recombination is initiated during the replication of the viral genome. It is generally orchestrated by either reverse transcriptase (RT; in retroviruses like HIV (15)) or RNA-dependent RNA polymerase (RdRp; in all other RNA viruses). As RT or RdRp synthesise a complementary DNA or RNA strand from a template RNA molecule they will frequently disassociate temporarily from the template. Whereas in most cases they will re-associate with the template, in some instances they may re-associate with a different template molecule to the one that they were originally on.

This dissociation and reassociation process is known as template switching. The outcome of which is the production of a hybrid RNA molecule, containing a patchwork of genetic material derived from the different parental templates from which it was copied. This newly formed recombinant genome will contain genetic elements from both parental viruses, and could therefore potentially possess novel biological traits. The frequency and efficiency of template switching can be influenced by several factors, including the structural configuration of the viral RNA (16,17), and the close spatial association of homologous sequences (18): the latter is particularly pertinent in retroviruses like HIV where two genome copies are tightly associated within virions during the reverse transcription phase of the replication cycle.

Given the extra complexity of retrovirus replication cycles relative to those of other RNA viruses, there are potentially more opportunities for recombination to occur in retroviruses, The prototypal example of homologous recombination in retroviruses is HIV. HIV-1 has been shown to have very high rates of recombination (15,19–21)), with recombinants of divergent HIV subtypes currently being responsible for ~29% of global infections (22).

The exact mechanism of recombination within HIV-1 and other types of retroviruses is disputed. The point in the replication cycle at which recombination is most likely to occur is during reverse transcription. In HIV the dynamic copy-choice model (23) has been proposed to describe recombination during the creation of HIV-1's negative-sense strand, and the strand displacement assimilation model (24) has been proposed to explain template switching during positive-sense DNA synthesis. The existence of multiple points in the retrovirus replication cycle where template switching can occur may contribute to the high rates of recombination seen in these viruses (25).

Reassortment

Besides homologous recombination, there is another type of recombination, called reassortment (Figure 0.1), that occurs in some viruses which have genomes consisting of multiple discrete RNA molecules. The genomes of these “multi-chromosomal” viruses, the best known of which is influenza A virus, are referred to as segmented: Reassortment essentially involves the exchange of entire genome segments between different viral genomes. If reassortment occurs between antigenically distinct strains of viruses like influenza A, it can lead to something referred to as an antigenic shift where new “reassortant” genomes are capable of evading population-wide anti-viral immunity (26,27).

Reassortment occurs in viruses that have either segmented or “multipartite” genomes. Unlike segmented genome segments, which are encapsidated together for transmission within a single viral particle, each component of a multipartite genome is encapsidated within a separate virion for transmission. For multipartite viruses, following replication, a variable number of genome components are enclosed in different capsids each of which is independently transmitted. The virus subsequently needs capsids containing all of the components to be co-transmitted to the same cell to successfully initiate a new infection (28). Multipartite viruses are very rare and as such will not be discussed further as their mechanisms are also not well understood.

Reassortment in viruses with segmented genomes simply involves the 'incorrect' packaging of segments resulting in a recombinant virion with segments originating from different parental viral genomes (4). There are two main models describing this process (i) the random incorporation model (29), in which the virus does not differentiate between which segments go into each virion; and (ii) the selective incorporation model in which there exist packaging signals resulting in specific segments being incorporated into the virions (30–34). There is growing evidence that selective incorporation is the ground truth for most (if not all) segmented viruses (35,36).

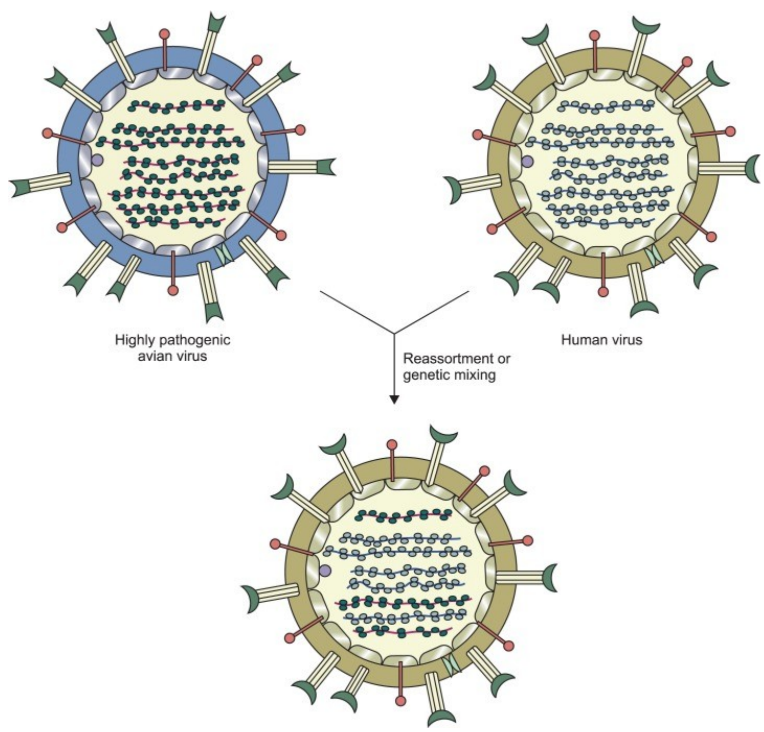


Figure 0.1: *Schematic representation of antigenic shift or reassortment. Due to the segmented nature of the viral RNA, the gene segments of one virus can reassort with those of another virus during replication in the infected host cell and form new pathogenic flu viruses. The newly formed virus can be especially dangerous if a human adapted virus, such as H1N1 or H3N2, acquires genes from a highly pathogenic strain, such as H5N1.*(37)

The common feature of recombination in both models of virion packaging (random and selective) is that after segment replication, during the packaging phase, viral segments from different lineages are combined into a single virion creating a recombinant virus. Therefore, reassortment can lead to the formation of viral offspring with genes originating from multiple parent viruses. This mixing of segments can potentially confer significant fitness advantages to the "reassortant" genomes - such as enhanced immune evasion properties if the parental viruses have evolved in the context of different immunological pressures.

The influenza A(H1N1) lineage is one of the best studied “reassortants” of this type. In this case reassortment occurred within a pig and involved the bringing-together of genome segments that originated from avian, human, and pig adapted influenza A lineages (38). The resulting reassortant was well adapted to transmission between humans and was highly resistant to neutralisation by pre-existing anti-influenza immunity within the human population.

Recombination in DNA Viruses

DNA viruses also undergo both homologous and non-homologous recombination. As with RNA viruses, homologous recombination is more common than non-homologous recombination. There are several mechanisms by which DNA viruses can undergo recombination. Copy-choice recombination (described above) is a common cause. This section will discuss the general principles of the other ways in which recombination can occur in DNA viruses.

Across the tree of life there appears to be conservation of certain DNA recombination strategies (39), α-herpesviruses are typically used as a model of eukaryotic DNA replication and recombination (40–42), and hence provide a good foundation for understanding the mechanisms at play. Herpesviruses share a recombination strategy similar to bacteriophages involving concatemers, two-component recombinases, and the hijacking of cellular proteins (43).

One such conserved strategy is DNA double-strand break (DSB) repair by homologous recombination. This process is initiated by a double strand break, a relatively common event that, if left unrepaired, would be lethal to viruses and, in eukaryotes, could promote tumorigenesis. Repair of DNA double-strand breaks can occur by microhomology-mediated end joining (MMEJ), classical nonhomologous end-joining (c-NHEJ), single strand annealing (SSA) or strand invasion (SI) mechanisms.

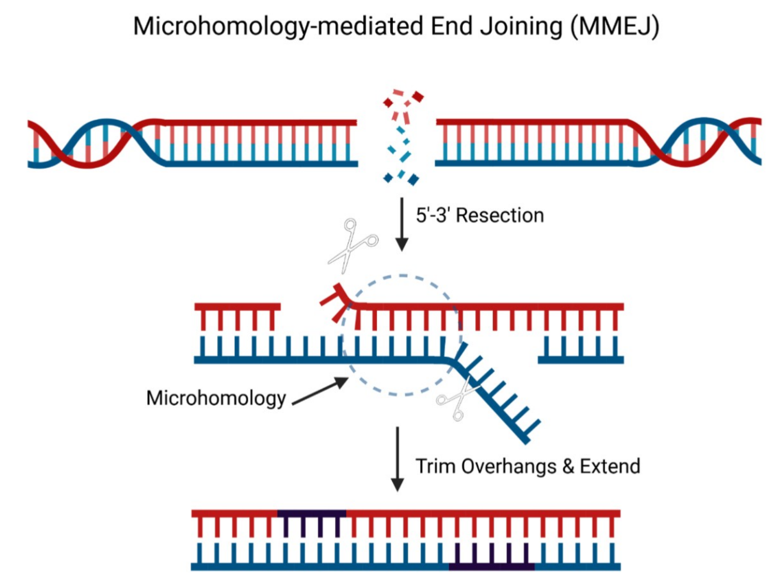


Figure 0.2 *showing a schematic of microhomology end joining (MMEJ). The initial double-strand break (DSB) occurs in the viral DNA. Limited end resection by MRN complex and CtIP exposes microhomology regions (2-20 bp). Complementary microhomology sequences anneal to each other. Non-complementary 3' flaps are removed by nucleases (e.g., XPF-ERCC1 complex). DNA polymerase θ (POLQ) extends the 3' ends, filling gaps. DNA ligase III, in complex with XRCC1, seals the remaining nicks. Repair is completed, resulting in a continuous DNA strand with a characteristic deletion between the original microhomology sites.*

Note: MMEJ operates independently of classical NHEJ and homologous pathways, providing DNA viruses with an alternative repair mechanism when canonical pathways are compromised. This process contributes to viral genome integrity but may introduce small deletions that could affect viral genetic diversity.

Microhomology-mediated End Joining (MMEJ)

MMEJ (*Figure 0.2*) is characterised by the use of micro-homologous sequences (typically 5-25 base pairs) flanking the DSB site to guide the repair process. In the context of DNA virus recombination, MMEJ can play a role in the integration of viral DNA into the host genome (44), as well as in the repair of viral DNA breaks. MMEJ involves the resection of DNA ends at the site of the DSB to expose micro-homologous sequences. These micro-homologous regions then anneal to each other, guiding the repair process. The non-homologous overhangs are then removed, and the DNA is ligated. This process often results in deletions at the repair site due to the loss of nucleotides between the micro-homologous sequences (45).

Classical Non-Homologous End-Joining (c-NHEJ)

c-NHEJ (Figure 0.3) is a mechanism that repairs double-strand breaks in DNA by directly ligating the broken DNA ends together without the need for extensive homology. This can result in the joining of non-homologous or minimally homologous DNA ends, which can contribute to viral evolution by potentially creating novel genetic recombinants. The efficiency and accuracy of c-NHEJ can vary depending on the structure of the DNA ends and the cellular context. When the ends are compatible (e.g., blunt ends or complementary overhangs), repair can be relatively accurate. However, when ends require processing, c-NHEJ can result in small insertions or deletions at the repair junction, potentially affecting viral gene function or regulation. c-NHEJ can have both beneficial and detrimental effects on viral replication. Viruses, such as Hepatitis B Virus, appear to exploit c-NHEJ as part of their formation and persistence (46). Herpes simplex virus can manipulate DNA damage response pathways, including c-NHEJ, to facilitate their replication and integration into the host genome (47).

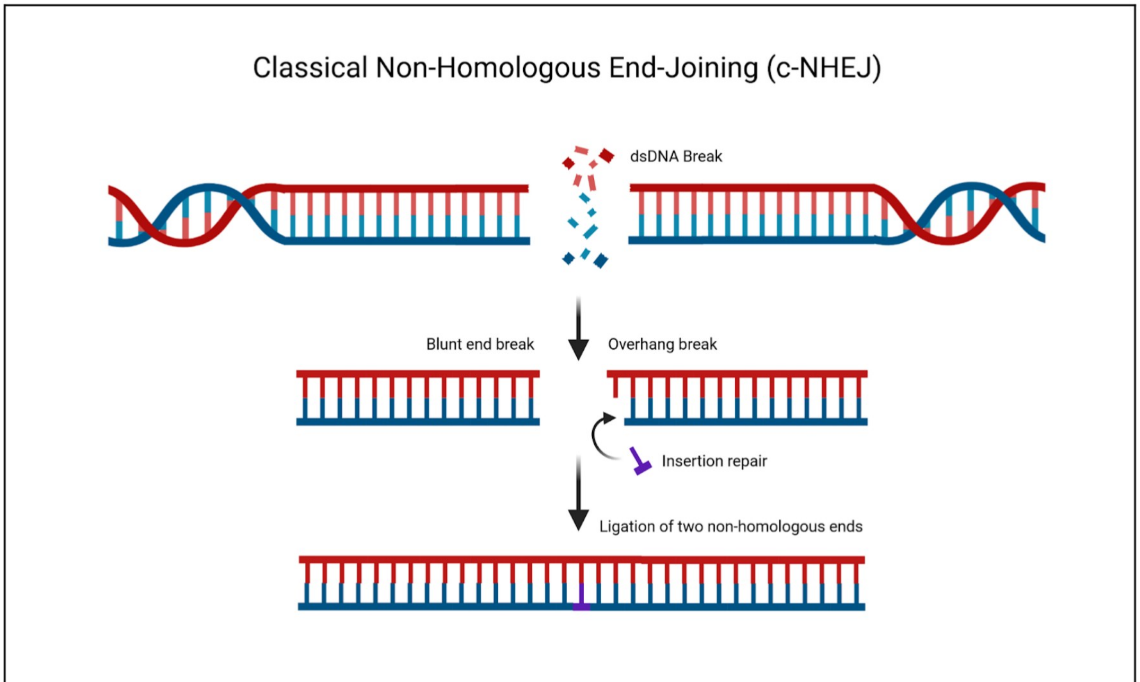


***Figure 0.3**: Classical Non-Homologous End Joining (c-NHEJ)in DNA Virus Double-Strand Break Repair*

Classical Non-Homologous End Joining (c-NHEJ) repairs double-strand breaks in viral DNA through a sequential process. First, the Ku70/Ku80 heterodimer binds to DNA ends, protecting them from degradation. DNA-PKcs then joins to form the DNA-PK complex. If needed, Artemis nuclease processes the ends to create ligatable termini. X-family polymerases (Pol μ and Pol λ) fill any gaps before the XRCC4/DNA Ligase IV complex, with XLF, seals the break. This results in a repaired DNA strand with minimal sequence changes. As the predominant DSB repair pathway in mammalian cells, c-NHEJ functions throughout the cell cycle, requires little to no sequence homology, and enables quick repair with limited nucleotide loss—making it an efficient mechanism frequently utilized by DNA viruses.

Single Strand Annealing (SSA)

Single strand annealing (figure 0.4) is a DSB repair pathway that operates when homologous sequences (longer than the micro-homologous sequences) flank the DSB site. SSA involves the resection of DNA ends at the site of the DSB to expose single-stranded DNA. These single-stranded regions can then anneal to each other at the site of homology. The non-homologous overhangs are then removed, and the DNA is ligated. In the context of DNA virus recombination, SSA can play a role in the repair of viral DNA breaks, leading to the formation of specific recombination products. Herpes Simplex Virus (HSV) has been shown to stimulate SSA in infected cells. The virus encodes proteins, such as ICP8, that have been implicated in promoting SSA and other recombination-dependent DNA synthesis mechanisms (47).

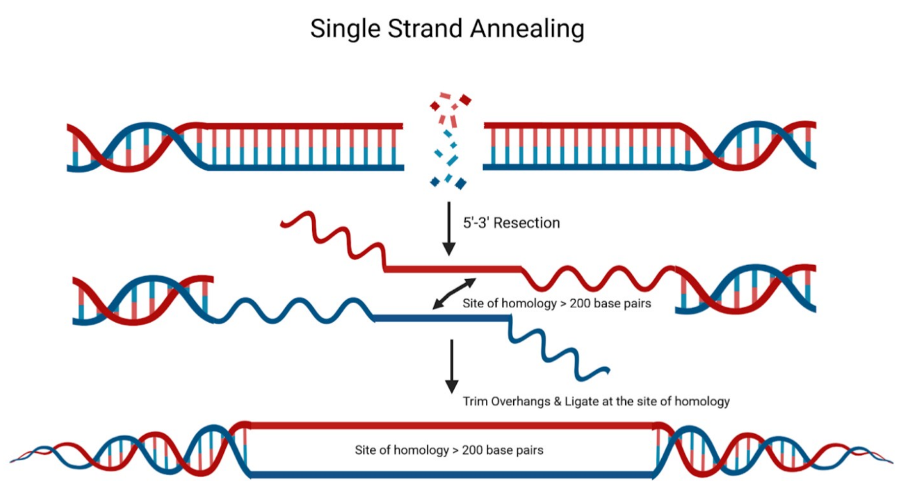


Figure 0.4: *Single-Strand Annealing (SSA) in DNA VirusDouble-Strand Break Repair*

Single-Strand Annealing (SSA) repairs double-strand breaks in viral DNA when flanking homologous sequences are present. The process begins when the MRN complex and additional nucleases perform extensive end resection, generating long 3' single-stranded overhangs. These exposed single strands allow complementary homologous sequences on opposite sides of the break to anneal, facilitated by Rad52 protein. Non-homologous 3' flap overhangs are then removed by the XPF-ERCC1 endonuclease complex. DNA polymerases fill any remaining gaps, and DNA ligase seals the nicks to complete repair. This pathway inevitably causes deletions between the repeated sequences, resulting in loss of genetic information. SSA represents a significant but mutagenic repair mechanism for DNA viruses, particularly useful when extensive homology exists and when more conservative repair pathways are unavailable.

Strand Invasion (SI)

SI is another mechanism of DNA DSB repair that results in recombination events (typically homologous). The requirements for SI-mediated recombination include a specialised ATP-dependent recombinase that can perform SI (ICP8 in herpesviruses, Rad51 in eukaryotes, RecA in bacteria, and UvsX in T4 bacteriophage) (43). After a DNA DSB occurs in the herpesvirus genome, the DNA ends undergo resection to produce single-stranded DNA (ssDNA) overhangs. In herpesviruses, the viral protein ICP8, which has single-stranded DNA-binding activity promotes strand invasion and together with helicase-primase promotes strand exchange (48). ICP8 can coat the ssDNA and facilitate the search for a homologous sequence in the viral genome. Once a homologous sequence is found, the ICP8-coated ssDNA invades this duplex DNA, forming a structure known as a D-loop. The formation of the D-loop facilitates the synthesis of new DNA using the invaded strand as a template, leading to the repair of the break and potential recombination.

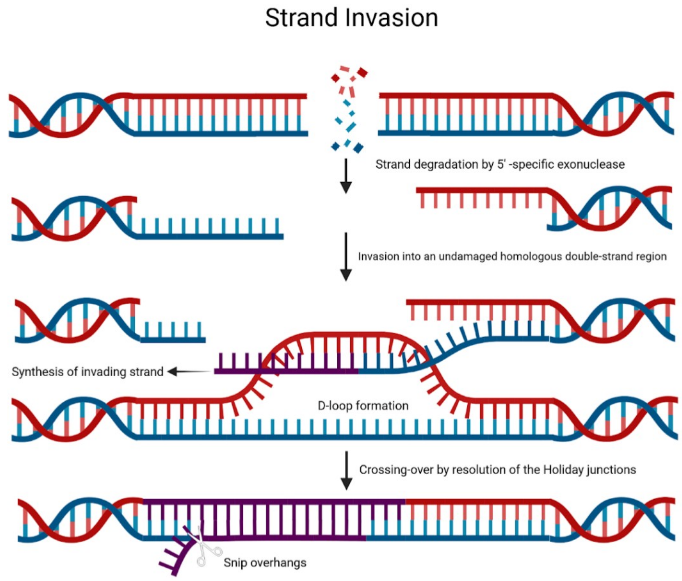


***Figure 0.5**: Homologous Recombination via Strand Invasionin DNA Virus Double-Strand Break Repair*

Homologous Recombination (HR) via strand invasion repairs double-strand breaks in viral DNA with high fidelity using a homologous template. The process begins when the MRN complex and CtIP initiate 5' to 3' end resection, generating 3' single-stranded DNA overhangs that are rapidly coated by RPA proteins. BRCA2 then facilitates the replacement of RPA with Rad51, forming nucleoprotein filaments that search for and invade a homologous sequence on the sister chromatid or homologous chromosome. This strand invasion creates a displacement loop (D-loop) structure, where the invading strand serves as a primer for DNA synthesis. As DNA polymerase extends the invading strand, second-end capture occurs, followed by branch migration and resolution of the resulting Holliday junctions by specific resolvases. This leads to precise repair with minimal risk of sequence alterations. HR via strand invasion provides DNA viruses with an accurate repair mechanism that preserves genetic integrity, though it primarily operates during S and G2 phases when sister chromatids are available as templates.

Factors Affecting Recombination Rates

There are many factors which impact viral recombination rates. Rates of recombination therefore vary greatly amongst viruses, even similar viruses. Many of these factors also interact with each other. Important factors include viral prevalence and co-infection rates, the intrinsic characteristics of the virus, the efficiency and fidelity of the viral replication machinery, and the degrees of sequence similarity shared by viruses co-infecting the same cell (which will strongly influence the relative rates of homologous vs non-homologous recombination).

Co-Infection

A primary determinant of the rate of recombination is the co-infection rate, which itself is influenced by the prevalence of the virus. For detectable viral recombination to transpire, a host cell must be coinfected with at least two genetically distinct viral particles. The likelihood of recombination is linked to the frequency of such coinfections. In environments where coinfection rates are high, there is a concomitant increase in the potential for genetic exchange between the cohabiting viruses. Therefore, the factors affecting co-infection rates are paramount to recombination rates.

It has long been understood that viruses can form aggregates/plaques upon exiting infected cells, this phenomenon is present in both enveloped and non-enveloped viruses (49–51). A long-standing assumption in virology was that these aggregates consist of genetically identical virions from the same parent virus. This understanding is the basis for viral quantification by plaque assay. It has now been shown that these aggregates can consist of multiple distinct viral genomes (52). Furthermore, it has been shown that ‘chimeric viral plaques’ consisting of multiple genetically distinct virions lead to an increase in the rate of cellular co-infection (53).

Another factor affecting the rates of cellular co-infection is bacteria mediated viral co-infection. Bacteria are capable of facilitating infection of several unrelated RNA viruses through several mechanisms (54). Several authors have demonstrated that intestinal bacteria promote co-infection of eukaryotic cells by distinct and different viruses including poliovirus, rotavirus, reovirus, and norovirus (55–57). Bacteria are able to promote viral infection via direct effects on the viruses such as binding preferentially to them (55) or by indirect effects on the host (57).

Another method of viral co-infection is via membrane vesicles containing multiple virions. This process is the non-lytic, cell-to-cell transmission of virions through membrane vesicle structures. It has been shown, *in vitro*, that several polioviruses may be packaged into phosphatidylserine-rich vesicles which can be transported to neighbouring cells facilitating co-infection (58). Other viruses, such as hepatitis C, can exit infected cells via exosomes, vesicle like structures, mediating the transferral of infectious viral particles and RNA to neighbouring cells (59,60).

Intrinsic Viral Characteristics Affecting Recombination Rates

The intrinsic nature of different viruses affects the rate of recombination, with the frequency of recombination varying widely (61). (+) ssRNA viruses tend to have higher rates of recombination than (-)ssRNA viruses (4,8), with retroviruses having recombination rates exceeding that of mutation rates (19). Furthermore, the linearity or circularity of virus genomes influence the types and rates of recombination that they are subject to; linear-to-linear recombination tends to be more common than circle-to-circle and circle-to-linear recombination (62). The length of the viral genome also impacts recombination rates, with an inverse relationship between the length of the genome and the rates of recombination (63).

Fundamental to each virus's life cycle is its replication mechanism and machinery. There are a variety of replication mechanisms employed by viruses and each affects the recombination rate and the types of recombination that the viruses employing those mechanisms may undergo, as discussed above.

The classical protein complexes responsible for most recombination events are nucleotide polymerases. The interplay between polymerase fidelity, proofreading, and recombination is complex. The fidelity of the polymerase used by a given virus for replication greatly affects the rates at which the virus will recombine, with experiments showing differing rates of recombination in the same viral population when controlling just for the fidelity of the polymerase (64).

The ability of the polymerase to proofread also affects the rates of recombination, with a lack of proofreading increasing the likelihood of template switching, and hence recombination, occurring. Proofreading is generally a feature of DNA polymerases but is rare in viral RdRps and is absent from all known viral reverse transcriptases (65).

Homologous Versus Non-Homologous Recombination

Another factor affecting the likelihood of recombination is the degree of sequence similarity between the recombining genomes. Homologous recombination, which relies on the alignment of two similar sequences, is more prevalent than non-homologous recombination (7). The 'true' rates of homologous and non-homologous recombination are difficult to determine, as the rate of recombination generally refers to the rate of 'successful' recombination which is more easily measured than how often recombination events actually occur. Newly formed recombinants are subject to natural selection and competition with their parental viruses. The only recombinants that are ever detected in nature are those that are evolutionarily fit enough to outcompete their parental viruses to a degree where they are transmitted and initiate enough new infections to eventually be discovered in virus sampling surveys. Despite this technicality, the more similar two sequences are the more likely they are to produce fully-functional recombinants such that detectable recombination events between closely related viruses should be more common than those between distantly related viruses which, in turn, should be more common than those between completely unrelated (i.e non-homologous) viruses.

The Importance of Detecting Recombination Events

The detection and characterization of recombination events and recombination breakpoints are paramount to understanding viral evolution. Mutations are the arbiter of genetic variation and hence evolution. Recombination, however, has the ability to reshape the evolutionary history of a virus by completely changing the content of the genome in a non-linear fashion, creating a mosaic virus. This mosaicism complicates the study of viral evolution as differing segments of the virus now have different evolutionary histories. Unfortunately, this simple fact seriously undermines the validity of most phylogenetics-based nucleotide sequence analysis techniques.

Specifically, by drawing inference power from accurate “phylogenetic tree models” of how different viruses are evolutionarily related to one another, these extremely common techniques implicitly assume evolution in the absence of recombination. Ignoring recombination and erroneously assuming that the evolutionary history of a given group of sequences can be accurately represented by a single phylogenetic tree can seriously obscure signals of natural selection, yield inaccurate estimates of how fast sequences are diversifying, undermine our ability to trace past dispersal dynamics, undermine the accuracy with which the phenotypes and genotypes of ancestral viruses can be inferred, and obscure signals in sequence data of past population growth and decline.

The implications of recombination extend far beyond its impact on our ability to draw inferences from nucleotide sequence datasets. Recombination has the potential to alter the evolutionary trajectories of viruses (66), facilitating the acquisition of novel traits, expanding their host ranges (67), aiding in the development of drug resistance, facilitating immune evasion, and causing difficult to predict changes in viral pathogenicity (68).

Detecting Recombination Events

Unveiling recombination events is a formidable task, often requiring sophisticated computational tools and meticulous analysis. The challenges arise from the inherent nature of recombination, which can leave only subtle and ambiguous signatures of its past occurrence in viral genomes. Furthermore, even with the best sampling techniques and next-generation sequencing advancements, the likelihood of sampling ‘pure’ parental lineage genomes of recombinants is near zero. As it is near impossible to directly compare the ‘parental’ viruses to the recombinant virus, methods must look for recombinant signals in the respective parental and recombinant lineages.

Sequence Alignment

The ubiquitous starting point for recombination event signal detection is to sample and sequence a large number of evolutionarily related virus genomes. These sampled genome sequences are packaged together in what is called a multiple sequence alignment. Sequence alignment is a fundamental technique in bioinformatics and computational biology. Sequence alignment involves arranging DNA, RNA, or protein sequences to identify regions of similarity. These regions may be conserved due to functional importance, evolutionary history, or structural constraints. Aligned sequences of nucleotide or amino acid residues are usually displayed as rows within a matrix. Gaps are introduced between the residues to ensure that identical or similar characters line up in successive columns, often accounting for insertions and deletions.

Computational approaches to sequence alignment are categorised into global and local alignments. Global alignment optimises to cover the entire length of the sequences, while local alignment focuses on identifying regions of similarity within longer, overall divergent sequences. Although local alignments are often preferable, they are more challenging to calculate. Various algorithmic approaches are used in sequence alignment, including dynamic programming, which is accurate but slow, and heuristic or probabilistic methods that are faster and suited for large-scale searches but do not guarantee the best matches.

Multiple sequence alignment is the backbone of recombination detection, it is the process of aligning three or more genetic sequences. It is more challenging than pairwise alignment and typically far more computationally intensive. Due to the computational cost most multiple sequence alignment techniques use heuristic methods rather than global optimization. The most commonly used alignment programs are Clustal Omega (69), MAFFT (70), and MUSCLE (71) all of which use heuristic methods.

Recombination Signal Detection Methods

Once a multiple sequence alignment has been created, it is possible to search it for evidence of recombination. There are many methods available for recombination signal detection. The methods described below are found in the Recombination Detection Program version 5 (RDP5) (72). These methods are outlined as they are used in RDP5 but are all also commonly used by other recombination detection tools to detect recombination. The three primary methods and four secondary methods used in RDP5 that form the basis of this project are: GENECONV, RDP, and MAXCHI (the primary methods); and, BOOTSCAN/RECSCAN, CHIMAERA, 3SEQ, and SISCAN (the secondary methods used to confirm signals of recombination detected by the primary methods).

Primary Detection Methods

GENECONV

Given an alignment, **GENECONV** (73,74) looks for aligned regions that are highly similar such that they may have arisen from recombination events. Specifically, sequences that contain unusually long identical regions or regions with high similarity based on a similarity score described below.

Procedure:

1. **Monomorphic Site Exclusion:** Select sequences and remove constant or highly selected sites, retaining only polymorphic sites in the alignment.

2. **Pairwise Comparison:** For each sequence pair, identify regions that are either:
    1. Identical and unusually long for that sequence pair.
    2. Highly similar. Similarity is scored whereby a nucleotide match awards a point and discordant sites result in a penalty. The penalty amount is adjusted by the density of polymorphic sites between the two sequences and a user-specified mismatch intensity parameter (G-scale).

3. **Significance Testing:** Assign p-values to high-scoring regions using one of two methods:
    1. Permutations: Accurate but slow.
    2. Karlin and Altschul (KA) Method: Fast but approximate, with p-values corrected for multiple comparisons. Karlin-Altschul P-values are the basis of the popular BLAST (75) method for finding sequence matches in DNA or protein databases.

The RDP Method

The **RDP** method (72) screens sequence alignments for evidence of recombination by examining every possible sequence triplet through a three-step procedure.

Procedure:

1. **Discard Phylogenetically Uninformative Sites**: Within each triplet of sequences (defined as sequences A, B and C), non-informative sites are removed. Non-informative sites are those that are identical in all three sequences or completely different in all three sequences.

2. **Calculate Percentage Identity**: A sliding window moves along the three sequences one nucleotide at a time and the average percentage identity for each possible pair of sequences (A-B, A-C, and B-C) is calculated at each position. Potential recombinant regions are identified if the identity of A-C or B-C is greater than A-B.

3. **Statistical Analysis**: The binomial distribution is used to approximate the probability of nucleotide identity runs occurring by chance. A p-value is calculated and corrected for multiple comparisons (Bonferroni correction).

MAXCHI

The maximum χ^2 method was initially proposed as a method to detect the 'crossover points' of mosaic genes in prokaryotes and to ascribe a statistical value to detect these events (76). This method was subsequently implemented in a program called **MAXCHI** (77).

The MAXCHI method examines sequence pairs within an alignment to detect recombination breakpoints by identifying significant differences in the proportions of variable and non-variable polymorphic positions in adjacent sequence regions. The method works best when only two 'parental' sequences and a recombinant sequence are compared. MAXCHI provides information on the positions of potential breakpoints but does not give information on the extent of recombinant regions.

Procedure:

1. **Discard Monomorphic Sites:** Discarding all monomorphic sites, leaving an alignment of polymorphic sites, with an option to exclude sites containing gaps.

2. **Fixed Length Window Analysis:** For each possible sequence pair, a window of set length with a central partition is moved along the sequences one nucleotide at a time.

3. **Chi Squared Calculation:** At each window position, a 2x2 χ^2 value is calculated to reflect the difference in the number of variable sites between the sequences on either side of the partition. Peaks in these χ^2 values indicate potential recombination breakpoints.

Secondary Detection Methods

CHIMAERA

This method is a modification of the MAXCHI method. The two key differences between **CHIMAERA** (77) and MAXCHI is (i) the way in which polymorphic sites are chosen and (ii) CHIMAERA can only be used to screen triplets. In principle CHIMAERA examines each triplet combination in turn to determine if one of the sequences could be a recombinant of the other two sequences. Unlike MAXCHI, CHIMAERA provides information on the positions of potential breakpoints and can give information to the extent of recombinant regions.

Procedure:

1. **Discard Monomorphic Sites:** Discard identical sites as they provide no information and sites where the selected recombinant sequence does not match either of the two parental sequences.

2. **Convert to Binary Matches or Mismatches:** The remaining sequences are then compressed into a linear string of 1's and 0's, denoting matches with one parental sequence or the other.

3. **Moving Window Analysis:** A window with a central partition is moved along this string, and at each position, a 2x2 χ^2 value is calculated to quantify the difference in proportions of 1's and 0's on either side of the partition. Peaks in these χ^2 values along the alignment indicate potential recombination breakpoints.

SISCAN

The ‘sister-scanning’ (**SiScan**) method developed by Gibbs, Armstrong, and Gibbs (78) aims to directly measure the variations in phylogenetic signals in sequence data that result from recombination, test the significance of these signal variations and attempt to distinguish misleading signals. The method compares four nucleotide sequences: two ‘parental’ sequences, a recombinant, and a dissimilar sequence (either randomly generated or taken from sequence data provided). A monte carlo randomisation is performed for all positions (columns) within a window and Z-scores are obtained, this process is repeated for each possible triplet in the alignment.

Procedure:

1. **4th Sequence Selection or Generation:** Each triplet in the alignment is scanned using this method, for each triplet a fourth sequence is either constructed by ‘horizontal’ randomisation (based on one of the triplet sequences) or drawn from the alignment; this can either be the most divergent sequence in the alignment or the sequence most closely related to the three sequences but more distantly related than the three sequences are to each other - i.e. the nearest outlier.

2. **Count nucleotide identity patterns within a window:** A window is slid across an alignment of four nucleotide sequences. At each position, the algorithm counts how many times each of 15 pre-defined nucleotide identity patterns occurs. These patterns represent the different ways the four nucleotides can be shared among the sequences, demonstrated in table 1.2.

3. **Sum counts for sister pairings:** The counts are summed for patterns where two specific sequences are identical, indicating a potential sister relationship between those two sequences. This is done for each possible sister pairing.

4. **Sum counts for informative and quasi-informative sites:** The algorithm sums the counts for informative sites (where two pairs of sequences share the same nucleotide) and quasi-informative sites (which differ from informative sites by one nucleotide).

5. **Generate randomised sequences:** The algorithm generates randomised versions of the sequences by shuffling the nucleotides within the window. It then counts the patterns in these randomised sequences, repeating this process multiple times to create a distribution of expected pattern counts under the null hypothesis of no phylogenetic signal.

6. **Calculate Z-scores:** For each pattern and sum of patterns, the algorithm calculates a Z-score, which measures how many standard deviations the observed count is from the mean count in the randomised sequences. This assesses the significance of the observed patterns.

7. **Plot Z-scores:** The Z-scores are plotted against the position of the window in the alignment. Peaks in the Z-score plot indicate regions with strong phylogenetic signals, while regions with Z-scores close to zero suggest no clear signal or potentially misleading signals due to compositional similarities.

Table 1.2 Definitions of SiScan Nucleotide Identity Patterns

Pattern	**Nucleotide identity between sequences (taxa) 1, 2, 3, and 4**
P1	1 ~ 2 ~ 3 ~ 4
P2	1 = 2 ~ 3 ~ 4
P3	1 = 3 ~ 2 ~ 4
P4	1 = 4 ~ 2 ~ 3
P5	2 = 3 ~ 1 ~ 4
P6	2 = 4 ~ 1 ~ 3
P7	3 = 4 ~ 1 ~ 2
P8	1 = 2 ~ 3 = 4
P9	1 = 3 ~ 2 = 4
P10	1 = 4 ~ 2 = 3
P11	1 = 2 = 3 ~ 4
P12	1 = 2 = 4 ~ 3
P13	1 = 3 = 4 ~ 2
P14	2 = 3 = 4~ 1
P15	1 = 2 = 3 = 4

Table 1.2: Definitions of the nucleotide identity patterns used to classify the positions in an alignment. If two or more sequences have the same nucleotide at a position they are joined by an equals sign. If the nucleotide differs then a tilde sign is used.

The sister-scanning method may not be effective when analysing less variable parts of a sequence alignment or when examining less divergent sequences. This is because the method analyses all sites within a window, and if there are too few variable sites within a window, the analysis may not be statistically robust. Increasing the window size can help address this issue, but it can also make it more difficult to detect recombination signals from small recombinant regions.

BOOTSCAN/RECSCAN

BOOTSCAN (79) is a method used to detect recombination breakpoints within a nucleotide sequence alignment. The alignment consists of a potentially recombinant sequence and a set of (non-recombinant) reference sequences. It utilises a sliding window approach, where a window of a predetermined size is moved along the alignment. At each window position, bootstrap replicates are generated, and phylogenetic trees are constructed to assess the relationships between the sequences. By examining the changes in tree topology across different window positions, BOOTSCAN can identify potential recombination events. The algorithm can also infer the parental sequences involved in the recombination (but only if non-recombinant parental sequences are accurately defined) and estimate the approximate location of the breakpoint.

Procedure:

1. **Window Sliding:** A window of fixed size is moved along the sequence alignment, shifting by a specified number of nucleotides at each step.

2. **Bootstrap Replication:** For each window position, bootstrap replicates are created by resampling the columns of the alignment within that window.

3. **Tree Construction or Distance Matrix Calculation:** Distance matrices or phylogenetic trees are constructed for each bootstrap replicate, using methods like neighbour-joining or UPGMA.

4. **Topology Analysis:** The relationships between sequences in the trees or distance matrices are analysed to identify changes in topology (if trees are used) or pairwise identity relationships (if distance matrices are used) that may indicate recombination.

5. **Bootstrap Support Calculation:** The bootstrap support for different tree topologies or pairwise sequences are calculated to assess the statistical significance of potential recombination signals.

6. **Breakpoint Estimation:** The positions of recombination breakpoints are estimated based on the transitions in tree topologies or changes in bootstrap support.

RECSCAN (80) is an automated version of BOOTSCAN initially implemented in the RDP2 software package and available in RDP5. Unlike the original BOOTSCAN, which, unlike the original BOOTSCAN method, requires no prior knowledge of potential recombinant and non-recombinant sequences:,i.e. RECSCAN can automatically search for recombination signals within a set of sequences without the prior definition of a set of known non-recombinant parental sequences. It does this by systematically examining all possible triplets of sequences and looking for patterns of alternating ancestry that suggest recombination.

There are however potential issues with techniques utilising this approach.

1. **Statistical Confidence:** There is no well-defined statistical threshold to confidently determine if a detected region is truly recombinant. While high bootstrap support for different tree topologies in different regions may suggest recombination, it does not directly translate to a specific confidence level. Binomial or chi-squared tests can help identify significant regions, but there is no clear way to correct for multiple testing, which could lead to false positives, especially with large datasets.

2. **Fixed Window Size:** BOOTSCAN requires a fixed window size, which can be problematic when nucleotide substitution rates vary along the sequence length. In regions with low variability, small recombinant regions might be missed, while in highly variable regions, the fixed window size might not be optimal for detecting larger recombinant regions.

3SEQ

Boni et al. (81) describe **3SEQ** as an exact, nonparametric, method to detect recombination events in nucleotide sequences. 3SEQ focuses on polymorphic sites in the alignments and scans triplets like other methods described. Each sequence in a triplet is in turn queried to determine if it could potentially be the recombinant of the other two sequences in the triplet.

Procedure:

1. **Identify informative sites:** The method focuses on informative sites, which are nucleotide positions where the child sequence matches one parent but not the other. Thus all monomorphic sites are discarded, and all sites where neither of the two ‘parental’ sequences matches the selected ‘recombinant’ sequence are also discarded.

2. **Random walk representation:** The sequence of informative sites is represented as a random walk, where a match with one parent is an "up" step (+1) and a match with the other parent is a "down" step (-1).

3. **Calculate test statistics:** The test statistic, $\Delta_{m,n,b}$, is calculated. This statistic measures the difference between the observed pattern of matches and mismatches in the child sequence compared to the expected pattern under the null hypothesis of no recombination. The subscripts m and n represent the number of informative sites matching each parent, and b represents the number of breakpoints (locations where the child sequence switches from matching one parent to the other).

4. **Determine statistical significance:** The probability of observing a test statistic value ($\Delta_{m,n,b}$) as extreme as or more extreme than the observed value is calculated under the null hypothesis of no recombination. If this probability (the P-value) is below a chosen significance threshold (e.g., 0.05), then the null hypothesis is rejected, and recombination is inferred.

The authors highlight that their method is exact, nonparametric, and computationally efficient, making it suitable for analysing large datasets. They also emphasise that the method can distinguish between recombination and other evolutionary processes like variation in mutation rates, which is a potential confounding factor in recombination detection. As with the CHIMAERA method, 3SEQ relies on matches between parental and recombinant sequences and may have trouble identifying recombination when only one parental sequence is present in an alignment.

Recombination Identification Techniques & Statistics

Once a potential recombination event has been detected amongst three sequences (a triplet), the actual recombinant amongst the triplet must be identified. RDP5 makes use of several statistical methods to determine which of the three sequences is the recombinant. These methods analyse different aspects of the sequences' relationships and evolutionary patterns to make this determination. RDP5 then utilises a decision tree based on these statistics to determine which of the sequences is the recombinant. The statistics utilised by RDP and in this thesis are:

1. The **PHYLPRO** method (82), or the phylogenetic profile (**PhPr**), calculates pair-wise Jukes-Cantor distances between a sample sequence and all other sequences in the

dataset, using segments of the alignment bounded by the approximate recombination breakpoints. The method then determines Pearson's correlation coefficient between these distance lists. The sequence yielding the lowest correlation coefficient is then identified as the recombinant. However, this method can be less reliable when numerous sequences in the sample descend from the same recombinant ancestor.

2. **TreePhPr** is a variation of PHYLPRO which utilises branch lengths from neighbour-joining trees rather than genetic distances.

3. Two additional PHYLPRO variants, **SubPhPr** and **TreeSubPhPr**, calculate the sum of squares of differences in distances between triplet sequences and the remainder of sequences in the two alignments. These methods operate on the principle that the difference in distances between the recombinant and other sequences should exceed that of the parental sequences. SubPhPr uses the genetic distances whilst TreeSubPhPr uses tree branch length.

4. The **SubDist** and **TreeSubDist** methods measure average phylogenetic correlation between alignments when each sequence in the triplet is sequentially removed. These methods assume that removing the recombinant sequence will result in the greatest increase in phylogenetic correlation between alignments. SubDist employs distance matrices whilst TreeSubDist uses tree branch length distances.

5. **ParsimonyO** and **ParsimonyI** are modifications of subtree pruning and re-grafting (SPR) methods. These approaches (as implemented in RDP5) construct neighbour-joining trees from alignment portions bounded by recombination breakpoints and determine the minimum number of SPR operations needed to convert one tree into another. This is unlike the original methods in which trees are constructed using different genes (83) These methods specifically consider only the subtree containing potential co-recombinant sequences and assume this set is monophyletic.

6. The **O:E** and **O:EDist** methods identify recombinants by comparing observed recombination signals with theoretical expectations. These methods exploit the principle that recombination events should be detectable across multiple sequence combinations when sampling is sufficient. The process identifies recombination signals with a minimum 30% sequence overlap between breakpoints. It then

constructs theoretical sets of triplets that would show recombination signals if each sequence were the recombinant. The sequence whose expected pattern best matches the observed signals is identified as the recombinant.

7. The **DistRank statistic** is determined by comparing two genetic distance rank statistics, RankF1 and RankF0. To calculate these, the nucleotide distance between the target sequence and all others in an analysed alignment is summed to produce a “sumdist” score - this is done separately for the region between the breakpoints and for the region in the remainder of the alignment. For each sequence in a triplet used to detect a recombination signal, the RankF0 score is calculated as the rank position of its “rest of the alignment” sumdist score, and the RankF1 score is calculated as the rank position its “between the breakpoints part of the alignment” sumdist score.

8. The **dMax (VisRD)** method (84) analyses quartets of sequences to identify recombinants by examining changes in evolutionary relationships across different regions of the alignment. The method constructs numerous four-taxon maximum parsimony trees containing each sequence from the triplet in turn. For each quartet, it determines map locations using two portions of the alignment: the fragment between the recombination breakpoints and the remainder of the alignment. The difference between these map locations, denoted as d, is recorded for large numbers of quartets containing each triplet sequence. The underlying principle is that recombinant sequences will show greater phylogenetic incongruence between different regions of the alignment compared to parental sequences. Therefore, the triplet sequence that yields the greatest difference in map locations across all examined quartets (dMax) is identified as the recombinant.

9. The **Conflict** statistic examines the degree to which distances are smaller between potential co-recombinant sequences than with other sequences in the alignment. This method assumes that true recombinant sequences descended from the same ancestor should be more similar to each other than to any other sequence in the alignment.

10. **OuCheck** assesses how phylogenetic relationships between triplet sequences and other sequences are disrupted by recombination. It analyses rooted neighbour-joining tree topologies constructed from regions between recombination breakpoints and the remaining alignment. For each triplet sequence, the method counts how many phylogenetic relationships with other sequences in the alignment

remain consistent between both trees. The recombinant is identified as the sequence maintaining the fewest unchanged relationships. The contribution of OUCheck to the identification of recombinants using the RDP5 decision tree is contingent on the value of another “sub-statistic” referred to as the “SimScore” which sets the OUCheck statistic for all sequences in a triplet to zero if the number of sequences carrying evidence of the same recombination signal as any one of the sequences in the triplet exceeds 50% of the number of sequences in the alignment.

11. **TrpScore** measures changes in rooted neighbour-joining tree positions for each triplet sequence between trees constructed from different alignment regions. Using branch averaging to account for sampling biases, the recombinant sequence is expected to show the highest number of topology changes.

12. **SetDistT** and **SetDistP** compare polymorphic site distributions between recombination breakpoints in the triplet sequences with those in the remainder of the alignment. It is expected that if the polymorphic sites are evenly distributed between the two regions, the recombinant sequence will be the one that is alternatively most closely related to the major and minor parents. These methods can suggest the presence of unsampled parental sequences based on polymorphic site density patterns.

These various statistical approaches are combined using a weighted consensus to identify the most likely recombinant sequence within a triplet. While these methods perform well when recombination is relatively rare, their accuracy can decrease in sequences with frequent recombination events, particularly when multiple sequences in the analysed triplet are recombinant.

Comparison of Recombination Detection Techniques

The effectiveness of recombination detection software is crucial for accurate identification of recombination events in viral genomes. However, several factors can lead to false positives or erroneous identification of recombination. These include analysing highly divergent or substantially similar sequences, asymmetric tree topology, linkage disequilibrium between nucleotide substitutions, and the "patchy-tachy" phenomenon where different sections of sequences exhibit unequal evolution rates (85,86).

Errors can also originate from data collection and preparation processes. Inconsistencies in genomic alignments, especially in reference-based genome assemblies, can lead to false positives. Chimeric sequences, formed during PCR amplification or sequencing data analysis, can also contribute to misreported events. These artefactual chimeric sequences are often present in databases, making it difficult to estimate the true roles of recombination during the evolution of many viral taxa where homologous recombination is rare (such as in (-)ssRNA viruses) .

Given the various challenges in accurately detecting and identifying recombination, it would seem apparent that detection methods should be quantitatively evaluated to determine the limits of different software applications. This involves assessing their false-positive rates and overall performance under different scenarios. Despite this need, there is a paucity of literature comparing software packages; with recent comparisons focusing on only a few methods (85) or providing an overview of the current methods but not quantitatively comparing them (86). Studies that do review and compare multiple recombination detection programs are outdated (77) or missing newer developments (87).

A reason for the paucity of research in this area is the difficulty in testing these methods. Despite next-generation sequencing resulting in a massive uptick in open source sequence alignments testing these methods has not become any easier. Simply put, empirical biological data is not a useful tool to compare methods as there is no readily available ground truth, meaning that comparisons must be made on the most likely recombinant but in edge cases or examples where two methods disagree we cannot conclusively say which method is correct.

Therefore, it stands to be that time-forward sequence simulators may be able to provide a fair comparison dataset. Genome evolution simulators like SANTA-SIM (88), SimBac(89), Bacmeta (90), and CoreSimul (91) provide methods to create reference alignments with varying levels of evolutionary details modelled. Time-forward sequence simulators can model several key evolutionary events with varying degrees of complexity. These include point mutations (both transitions and transversions), insertions and deletions of varying lengths, and recombination events between co-infecting viral genomes.

More sophisticated simulators (such as SANTA-SIM (88)) can incorporate selective pressures, allowing for purifying selection that maintains functional elements, positive selection that drives adaptation, and frequency-dependent selection that models host-pathogen dynamics. Population-level events can also be simulated, including bottlenecks, exponential growth phases, and stable population sizes, all of which influence the fixation rates of mutations. Some simulators account for more complex phenomena such as codon usage bias, RNA secondary structure constraints, and protein structural requirements.

However even advanced simulators may struggle to fully capture the intricate interplay of multiple selective pressures operating simultaneously in real viral populations, such as the need to maintain both protein function and RNA structure whilst evading host immune responses. One recent study (92) found that they could easily distinguish between an empirically sampled sequence alignment and a simulated one utilising machine learning techniques implying that despite the ever more complex modelling, nature is still not quite fully replicated.

Machine Learning

Machine learning and Artificial intelligence are currently central in public discourse with the advent of large language models such as OpenAI's ChatGPT and Google's Gemini, these models seem to promise that artificial general intelligence may soon be a reality in the not too distant future. These advanced models evolved from simple statistical models to their current forms. Early approaches focused on rule-based systems and basic statistical learning, where computers followed explicit programmed instructions. This progressed to more advanced statistical methods like logistic regression (93) and gradient boosters such as LightGBM (94) which could learn patterns from data.

The field then saw the development of neural networks (95) , initially simple structures inspired by biological brains. As computing power increased, these networks grew more complex, leading to deep learning architectures capable of handling tasks like image recognition and natural language processing. Recent developments have produced large language models and multimodal AI systems that can process text, images, and other data types simultaneously. Despite these advances, many current AI systems still rely on fundamental principles established in earlier statistical and machine learning approaches.

These tools are now widely available and easily implemented through open source software libraries such as scikit-learn (96) which provide implementations of hundreds of machine learning techniques, and tools wrapped in a simple to use python API. Furthermore, neural networks have become increasingly simple to implement thanks to libraries such as TensorFlow (97) and PyTorch (98). Training these models has also become simpler with increasingly powerful consumer grade graphics cards, provided by companies such as Nvidia, capable of rapid matrix multiplication and other mathematical functions necessary in modern day deep learning.

Summary

The extensive literature on viral recombination demonstrates that this process represents a crucial mechanism of genetic exchange occurring across all viral types, albeit with varying frequencies and through distinct molecular pathways. The process fundamentally requires cellular co-infection by multiple viral particles and can drive substantial genomic alterations affecting viral evolution, host range expansion, and modifications to pathogenicity (38,66–68).

The computational detection of recombination events remains challenging. While numerous detection methods exist, each exhibits specific limitations and potential biases(85,86,99). Contemporary approaches predominantly rely on sequence alignment and statistical analysis of sequence patterns, with tools such as RDP5 employing multiple detection methods to enhance accuracy (72). However, the field lacks comprehensive comparative analyses of these methods' effectiveness, partly due to the difficulty in establishing ground truth in empirical datasets (87).

Time-forward simulators present a potential solution for method evaluation by generating datasets with known recombination histories (88–91). However, even sophisticated simulators struggle to fully replicate the complexity of real viral evolution, particularly the intricate interplay between multiple selective pressures (92). This limitation, combined with the inherent challenges in recombination detection, suggests an opportunity for novel computational approaches, particularly machine learning methods that could potentially integrate multiple signals of recombination to improve detection accuracy.

The literature thus indicates a clear requirement for enhanced recombination detection methods, whilst highlighting the substantial challenges involved in developing and validating such approaches. This understanding provides the foundation for the machine learning-based approach to recombination detection developed in this thesis, which aims to address these methodological gaps through the integration of multiple detection signals and the application of contemporary machine learning techniques.

Aims & Objectives

1. **Simulate viral recombination data across a range of parameters:**
   a. Create a custom version of SANTA-SIM that implements recombination in a variety of forms, and keeps track of all recombination events.
   b. Generate viral sequences with different recombination rates, mutation rates and genome sizes using this custom version of SANTA-SIM.
   c. Create datasets under different evolutionary scenarios including constant population size, exponential growth, and varying selection pressures.
   d. Generate a challenging test dataset based on SARS-CoV-2 for model validation and comparison.
   e. Make these datasets publicly available with code to enable usage (https://github.com/JoshCullinan/RDPML)

2. **Create a dataset suitable for training machine learning models based on the simulated data:**
   a. Convert recombination events from simulated alignments into a structured dataset containing statistics from RDP5's detection methods.
   b. Label sequences with their true recombination status and position within triplets.
   c. Preprocess the data through scaling and feature selection.

3. **Evaluate RDP5's Baseline Performance:**
   a. Use RDP5's consensus statistic as a baseline for comparison.
   b. Calculate precision, recall, F1-scores and ROC AUC metrics.
   c. Generate confusion matrices to understand error patterns.

4. **Develop and compare multiple machine learning approaches:**
   a. Train binary classification models including logistic regression, gradient boosting and random forests.
   b. Implement neural network architectures for both binary and position-based classification.
   c. Evaluate models using precision, recall, F1-score and ROC AUC metrics.
   d. Compare performance against RDP5's baseline.

5. **Begin integration with RDP5:**
   a. Establish a framework for incorporating models into RDP5.
   b. Implement the logistic regression model into RDP5.

Methods

Development Environment

Viral genome evolution simulations were performed using facilities provided by the University of Cape Town's ICTS High Performance Computing team: hpc.uct.ac.za.

Machine learning models were trained on a personal computer using Ubuntu 24.04.1 LTS. The computer had an AMD Ryzen 5 3600 CPU, 32GB of DDR4 RAM, and an Nvidia 3070 8GB graphics card. Python 3.12.3 was used, with python libraries for machine learning: TensorFlow 2.18.0 and SciKit-Learn 1.5.2. The full requirement list can be found in the supplementary material and at https://github.com/JoshCullinan/RDPML along with the codebase.

Simulating Viral Recombination Data

Santa-Sim

SANTA-SIM (88) is a Java-based software application designed to simulate the evolutionary dynamics of a population of gene sequences over time. It employs a discrete component-based approach to model the underlying biological processes, encompassing replication (inclusive of recombination), mutation (inclusive of insertions and deletions), fitness assessment, and selection.

SANTA-SIM was chosen as the simulation package for this project as it is a powerful time-forward simulator with good fundamentals and it has an open source repository (https://github.com/santa-dev/santa-sim) which allowed for the development of additional features required for this project, specifically: (1) outputting a recombination event history and (2) computing a global alignment. These features were implemented into a custom forked version of SANTA-SIM and were made in conjunction with Philip Swanepoel (100). (https://github.com/phillipswanepoel/santa-sim/tree/Recomb_and_align)

This custom version was adapted to produce a "perfect" alignment using information from simulated indel and recombination events. A "perfect" alignment is one where all homologous nucleotides share a column, and where all gap characters represent actual insertion or deletion events within the sequences. Furthermore, this version stored information about each genome's recombination event history, as well as all of the recombination events that occurred during the simulation. The recombination events file contained a list of all the recombination events that occurred during the simulation which could be found in any of the sampled sequences. This file noted the event number, break points, generation at which the recombination occurred and the identity of the parent genomes. The second file generated was a sequence events map, this file consisted of the identity each sequence sampled as well as a list of each recombination event that the progenitors of that sequence underwent in time order of the events. By comparing the recombination events file and the sequence events map file it is possible to accurately map areas of the genome that contained recombinant sequences, including regions of overlap where old recombination events were overwritten by new events.

In summary, the two main additions to the santa-sim program was the ability to track recombinants and to create ‘perfect’ alignments.The recombination process works by (1) using a dual infection model where co-infection can occur (with a configurable rate) resulting in two strains being present in a cell, (2) allowing recombination to occur with a configurable probability during replication, (3) selecting breakpoints in the genome where genetic material switches between parents, (4) creating recombinant genomes by combining segments from both parent genomes, and (5) tracking recombination events for later analysis and alignment. Recombinant hotspots can also be configured. The system tracks complete recombination histories, which enables the alignment feature to correctly represent genetic relationships between sequences. A more granular explanation of the recombination history code can be found in the appendix..

These two additions allowed for the generation of several large datasets of time-forward evolution of viral populations (originating from a single progenitor genome), that contained a history of which genomes were recombinants, where the breakpoints had occurred, which genomes had been parents, and it was possible to describe the state of the genome at the time of recombination.

Simulation of Viral Evolution for Machine Learning Datasets

Santa-Sim uses XML formatted configuration files. Six XML files were written, five for the purpose of creating a training set and one for the purpose of validation and testing. Each XML file specified a different virus (and in some cases a region of a virus). The files specified, amongst other variables, the sequence length, population size (and whether it was static or dynamic), which fitness function would be applied, whether Indels would be used, and if there were recombination hotspots present (*table 1.3*).

Table 1.3 shows the configurations of the six XML files used to create the machine learning database.

	XML-1	**XML-2**	**XML-3**	**XML-4**	**XML-5**	**XML-Test**
Original Virus (Region if applicable)	SARS-CoV-2 (Spike Protein)	HIV-1	SARS-CoV-2 (ORF1ab)	Bombali Ebolavirus (Whole Genome)	Zika Virus (Whole Genome)	SARS-CoV-2 (Whole Genome)
Sequence Length	3822	9181	5800	19025	10727	29902
Population Type and Size	Dynamic N=100, R=1.1, K=10000	Static N=4000	Static N=6000	Dynamic N=1000, R=1.2, K=50000	Static N=5000	Static N=3000
Fitness Function	Purifying Fitness	Exposure Dependant	Frequency Dependant	Age Dependent + Purifying	Purifying	Purifying - affecting sites with literature supporting selection
Recombination Hot and Cold Spots	Hotspot: 1000-1200 Coldspot: 50-150	Hotspot: 4000-4200 Hotspot: 9000-9100 Coldspot: 6200-6300	None	None	None	None
Indels	None	None	None	None	Rate=0.035, r=1 p=0.4	Rate=0.025, r=3 p=0.3

N = Population Size/Inoculum Size

Dynamic growth was modelled as logistic growth with R = Growth Rate, and K = Carrying Population.

The **Indels function** used: Rate = Probability of an indel occurring in an epoch, with the length of these indels defined by the negative binomial distribution. This distribution was defined by two parameters r and p, where r = number of successes till length is specified, p = probability of success.

Furthermore, SANTA-SIM allows for variables to be specified at runtime - these included the recombination probability, mutation rate, generation count (at which to sample from the population), and sample size (*table 1.4).* In order to generate more diversity in the sampled genomes three options for each variable were created, and for each XML file all 81 combinations were used.

Table 1.4 showing the shared runtime parameters

Generation Count & Sampling Frequency	**Sample Size**	**Recombination Rate**	**Mutation Rate**
2500	50	0.005	0.00004
3250	100	0.01	0.00008

<table>
<tr><td>4000</td><td>200</td><td>0.02</td><td>0.000012</td></tr>
<tr><td colspan="4">Generation Count & Sampling Frequency determined the number of generations (or epochs) would be run for each configuration. At the predetermined number of generations a sample of the population would be taken which became the alignment for that configuration.

Sample Size indicates the number of genomes in each alignment.

Recombination Rate determined the probability of a recombination event happening per epoch - this was further affected by a static variable called dual infection probability which was set at 0.1 and determined the likelihood of two viruses co-infecting the same host.

Mutation Rate is the probability for a mutation to occur (per site and per generation). Furthermore, santa-sim allows for a rate determining relative preference for transitions versus transversions, and to further customise a rate bias can be set which determines the probabilities for transition/transversions for each nucleotide combination.</td></tr>
</table>

Each XML file defined the basics for each simulation, to add diversity the 81 runtime parameters were added. Each XML file with each of the 81 runtime parameters was run multiple times with random states ensuring different outcomes each time. The number of times each file was run was constrained by the memory available on the available HPC cluster - the ‘curie’ cluster was used which has 64 CPU cores and 128GB of RAM . Running the scripts in parallel used significant amounts of RAM and as the size of the genomes, and population sizes, increased so too did the RAM usage which limited the number of runs possible.

In total XML-1 was run through all 81 combinations a total of 15 times each, yielding 1215 alignments of various sample sizes (50, 100, 200). XML-2 was run in a similar manner with each combination being used 15 times, this however yielded 1010 alignments as some alignments exceeded the RAM quota and were aborted early. XML-3 yielded 718 files with the aim to run each combination ten times - with 92 aborted runs. XML-4 yielded 1080 alignments. XML-5 yielded 1296 files with each combination being run 16 times. XML-Test yielded 97 files. At the end of the simulating period 5319 alignments of various sample sizes were created for use in the machine learning training dataset, and 97 alignments for use in the test dataset.

The goal of such a large dataset was to produce an evolutionarily feasible dataset that captured as many possible combinations of events that could occur in the wild and be sampled in reality. The final test set was based on the entire SARS-CoV-2 genome, with the spike and ORF1ab regions manually selected as regions to undergo purifying selection.

Converting Simulated Viral Alignments into a Dataset for Machine Learning

The generated alignments consisted of strings of base pairs (nucleotides) of varying lengths. The goal of this step was to convert the alignments into a dataset of integers and real values that could be used in traditional machine learning techniques. This required the identification of the recombinant, minor and major parents. With the triplets identified they could be run through RDP5 to generate statistics from the various methods already described - these methods provide numerical values indicating the likelihoods of which of the triplet was the recombinant. With these values in a dataset, machine learning models could be trained to interpret all of the signals together to improve detection of the recombinants.

In order to do this a custom version of RDP5 was kindly created by Prof. Darren Martin. This version used the generated alignment file, a ‘recombination history’ file, and a ‘sequence events map’ file. The recombination history file contains information about each recombination event, which was the event number, event breakpoints, and age of the event (generation at which it occurred in the simulation). The sequence event map file was a representation of each sequence in the alignment and the events that were contained in each sequence - the event number was linked to the recombination history file. As in nature, recombination events can overlap, overwrite, or occur within other events, therefore, the sequence events map file allowed for the identification of these sub-sequence events.

This custom version of RDP5 made several changes which facilitated the machine learning dataset creation:

1. Command line options were added to automate many alignments being fed through the software and allow for output files that are typically not available client side.
2. The ‘raw’ uninterpreted statistics were made client-facing and could be saved in a .csv file with the prepended title RecombIdentifyStats.
3. The ‘interpreted’ statistics were made client facing, and saved as a prepended RecIDTests.csv file.
4. A special run mode was created which allowed for RDP5 to generate statistics for a triplet that it had not itself determined were minor parent, major parent, or recombinant. In this method the information from the recombination history file was used. This method of interpretation was found to be deleterious to the machine learning models as the statistics generated were too unlike what was within the normal output - therefore this method was replaced by change 5.

5. The method used to create the final dataset was to run the alignment files 'naively' through RDP5, forcing the program to compare each recombinant event to every other possible pair and create statistics as it normally would do so. These events were then reduced to only include 'real' events that were present in the recombination history file. Events were included provided that they:
   5.1. Included a true recombinant from the recombination history file, whether correctly labelled as a recombinant or incorrectly labelled as a parent.
   5.2. A 95% confidence interval of the breakpoint chosen by RDP5 included the breakpoints of the actual recombination event. Meaning that RDP5 had not just chosen the correct triplet but also the correct event correlating to the triplet.

With these changes made, a pipeline was created which took in each alignment fasta file, recombination history file, and sequence events map file with the aim to generate a large dataset. The pipeline took in the relevant files and with some intermediate formatting steps, generated a .csv file containing a single row for each recombination event, with the following structure.

Table 1.5 showing a representation of the data used during model training.

Triplet Pos 1	Triplet Pos 2	Triplet Pos 3	Statistic A 1	Statistic A 2	Statistic A 3	…
1	0	0	0.28009	0.08755	-0.08755	…
0	1	0	0.16013	0.47458	-0.51736	…
…	…	…	…	…	…	…

Table 1.5 shows a representation of the data that was generated and used during training, the first three columns are a one-hot-encoded representation of which sequence within the triplet is the recombinant. In the first row, the first column contains the recombinant and hence has a value of 1, whilst column two and three contain 0's as they are parental sequences. The next three columns (Statistic A 1, A 2, A 3) contain the first generated statistic for the first three triplet columns, i.e., Statistic A1 is Statistic A for triplet 1, whilst Statistic A2 is Statistic A for triplet 2 and so on. Therefore, this file contains 120 statistic columns, 40 for each of the respective triplets. Two files were created, a training file consisting of all of the data from simulations of XML-1 through 5, and then the test set consisting of data from the Unseen-XML simulation. In total these files contained 491 124 sequences for classification, which is 163 708 triplets.

The decision made in step 5 of the dataset creation method resulted in a significant difference in the number of recombinants present in the training and test set when compared to the method used in step 4 - this was due to the decision to only accept recombinants were RDP5 had selected the correct breakpoints and selected the recombinants as either one of the parents or the recombinant. The largest difference between recombination events present in the alignment and those used to train on was 772 events. The mean difference across all data was 80.17 events, with a median of 47.00.

As described above there were 6 XMLs of various settings used to simulate the dataset, *table 1.6* shows the summary statistics for the difference in accepted events and the number of simulated events. The alignments with few differences indicate that the simulated events were similar to events that RDP5 would have correctly detected, but not necessarily correctly identified. Whilst a higher difference indicates that the alignment provided significant challenges to the detection of recombinants, and their breakpoints.

Table 1.6 shows the difference between accepted events and number of simulated events per XML.

	Number of alignments	**Mean Difference**	**Median Difference**	**Standard Deviation**	**Minimum Difference**	**Maximum Difference**
XML-1	1215	59.4	38.0	52.1	2	209
XML-2	1109	97.5	83.0	62.0	8	319
XML-3	718	205.9	143.5	173.1	15	772
XML-4	1080	34.3	29.0	23.2	0	125
XML-5	1271	45.6	29.0	39.9	0	175
XML-Test	97	175.4	186.0	73.25	22	337

Data Exploration and Preparation

The data was then examined, manipulated, and cleaned using python and python libraries including NumPy, Pandas and SciKitLearn. For the training of binary models, where the goal was predict whether a sequence was a recombinant or parents based on just that sequences statistics, the dataset was formatted in such a way that each row contained a response variable (1 for recombinant, 0 for parents) and the related statistics. This binary training dataset consisted of 464 505 sequences for identification. A validation set was created in the same way which consisted of 10 002 samples, and finally the unseen test set consisted of 16 617 sequences for classification.

The dataset included three ‘Consensus Statistics’ that were RDP5’s own decision tree based algorithm to determine which of the three sequences in the triplet is the likely recombinant. These statistics were used to assess RDPs performance on the test set. They were not used in model training, and were subsequently removed from the training, validation, and testing datasets. This left 111 statistics (37 per sequence) from the original 120.

The next step was to remove statistics, from this point called features, that had no variance. This form of preprocessing removes features that will not provide value to the model and increase noise in the data. The features removed all had a variance of 0, and hence would not contribute to training - this left 96 features. These features were then scaled. Standardisation of a dataset is a common requirement for many machine learning estimators, without standardisation they might not perform as expected, especially if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). Scaling features helps with faster model convergence and optimisation by preventing gradient blow outs and keeping features with smaller variances relevant compared to features with larger variances. The standard scaling function was used, this function takes the sample value x and calculates a scaled value z:

$$z = (x - u) / s$$

Where *u* is the mean of the training set for that feature, and *s* is the standard deviation of the training samples. This scaled value z was used for model training. The validation dataset and test sets were treated in the same way with the removal of the consensus feature and standardisation (with the mean and standard deviation of the training set used for all datasets).

Binary Model Training

Logistic Regression

The first model fitted to the data was a logistic regression (93) model from the SciKit-Learn library (96). Logistic regression is a statistical method used for modelling the probability of a binary outcome, typically represented as 0 or 1. It employs a logistic function (or sigmoid curve) to transform a linear combination of predictors into a probability value. This probability can then be used to classify new observations into one of the two outcome categories. Logistic regression can also be used to model multiple outcomes using a one versus rest approach, where multiple binary logistic regression models are made for each of the outcomes.

In this case the model approached the problem as binary, i.e parent or recombinant, and did not see the data for the features of the other two in the triplet. Thus the model was trained to look at 32 features which were the data for one of the sequences in a triplet and determine if this was likely a parent or a recombinant from this data.

The model was trained using the class_weight parameter - this aids in model training with unbalanced datasets. Particularly important considering that there are 2 parents for every 1 recombinant leading to a skewed dataset. The unbalanced nature of the dataset leads to the algorithm favouring the detection of parents over recombinants as it will predict 66% correctly if it always picks parents. The class_weights parameter applies a penalty to the correct prediction of parent and a bonus to the prediction of recombinant. With the penalty/bonus ratio being inversely proportional to the number of parents and recombinants - in this case: 0.75 for parents and 1.5 for recombinants.

Furthermore to prevent overfitting an elasticnet regularisation was applied to the algorithm. An Elasticnet penalty incorporates both an L1 (Lasso) and L2 (Ridge) penalty. The L1 component promotes sparsity by potentially setting some coefficients to exactly zero whilst the L2 component helps handle correlated features and prevents coefficients from becoming too large.

The combination often performs better than either L1 or L2 alone, especially when dealing with correlated predictors, high-dimensional data with some irrelevant features, cases where the goal is both feature selection and coefficient shrinkage. A ratio of 0.5 was used, meaning that half the regularisation term was L1 and half was L2. The cost function was minimised using scikit-learn's implementation of the SAGA (Stochastic Average Gradient Ascent) solver (101), a memory efficient solver that does not need to calculate and store full gradients, frequently converging faster than traditional stochastic gradient descent.

LightGBM

Light gradient boosting machine (LightGBM) (94) is a high-performance implementation of a gradient boosting ensemble algorithm, a meta-algorithmic approach that leverages the strengths of multiple sub-algorithms to achieve superior predictive performance through variance reduction. Specifically, LightGBM constructs an ensemble of decision trees.

Decision trees are rudimentary yet powerful prediction models that emulate human decision-making and readily accommodate both regression and classification tasks.

Decision trees are built via recursive partitioning, a non-parametric methodology that avoids distributional assumptions and employs a tree-like structure to represent the intricate relationships between input features (continuous, discrete, or categorical) and potential outcomes. LightGBM is a particularly attractive method due to the performance improvements it has over alternative gradient booster methods.

A LightGBM model was trained to predict binary outcomes in the same manner as the logistic regression model. The default hyperparameters were used except for those described in *table 1.7*. A learning rate of 0.01 was used as it allowed for relatively quick convergence without overfitting to the training data. Class weights of 0.75 for parents and 1.5 for recombinants were used to counter the imbalanced nature of the problem as discussed in the logistic regression section.

One technique to prevent overfitting was the addition of L2 regularisation to the cost function at a value of 0.3, this penalty term prevents the model from overreliance on correlated features and aids with preventing gradient blow outs. Early stopping was used as another form of overfitting prevention and to avoid unnecessary training times, the early stopping feature tracked the loss of the validation fraction, which consisted of the 10 002 validation samples.

Table 1.7 showing the hyperparameters used in the LightGBM model

Learning rate	0.01
Maximum features per split	0.95
L2 (Ridge Regularisation)	0.30
Class Weight	0.75 for parents & 1.5 for recombinants
Early Stopping	True - validation loss tracked.

Random Forest

Random forests (102) are an ensemble learning method that constructs multiple decision trees during training and outputs the majority vote (for classification) or average prediction (for regression) of the individual trees. The key principle behind random forests is to leverage the power of multiple, diverse decision trees to improve predictive accuracy and reduce overfitting. Within the model construction process, randomness is introduced through two distinct avenues:

1. Bootstrap aggregation (Bagging): Each tree is trained on a random sample (with replacement) of the training data, introducing diversity in the training sets.
2. Random feature subsets: At each node of each tree, a random subset of features is considered for splitting, further decorrelating the trees and reducing variance.

This combination of bagging and random feature selection results in a robust and accurate model that is particularly effective for high-dimensional data and complex relationships.

Random forests are adept at classification problems, whether binary or multi-class. A binary task was chosen in this case. The SciKit-Learn implementation of random forests was used. The model was trained on the same training dataset, as the other models. The default settings were used except for those listed in *table 1.8*. The default criteria for node splitting is gini index. The same class weights used for the logistic regression model (0.75 for parents and 1.5 for recombinants) were used to compensate for the imbalance in parents and recombinants in the dataset.

Table 1.8 showing the hyperparameters used in the random forest model

Max Depth	16
Number of estimators	280
Minimum sample split	5
Minimum samples per leaf	2
Max leaf nodes	None
Class Weights	0.75 for parents & 1.5 for recombinants
Minimum impurity decrease	0
Maximum features	Square Root of the number available.

Max Depth (16): *This parameter determines how deep each tree in the forest can grow. A depth of 16 was chosen as it allows for sufficient model complexity to capture important patterns whilst preventing the extreme overfitting that can occur with deeper trees.*

Number of Estimators (280): *This specifies the number of trees in the forest. The value 280 was chosen through empirical testing, as it provides enough trees for stable predictions through averaging whilst maintaining reasonable computational efficiency. Generally, more trees reduce variance but have diminishing returns beyond a certain point.*

Minimum Sample Split (5): *This sets the minimum number of samples required to split an internal node. A value of 5 ensures that splits only occur when there is sufficient data to make meaningful divisions, helping prevent overfitting on noise in the training data whilst still allowing the model to learn important patterns.*

Minimum Samples per Leaf (2): *This establishes the minimum number of samples required at a leaf node. Setting this to 2 prevents the creation of leaves with single samples, which helps reduce overfitting whilst still allowing the model to capture fine-grained patterns in the data.*

Max Leaf Nodes (None): *When set to None, trees are allowed to grow until they reach their maximum depth or until no further splits are possible. This allows the model to create as many leaf nodes as needed within the constraints of other parameters.*

Class Weights (0.75 for parents & 1.5 for recombinants): *These weights address the class imbalance in the dataset. The higher weight for recombinants (1.5) compensates for their lower frequency in the dataset, whilst the lower weight for parents (0.75) prevents their overrepresentation from dominating the model's decisions.*

Minimum Impurity Decrease (0): *This parameter controls the minimum reduction in impurity required to make a split. Setting it to 0 allows splits to occur regardless of the impurity decrease, relying on other parameters to control model complexity.*

Maximum Features (Square Root of available features): *This determines how many features to consider when looking for the best split. Using the square root of the total number of features is a common heuristic that helps ensure diversity among trees whilst maintaining computational efficiency.*

Other classification models were utilised including Gaussian naive Bayes, and nearest neighbours. These models have been shown to improve performance; the details of these models are not included here because they ultimately failed to enhance performance. A summary of their performance and parameters is included in the appendix.

Neural Network Training

Neural networks (95), a subset of machine learning algorithms, are computational models that draw inspiration from the structural and functional principles of the human brain. At their core, they consist of interconnected processing units called "neurons" or "nodes" organised in layers. The flow of information through the network is facilitated by weighted connections between these neurons.

Each neuron receives multiple inputs from the preceding layer, applies a weighted sum to these inputs, and then passes the result through a non-linear activation function. This activation function introduces non-linearity into the network, enabling it to model complex relationships within the data. The output of one neuron then serves as an input to neurons in the subsequent layer, thus propagating information through the network.
The power of neural networks lies in their ability to learn from data. This learning process involves adjusting the weights of the connections between neurons to minimise the discrepancy between the network's output and the desired output. The predominant algorithm employed for this weight adjustment is called "backpropagation." In backpropagation, the error (or loss) between the network's output and the true labels is computed. This error is then propagated backward through the network, and the weights are

updated in a manner that reduces the error. This iterative process of forward propagation (computing outputs) and backpropagation (adjusting weights) continues until the network converges to a state where it can accurately predict the desired output.

The ability of neural networks to employ non-linearity allows them to model complex relationships and predict a wide variety of output types. In the context of this project the neural networks were tasked to predict binary outcomes and multi-class outcomes (major parent, minor parent, and recombinant).

Binary Classifier Neural Network

The binary classifier neural network took in the same 32 features from the 464 505 training sequences as described elsewhere. A variety of model structures were tested, but the most performative model consisted of four blocks after an input layer of 32 nodes (*figure 1.1*). The first block consisted of a dense layer of 64 nodes with a leaky rectified linear (LReLU) activation function. These were passed to a batch normalisation layer and then passed through a dropout layer (20%). Batch normalisation standardises layer outputs to zero mean and unit variance, stabilising training and enabling faster learning rates. Dropout randomly disables neurons during training, preventing overfitting and creating an ensemble effect by forcing the network to learn redundant representations. Both techniques improve neural network training stability and generalisation.

Following this a residual layer was used, residual layers create shortcuts in neural networks by adding a layer's input back to its output, allowing the network to learn residual functions rather than complete transformations. This helps combat the vanishing gradient problem in deep networks and enables better training of very deep architectures by providing direct paths for gradient flow. The next block in the model consisted of a dense layer of 32 nodes with a LReLU activation function. This was passed through batch normalisation, and then another dropout layer. After this layer the residual layer from the previous block was then added back to the gradients.

The next two blocks consisted of a typical bottle neck structure of 16 nodes and then 8 nodes with each block containing a layer normalisation and dropout layer (20%). The final layer was a single dense node with a sigmoid activation function to predict parent or recombinant (0 or 1).

The model was trained for 500 epochs or training was stopped early by a callback function if the validation score was not improving (with a patience of 10 epochs, and minimum loss change of 0.0001). The model was trained with a batch size of 512 samples.

The binary focal cross-entropy loss function (103) was utilised. This loss function measures the difference between predicted probabilities and actual binary labels (0 or 1). It penalises incorrect predictions. The loss is the average difference between predicted and actual values across the dataset. Binary *focal* cross-entropy is a loss function that modifies the standard binary cross-entropy loss function by adding a modulating factor (*gamma*) that reduces the contribution of easily classified examples and places more emphasis on hard, misclassified examples during model training. Furthermore, a weight balancing factor (*alpha*) was used which downweights the loss of the majority class (parents) to aid with classification of the less prevalent class (recombinants). For our training, a *gamma* of 3.0 and an *alpha* of 0.65 were used.

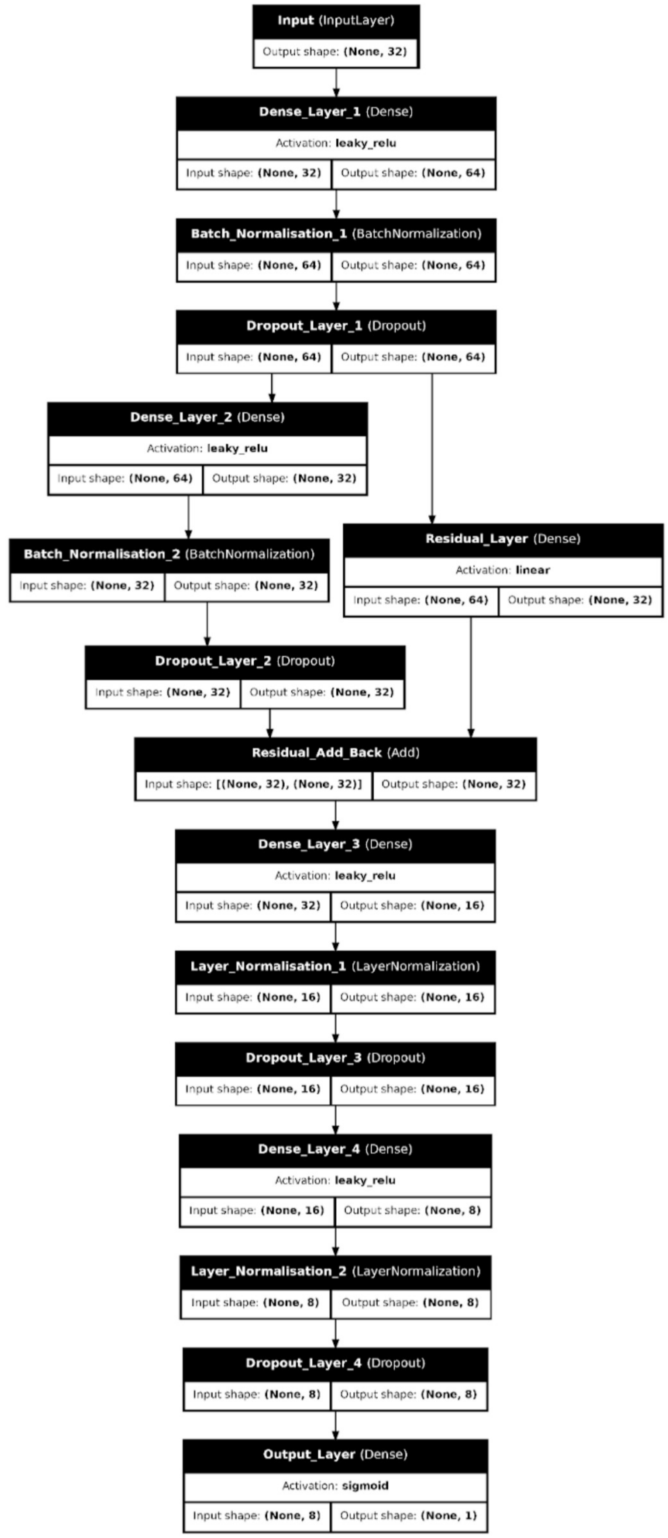


Figure 1.1 *shows the structure of the binary classifierneural network. A 'block', as referred to in the text, consists of a dense layer, a normalisation layer (batch or layer) and a dropout layer. The boxes show the activation function that was used with each layer. The final activation function is a sigmoid function as it takes the input from the 8 final nodes and outputs one prediction between 0 and 1.*

Optimisation of this loss was carried out by a variety of optimisers but two methods were the most performative: stochastic gradient descent with momentum (SGD) and the AdamW, a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments with an added method to decay weights (104), with AdamW proving to be the most performative. The initial learning rate for AdamW was 0.0001 with a callback which reduced the learning rate by a factor of 0.5 if the validation loss score began to plateau.

The trained model was deployed in two distinct prediction modes: binary classification and dependent classification. In binary classification mode, the model processed each potential recombination event independently, outputting a single probability score to indicate whether the sequence was a recombinant (>0.5) or parent (<0.5). Alternatively, in dependent analysis mode, the model evaluated each of the three sequences within the triplet independently, generating individual probability scores for each; the virus with the highest recombinant probability score was then designated as the recombinant, while the others were classified as parents.

Position Selection Neural Network Method

All models trained before this model utilised a binary problem of parent or recombinant, and did not take into consideration the features of the other two samples in the triplet. This model was created to model the relationship between all 96 features in the dataset per triplet and predict the recombinant from this. This changed the number of samples from 464 505 with 32 features to 154 834 with 96 features, i.e. the number of triplets in the dataset. In training the network it became clear that the distribution of the position of the recombinant within the three options was heavily skewed to position 1 (65%), whilst position 2 and 3 had only 17% and 18% respectively, this lead to unbalanced training where the model began to predict predominantly position 1 to guarantee 65% recall and precision.

To rectify this, the training set, validation set, and test set were adjusted to reposition the recombinants randomly such that the distribution was 33% in position 1, 2, and 3. This was done in such a way to retain the correct statistics for each sequence, but balance the position of the recombinant so that the model had to learn genuine features rather than just predict a predominant position.

These features were then scaled using the same standardisation method described above. In order to improve computational performance the dataset was converted from a one-hot-encoded representation to a sparse representation. A sparse representation of data is a way of representing data where most of the elements or features are zero. This is in contrast to a dense representation, where most elements have non-zero values.

The loss function used for this model was a sparse implementation of categorical cross entropy. Categorical cross-entropy is a loss function used in multi-class classification problems where the model outputs a probability distribution over all possible classes. It measures the dissimilarity between the predicted probabilities and the true labels, typically represented as one-hot encoded vectors, but in this case takes in a sparse representation. By penalising incorrect predictions, especially those with high confidence, the loss function directs the model during training to learn parameters that produce probability distributions closely aligned with the true class labels.

Minimisation of this loss function was carried out by the AdamW optimiser mentioned earlier. The initial learning rate was 0.0001, with the same callback for learning rate reduction on validation loss plateau as employed in the binary neural network. The network was trained with a batch size of 256. The network was trained to 500 epochs or until an early stopping callback was activated which terminated training, occurring when the validation dataset loss was no longer improving - the model with the best weights was restored at early stopping.

The structure of the network can be seen in *figure 1.2*. This network has a similar layout to the binary network in that it has an Input layer, followed by a bottleneck structure with a residual layer, the major difference is the size of the input layer (96 features), and the output layer consists of a softmax activation function to predict a position rather than a binary outcome.

The network starts with an input layer of 96 nodes. This is passed to a dense layer of 96 nodes with an LReLu activation function. This is passed to a batch normalisation layer and then a dropout layer (20%). This first block is passed to a residual layer with a linear activation function which is saved to be added to the network later. The next block consists of a dense layer with 64 nodes with a LReLu activation function, a layer normalisation layer and then a dropout (20%) layer, at this point the residual layer is added back to the network. From this point forward the next two blocks consist of dense layers with LReLu activation functions, normalisation and dropout layers. The two dense layers have 30 and 18 nodes respectively. The final layer is a dense layer of 3 nodes. It outputs a probability distribution for which one of the triplet was the recombinant. This is done by the softmax activation function which, for example, produces an output of [0.1, 0.3, 0.6], where the highest value is taken to be the recombinant.

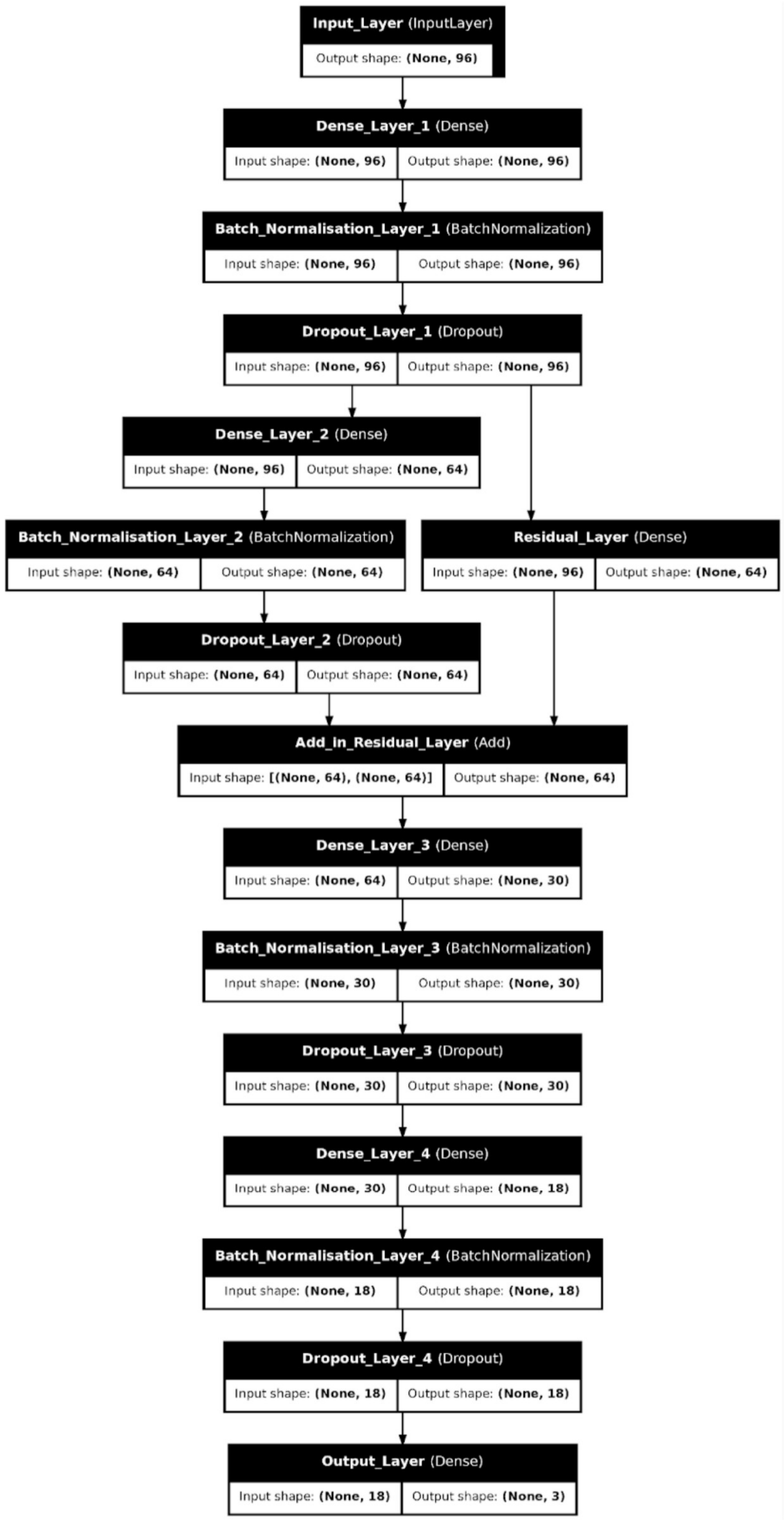


***Figure 1.2** shows the structure of the position selectionneural network. A 'block', as referred to in the text, consists of a dense layer, a normalisation layer (batch or layer) and a dropout layer. The boxes show the activation function that was used with each layer. The final activation function is a softmax function - it takes in the input from the final layer and outputs a probability distribution.*

Model Performance

The models were tested on a hold out test data set that was simulated specifically to provide a more challenging dataset of faint recombinant signals from a whole genome based on SARS-COV-2. The simulation settings for this dataset have already been described.

Metrics to assess performance were selected due to the problem being an unbalanced class classification task. Due to the class imbalance typical metrics such as accuracy do not provide the entire pictures, for example in this case there are always two parents to a recombinant - if the model was to predict only parents it would achieve 66% accuracy which is not a true reflection of the models performance. Due to this other key metrics were used to assess performance: precision, recall, and f1-score. Accuracy was calculated for each model but was not a key metric on account of the unbalanced data. These results were tabulated for simple cross examination.

Precision and recall are two useful metrics employed to assess the efficacy of classification models, particularly in scenarios with imbalanced datasets, such as this thesis. Precision quantifies the proportion of correctly predicted positive instances out of all instances predicted as positive. A high precision score indicates a low rate of false positives, which does not necessarily mean good overall detection of positives in the data. It is calculated as:

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)}$$

Recall (also known as sensitivity), measures the proportion of correctly predicted positive instances out of all actual positive instances. A high recall score indicates a low rate of false negatives, implying the model has good detection but potentially at the cost many false positives. It is calculated as:

$$Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)}$$

There is often a trade-off between precision and recall. If you increase precision, recall might decrease, and vice versa. This trade-off is typically managed using the f1 Score, which is the harmonic mean of precision and recall, balancing both metrics. The f1 score penalises the model when either precision or recall is low. Even in an unbalanced dataset:

- If precision is high but recall is low (meaning the model is highly selective and predicts fewer positives), the F1 score will be low.

- If recall is high but precision is low (meaning the model predicts many positives but with many false positives), the F1 score will also be low.

It is calculated as:

$$F1\,Score \;=\; 2\times\frac{Precision + Recall}{Precision \times Recall}$$

Confusion matrices were used to visualise the model's predictions against the true labels, showing the counts of true positives, true negatives, false positives, and false negatives. Confusion matrices help identify where the model might be systemically making mistakes, like tending to misclassify parents as recombinants or vice-versa. These matrices were made with the number of samples and normalised to aid with comparison.

Lastly, the receiver operating characteristic (ROC) curves were plotted. These curves show the true positive rate versus false positive rate. The area under the curve (AUC) provides an agnostic value that is good for comparison between different models.

Code Availability

The code used in this project is available at https://github.com/JoshCullinan/RDPML, the majority of the code is written in python in jupyter notebooks for easy interpretation by the reader. This notebook contains most of the figures in this project in the correct context and may aid the reader's ability to understand the training and the evaluation of the models. The datasets are available on request.

Results

To achieve optimal identification of recombinant sequences from the statistics routinely generated by RDP5's various recombination detection methods, I investigated several machine learning approaches, starting with simpler models and progressively moving to more complex architectures. The goal was to find an approach that could effectively integrate the multiple signals of recombination detected by RDP5's various methods into a more reliable overall prediction of which sequences in analysed sequence triplets were recombinant.

I began with logistic regression as a baseline model due to its interpretability and ability to handle binary classification problems efficiently. While relatively simple, logistic regression can reveal important linear relationships in data and establish a performance benchmark against which more complex models could be compared. The gradient boosting (LightGBM) and random forest models were then explored as they excel at capturing non-linear relationships and have proven particularly effective for problems involving multiple interacting features - a characteristic that matches well with the multiple recombinant identification statistics generated by RDP5.

Neural networks were ultimately investigated due to their capacity to learn complex patterns and relationships between features. Two distinct neural network approaches were tested: a binary classification approach similar to the other models, and a novel triplet-based architecture that considered all three sequences simultaneously. This triplet approach was specifically designed to capture the inherent relationships between the three sequences involved in any potential recombination event.

The evaluation metrics were carefully chosen to address the inherent class imbalance in the data, where there are always two parent sequences for every recombinant sequence. While overall accuracy provides a general measure of performance, precision and recall were particularly important metrics as they offer more nuanced insights into model performance. Precision measures the proportion of correctly identified recombinants among all sequences predicted to be recombinants, while recall measures the proportion of actual recombinants that were correctly identified. The F1 score, being the harmonic mean of precision and recall, provides a balanced measure of model performance that accounts for both false positives and false negatives.

In addition to precision, recall, and F1 scores, the Receiver Operating Characteristic Area Under Curve (ROC AUC) was used as a key performance metric for evaluating the models. The ROC AUC is particularly valuable for this application as it measures a model's ability to discriminate between recombinants and non-recombinants across all possible classification thresholds, making it threshold-independent. A perfect model would achieve an AUC of 1.0, while random guessing would result in an AUC of 0.5. This metric is especially relevant for recombinant identification as it accounts for both sensitivity (recall) and specificity, providing a single score that captures how well the model balances the trade-off between true positive rate and false positive rate. The ROC AUC is also advantageous for evaluating models on imbalanced datasets, where there are inherently fewer recombinant sequences than parental sequences, as it is insensitive to class imbalance.

The models were evaluated on multiple datasets: a validation set drawn from the same distribution as the training data, and a more challenging test set designed to assess generalisation to more difficult cases. Additionally, I created a mixed dataset combining both routine and challenging cases to better reflect real-world applications. This comprehensive evaluation approach was designed to ensure that any improvements in recombinant identification would translate to practical benefits in real-world virus sequence analysis.

In the following sections, I present detailed results for each model, beginning with RDPs decision tree as a baseline for comparison, moving on to the simpler models and progressing to the more sophisticated neural network architectures. For each model, I provide performance metrics and analyse their strengths and weaknesses in identifying recombinant sequences.

RDP5 Baseline Performance

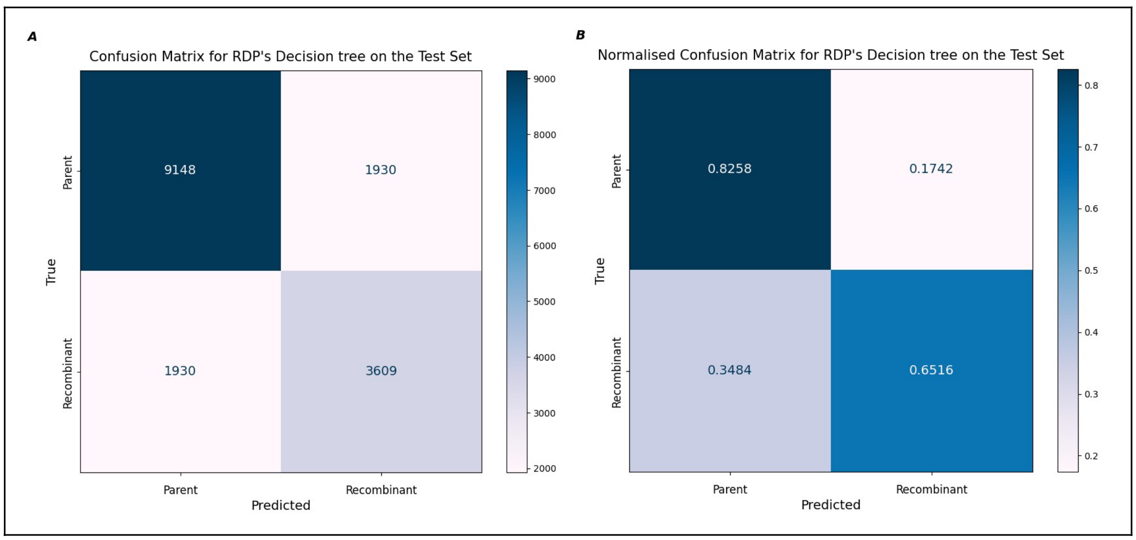
RDP5 has an inbuilt decision tree that produces a value called ‘consensus’. This value is based on the same statistics used in this project. The consensus statistic is used to help identify which of the three sequences in a triplet is the recombinant. With access to the consensus metric it is possible to make predictions on the same unseen test set, consisting of 16 617 samples, that the rest of the models were tested on. RDP does not provide probabilities of which of the triplet is the most likely, thus it is only possible to pick the highest score out of the three as the recombinant. This type of prediction forms the basis for many of the results in this chapter and will henceforth be called dependent decision making.

Using dependent decision making RDP achieved an overall accuracy of 76.77%. *Table 2.1* shows precision, recall and f1-scores with the number of examples per case. RDP's decision tree achieved precision and recall values of 82.58% for parents, and precision and recall values of 65.16% for the recombinants.

Table 2.1 shows RDPs performance metrics on the unseen test set.				
	Precision	Recall	f1-score	Support (n)
Parent	0.8258	0.8258	0.8258	11 078
Recombinant	0.6516	0.6516	0.6516	5539

Figure 2.1 (A&B) shows confusion matrices for the decision tree, these aid in assessment for false positive rate and true positive rate by showing the number of correctly predicted recombinants and parents versus the false predictions in a two by two matrix. *Figure 2.1 (A&B)* shows that RDP correctly predicted 9148 out of 11 078 parents (82.58%), and 3609 out of 5539 recombinants (65.16%). Due to the dependent prediction method that is being utilised, if a recombinant is misclassified as a parent it will follow that one of those parents will be correctly classified whilst the other sequence in the triplet will be misclassified.

Figure 2.1 (C) shows a receiver operating characteristics (ROC) curve, this curve is agnostic to class choice and plots the true positive rate versus the false positive rate. The area under the curve (AUC) is a useful metric for cross model comparison, in this case RDP achieved an ROC AUC of 0.7387, significantly better than random guessing (AUC of 0.5).



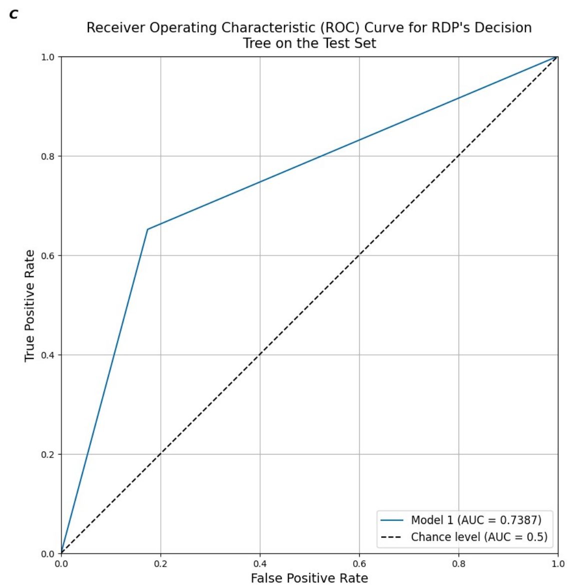


Figure 2.1: Showing the performance metrics for RDP's decision tree in the form of confusion matrices (A & B) and ROC curves (C).
***A**: Shows the confusion matrix for RDP's decision treemaking dependent decisions on the test set.*
***B**: Shows the normalised confusion matrix for RDP'sdecision tree making dependent decisions on the test set.*
***C**: Shows the receiver operating characteristic curvefor RDP's decision tree on the test set whilst making dependent decisions, achieving an ROC AUC of 0.7387.*

Logistic Regression

Logistic regression was selected as the first model to test due to its interpretability and its ability to effectively capture linear relationships between variables. The model's performance was evaluated on both validation and unseen test datasets to assess its predictive capabilities and generalisability.

Logistic Regression: Binary Decisions on the Test Set

The initial models were trained in a binary format, i.e., to predict in isolation whether a sequence was a parent or recombinant based on their associated statistics. Using this prediction method the logistic regression model achieved an accuracy of 75.39% when evaluated on the unseen test set comprising 16 617 sequences (11 078 parents and 5 539 recombinants). *Table 2.2* shows similar metrics to RDPs decision tree, with mediocre recall, precision, and f1-scores for recombinants, 0.6080, 0.7366, and 0.6662 respectively. The model performs better for the parents with a precision of 0.8527, recall of 0.7626, and f1-score of 0.8051, although this is likely due to the unbalanced dataset.

Figure 2.1 (A&B) shows the logistic regressions performance on the test set, the model correctly predicted 4080 recombinants out of 5539 (73.66%), and correctly predicted 8448 parents out of 11 078 (76.26%). The misclassifications were also similarly balanced with the model incorrectly labeling 2630 parents as recombinants (23.74%) , and 1459 recombinants as parents (26.34%).

Table 2.2 showing performance metrics for the logistic regression model making binary decisions on the unseen test set.				
	Precision	Recall	f1-score	Support
Parent	0.8527	0.7626	0.8051	11 078
Recombinant	0.6080	0.7366	0.6662	5 539

Figure 2.1 (E) shows the ROC curve for the model, the ROC AUC of 0.7496 improves upon RDPs decision tree AUC of 0.7387, indicating slightly improved discrimination between parents and recombinants. The binary decision making version of the logistic regression performs well but with a major caveat being a high rate of false positives resulting in poor precision for the recombinant class.

Logistic Regression: Dependent Decisions on the Test Set

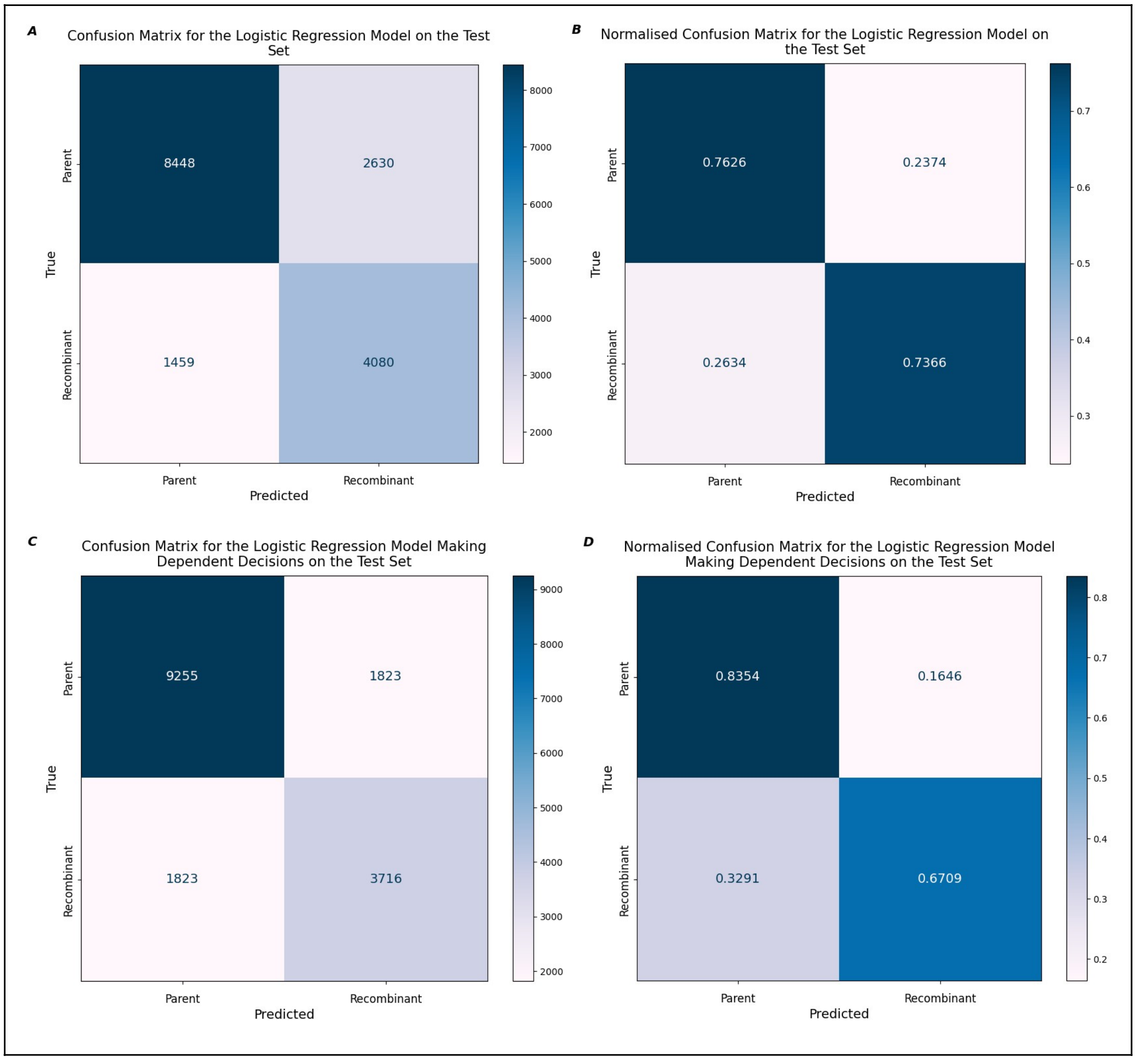
The logistic regression model was also tested using the dependent decision making technique, similarly to RDP. In this method the binary probabilities for each of the sequences in the triplet are predicted, and the sequence with the highest probability is made to be the recombinant - this is the most real world application for these models and will likely be the most useful point of comparison.

Table 2.3 showing performance metrics for the logistic regression model making dependent decisions on the unseen test set.				
	Precision	Recall	f1-score	Support
Parent	0.8354	0.8354	0.8354	11 078
Recombinant	0.6709	0.6709	0.6709	5 539

Using this method the logistic regression model achieved an overall accuracy of 78.06%. The model correctly predicted 3716 recombinants from 5539 sequences (67.09%), a worse performance than the binary prediction method, and 9255 parents from 11 078 sequences (83.54%) (*figure 2.1 C&D*). There is a significant drop off in the number of correctly predicted recombinants from 73.66% down to 67.09% (*figure 2.1 B & D*) - this is due to the dependent decision technique picking the highest scoring sequence as the recombinant.

This prevents both high ranking sequences from being selected as recombinants where previously at least one of the picks would be correct.

The impact of this approach is reflected in the model's performance metrics shown in *table 2.3*. For the recombinant class, the model yielded identical precision, recall and f1-scores of 0.6709, whilst the parent class showed consistent scores of 0.8354 across all measures. The model's overall discriminatory power, as measured by the ROC AUC, reached 0.7532 a small improvement upon the independent decision making process (0.7496) (*figure 2.1 E*). The curve helps show that the independent decision process can achieve a higher true positive rate but at the cost of a higher false positive rate.



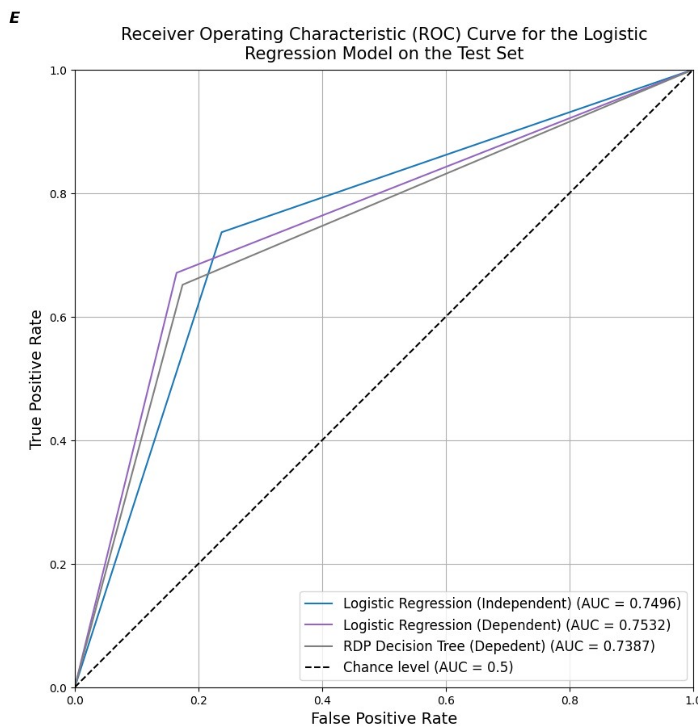


***Figure 2.2** shows the performance of the logistic regressionmodel in an independent mode and a dependent mode as plotted by confusion matrices (A-D) and ROC curves for comparison against RDP.*
***A:** Shows the confusion matrix for the logistic regressionmodel on the test dataset making binary decisions.*
***B:** Shows the normalised confusion matrix for the logisticregression model on the test dataset making binary decisions.*
***C:** Shows the confusion matrix for the logistic regressionmodel on the test dataset whilst making dependent predictions.*
***D:** Shows the normalised confusion matrix for the logisticregression model on the unseen test dataset whilst making dependent predictions.*
***E:** Shows the receiver operating characteristic (ROC) curves for the logistic regression model on the test dataset. The blue line indicates the independent decision making (AUC of 0.7496) and the purple line shows the dependent decision making (AUC of 0.7532), a random chance line plotted for reference with an AUC 0.5.*

Gradient Booster Classifier (LightGBM-Style)

After establishing that the logistic regression model slightly outperformed RDPs decision tree, a gradient booster based on LightGBM was selected as the next model due to its ability to capture complex non-linear relationships in high-dimensional data through an efficient decision tree ensemble approach. Despite being more complex than logistic regression, LightGBM type models maintain good interpretability while offering superior performance compared to other gradient boosting methods and strong protection against overfitting.

Gradient Booster: Binary Decisions on the Test Set

When evaluated on its independent binary decisions, the gradient booster classifier demonstrated improvements over both RDP's decision tree and the logistic regression model. The gradient booster achieved an overall accuracy of 76.67%. The confusion matrices (*figure 2.3 A&B*) reveal that the model correctly classified 8 627 out of 11 078 parents (77.88%) and 4 114 out of 5 539 recombinants (74.27%).

Table 2.4 shows the performance metrics for the Gradient booster model whilst making binary decisions on the test set.				
	Precision	Recall	f1-score	Support
Parent	0.8582	0.7788	0.8166	11 078
Recombinant	0.6267	0.7427	0.6798	5 539

This performance translated to precision scores of 0.8582 for parents and 0.6267 for recombinants (*table 2.4*). The recall scores were more balanced at 0.7793 and 0.7417 for parents and recombinants respectively, the disparity in precision scores indicates a considerable rate of false positive recombinant classifications. The model's ROC curve analysis (*figure 2.3 E*) yielded an ROC AUC of 0.7607, slightly higher than both RDP's decision tree (0.7387) and the logistic regression model (0.7496).

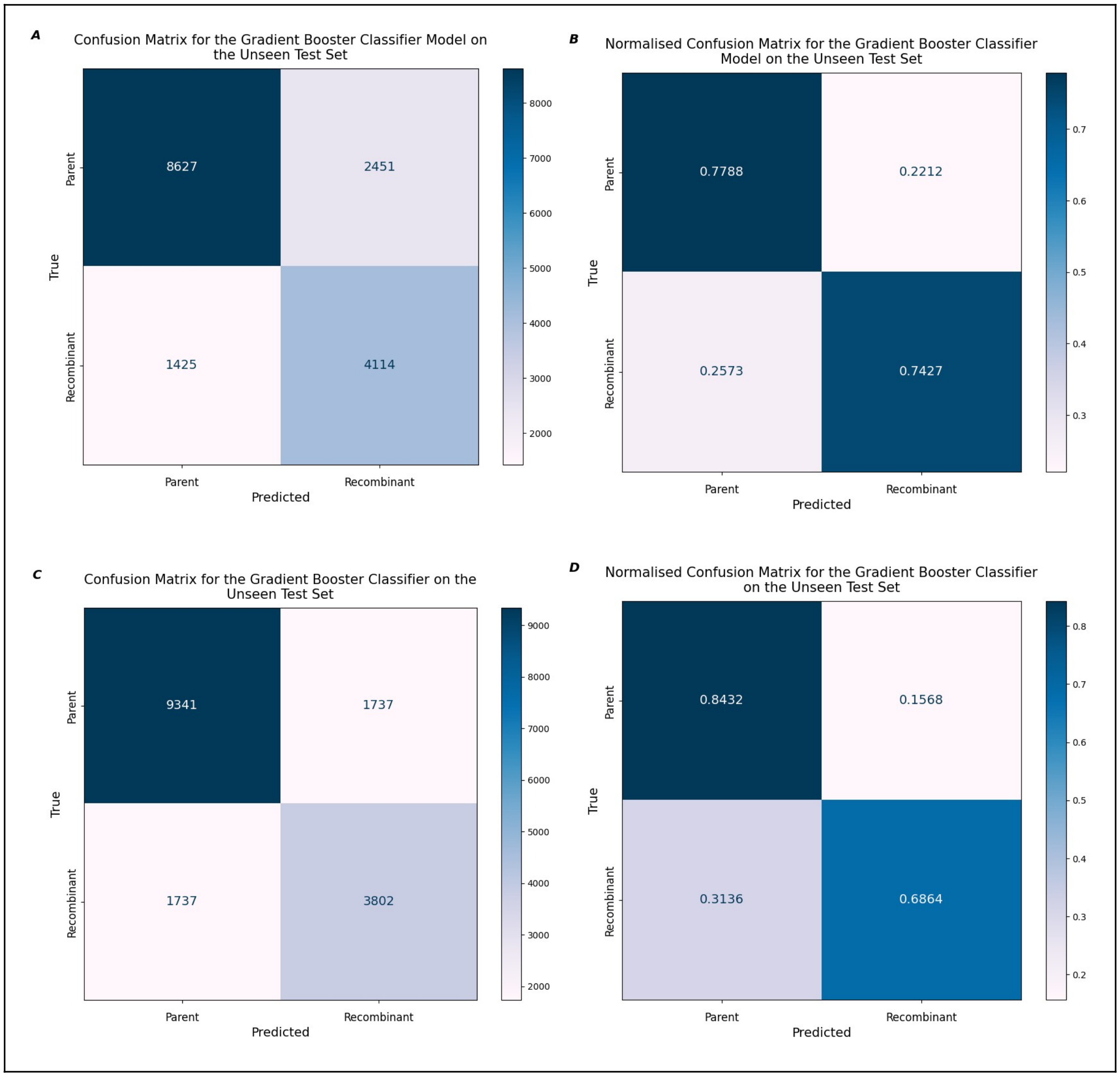
Gradient Booster Classifier: Dependent Decision on the Test Set

When employing dependent decision making, where the sequence with the highest recombinant probability score is selected from each triplet, the gradient booster's overall performance metrics improved with accuracy increasing to 79.09%, with an improvement in parent detection with 9 341 out of 11 078 parents (84.32%) being correctly identified. This unfortunately was at the cost of recombinant detection with only 3 802 out of 5 539 recombinants (68.64%) being correctly identified (*figure 2.3 C&D*), down from 4114 (74.27%).

Table 2.5 shows the performance of the gradient booster classifier whilst making dependent predictions on the test set.				
	Precision	Recall	f1-score	Support
Parent	0.8432	0.8432	0.8432	11 078
Recombinant	0.6864	0.6864	0.6864	5539

The dependent decision approach resulted in fair precision and recall scores of 0.8435 for parents and 0.6869 for recombinants (*table 2.5*). These metrics are a small improvement over both RDP's baseline performance (0.8398 precision, recall for parents; 0.6795 for recombinants) and the logistic regression model's dependent decision results (0.8354 precision, recall for parents; 0.6709 for recombinants).

The model's overall discriminatory power improved under dependent decision making, achieving an AUC of 0.7648 (*figure 2.3 E*). Similarly to the previous model, the independent version is able to achieve a higher true positive rate, at the cost of a higher false positive rate.



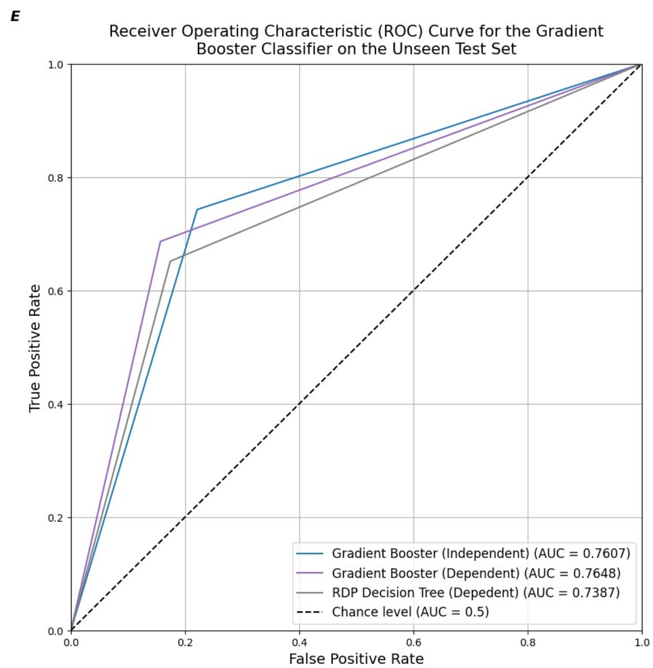


Figure 2.3 *shows the performance of the gradient boosterclassifier on the test set. This figure shows the confusion matrices for the independent and dependent decision making process and the ROC curves plotted together for comparison against RDPs decision tree.*
A: *Shows the confusion matrix for the gradient boostermodel on the test dataset making binary decisions*
B: *Shows the normalised confusion matrix for the gradientbooster model on the test dataset making binary decisions.*
C: *Shows the confusion matrix for the gradient boostermodel on the test dataset whilst making dependent predictions.*
D: *Shows the normalised confusion matrix for the gradientbooster model on the test dataset whilst making dependent predictions.*
E: *Shows the receiver operating characteristic (ROC) curve for the gradient booster model. The blue line indicates the independent decision making (AUC of 0.7607) and the purple line shows the dependent decision making (AUC of 0.7648), a random chance line plotted for reference with an AUC 0.5.*

Random Forest

Random forests were explored as a third model due to their ability to handle complex classification tasks through ensemble learning of multiple decision trees. By leveraging both bootstrap aggregation and random feature selection at each node, random forests can reduce overfitting while capturing non-linear relationships in the data. Although structurally more complex than logistic regression, random forests maintain good interpretability through feature importance rankings while offering robust performance across diverse datasets.

Random Forest: Binary Decisions on the Test Set

Building upon the gradient booster results, the random forest classifier was evaluated on the same test dataset of 16 617 sequences. When making binary decisions, the model achieved an overall accuracy of 78.00%, demonstrating comparable but slightly lower performance than the gradient booster classifier.

The confusion matrices (*figures 2.4 A&B*) show that the model correctly identified 9 194 out of 11 078 parents (82.99%) and 3 768 out of 5 539 recombinants (68.03%). These results translated to precision scores of 0.8385 for parents and 0.6667 for recombinants, with recall values of 0.8299 and 0.6803 for parents and recombinants respectively (*table 2.6*). The model's ROC curve analysis yielded an AUC of 0.7551 (*figure 2.4 E*), falling short of the independent gradient booster (0.7607) but surpassing both RDP's decision tree (0.7387) and the independent logistic regression model (0.7496).

Table 2.6 shows the performance metrics for the random forest classifier on the test set.

	Precision	Recall	f1-score	Support
Parent	0.8385	0.8299	0.8342	11 078
Recombinant	0.6667	0.6803	0.6734	5 539

Random Forest: Dependent Decisions on the Test Set

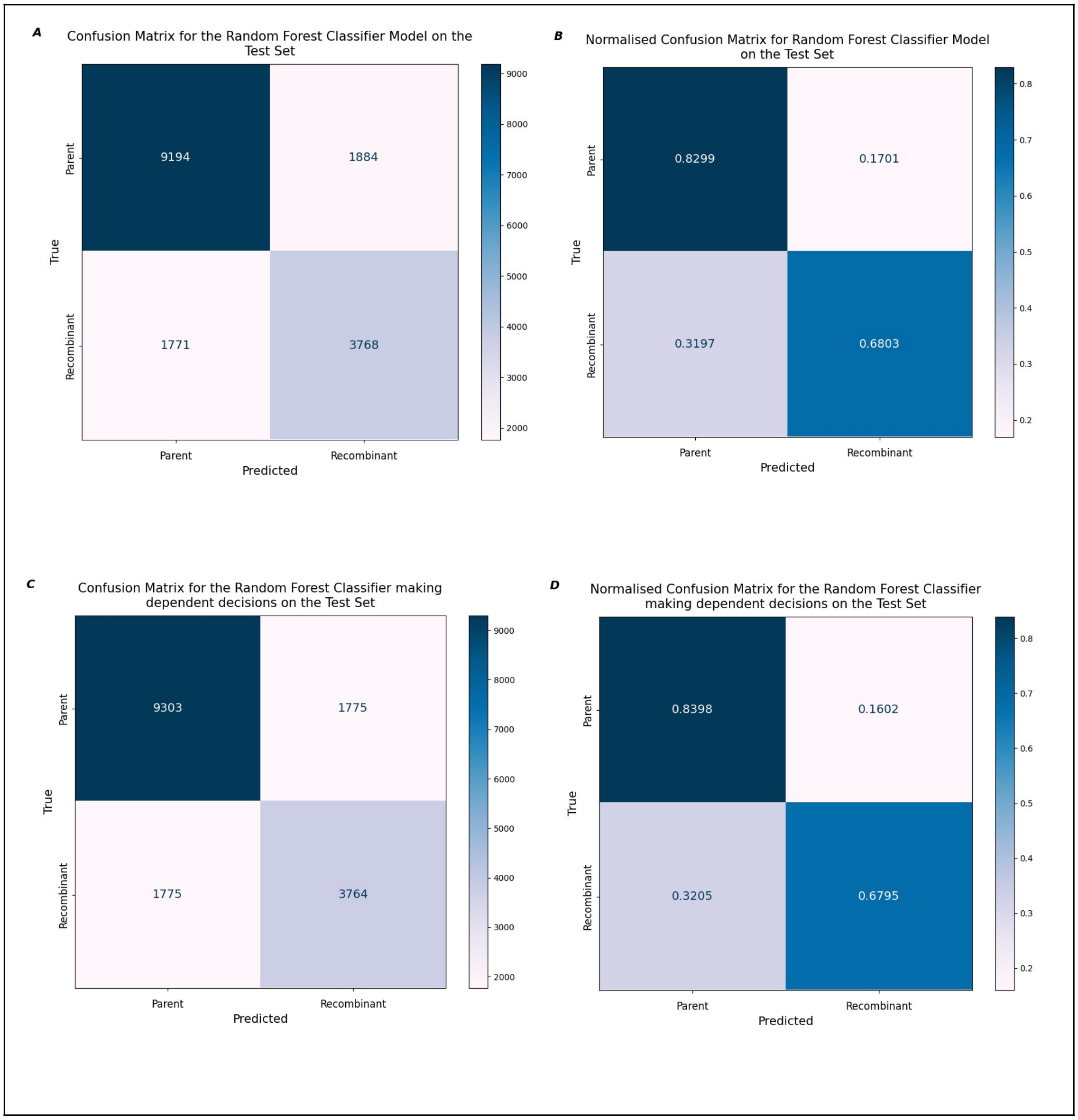
When employing dependent decision making, the random forest classifier showed improvement, achieving an overall accuracy of 78.64%. The confusion matrices (*figures 2.4 C&D*) reveal that 9 303 out of 11 078 parents (83.98%) and 3 764 out of 5 539 recombinants (67.95%) were correctly identified. The dependent decision approach resulted in precision and recall scores of 0.8398 for parents and 0.6795 for recombinants (*table 2.7*). Thus being the first model tested to have similar performance in independent and dependent decision making modes. The model's dependent decision making ROC curve showed an AUC of 0.7597 (*figure 2.4 E*), an improvement over its binary decision performance but still not matching the discriminatory power of the gradient booster's dependent approach (0.7648).

Table 2.7 shows the performance metrics for the random forest classifier on the test set whilst making dependent decisions

	Precision	Recall	f1-score	Support
Parent	0.8398	0.8398	0.8398	11 078
Recombinant	0.6795	0.6795	0.6795	5 539

Comparing and contrasting dependent decision models show that the gradient booster classifier demonstrated marginally superior performance, achieving an ROC AUC of 0.7648 and correctly identifying 68.64% of the recombinants (*figure 2.3*). The logistic regression model showed slightly lower performance with an ROC AUC of 0.7532 and correctly identifying 67.09% recombinants (*figure 2.2*). Whilst the random forest classifier achieved an

ROC AUC of 0.7597 and correctly identified 67.95% of the recombinants (*figure 2.4*). All of these models outperformed RDP5s baseline decision tree which achieved an ROC AUC 0.7387 and correctly identified 65.16% of recombinants (*figure 2.1*).



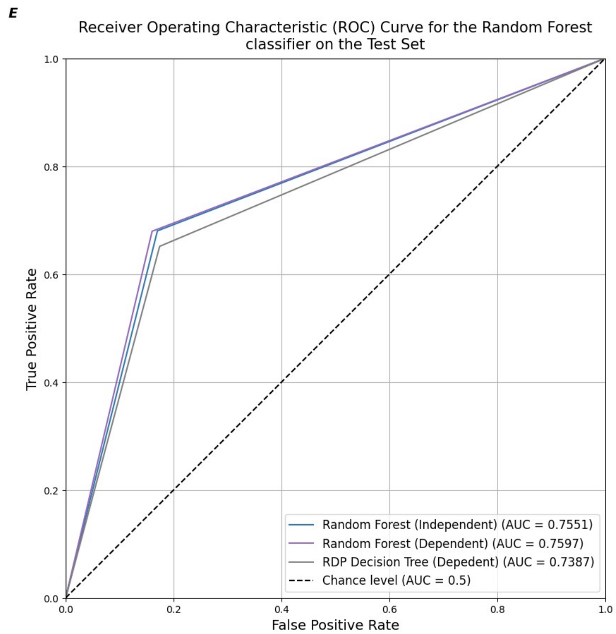


***Figure 2.4**: shows the performance of the random forest classifier on the test set. This figure shows the confusion matrices for the independent and dependent decision making process and the ROC curves plotted together for comparison against RDPs decision tree.*

***A:** Shows the confusion matrix for the random forestmodel on the test data.*

***B:** Shows the normalised confusion matrix for the randomforest model on the test data.*

***C:** Shows the confusion matrix for the random forestclassifier on the test dataset making dependent decisions*

***D:** Shows the normalised confusion matrix for the randomforest classifier on the test dataset making dependent decisions.*

***E**: Shows the receiver operating characteristic (ROC) curve for the random forest classifier. The blue line indicates the independent decision making (AUC of 0.7551) and the purple line shows the dependent decision making (AUC of 0.7597), a random chance line plotted for reference with an AUC 0.5.*

Binary Classifier Neural Network

Neural networks were explored as a more sophisticated approach to the recombinant identification problem, given their ability to learn complex, non-linear relationships between features and their proven effectiveness in biological sequence analysis tasks. The binary approach neural network was designed to process the same features as the previous models but with the added benefit of being able to learn hierarchical representations through multiple layers of neurons. This architecture was chosen to potentially capture subtle patterns in the recombination statistics that simpler models might miss, whilst maintaining the straightforward binary classification framework established by the earlier models.

Neural Network: Independent Decisions using the Binary Classifier Neural Network

The binary classifier neural network demonstrated strong performance when evaluated on the test dataset. Making independent binary decisions, the model achieved an accuracy of

76.75%. Looking at the confusion matrices (*Figure 2.5 A&B*), the model correctly identified 8 680 out of 11 078 parents (78.35%) and 4 073 out of 5 539 recombinants (73.53%).

The model achieved precision scores of 0.8555 for parents and 0.6294 for recombinants (*Table 2.7*), with recall values of 0.7835 and 0.7353 respectively. These metrics indicate good detection sensitivity for both classes, though the lower precision for recombinants indicates a higher false positive rate compared to parent identification. The ROC curve analysis yielded an AUC of 0.7594 (*Figure 2.5 E*), lower than the gradient booster (0.7607) but higher than RDP5's baseline (0.7387).

Table 2.7 showing performance metrics for the binary classifier neural network on the test set.

	Precision	Recall	f1-score	Support
Parent	0.8555	0.7835	0.8179	11 078
Recombinant	0.6294	0.7353	0.6783	5 539

Neural Network: Dependent Decisions using the Binary Classifier Neural Network

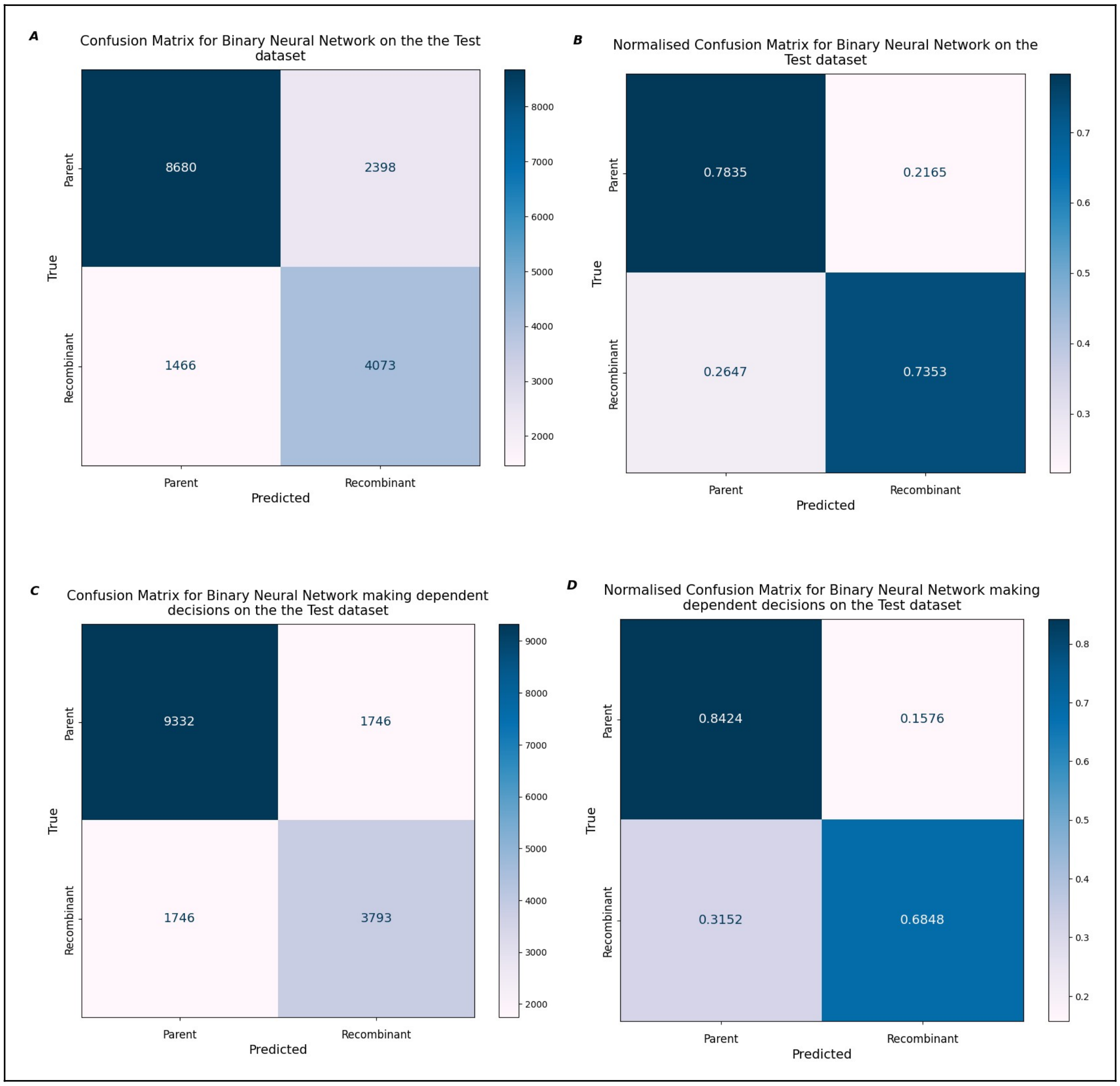
When employing dependent decision making, the model's performance improved, reaching an accuracy of 78.99%. The confusion matrices (*Figures 2.5 C&D*) show that 9 332 out of 11 078 parents (84.24%) and 3 793 out of 5 539 recombinants (68.48%) were correctly identified.

The dependent decision approach yielded precision and recall scores of 0.8424 for parents and 0.6848 for recombinants (*Table 2.8*). These results surpass RDP5's baseline performance and are almost equivalent to the gradient booster classifier's effectiveness.

Table 2.8 shows the performance metrics for the binary classifier neural network making dependent decisions on the test set.

	Precision	Recall	f1-score	Support
Parent	0.8424	0.8424	0.8424	11 078
Recombinant	0.6848	0.6848	0.6848	5 539

The model's discriminatory power under dependent decision making showed an AUC of 0.7636 (*Figure 2.5 E*), representing a slight improvement over its binary decision performance (0.7496). As with previous models, the independent version achieved a higher true positive rate at the cost of an increased false positive rate.



E

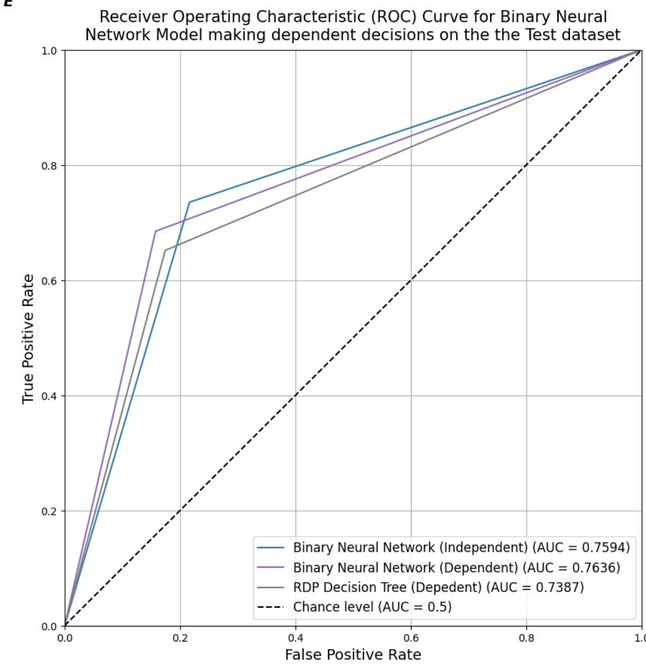


Figure 2.5: *the performance of the binary neural network on the test set. This figure shows the confusion matrices for the independent and dependent decision making process and the ROC curves plotted together for comparison against RDPs decision tree.*

A: *showing the confusion matrix for the binary classifierneural network making independent decisions on the test data.*

B: *showing the normalised confusion matrix for thebinary classifier neural network on the test dataset making independent decisions*

C: *showing the confusion matrix for the binary classifierneural network on the test dataset making dependent decisions*

D: *showing the normalised confusion matrix for thebinary classifier neural network on test set making dependent decisions*

E: *showing the receiver operating characteristic curve for the binary classifier neural network on the test dataset making both independent decisions (blue) and dependent decisions in (purple) with ROC AUCs of 0.7594 and 0.7636 respectively.*

Comparing the binary prediction method models' performances on the test shows a hierarchy in their effectiveness (*figure 2.6)*. The gradient booster classifier ranked first compared to the other models with an AUC of 0.7648, followed by the binary neural network at 0.7636, then the random forest classifier with an AUC of 0.7597, and finally the logistic regression model with an AUC of 0.7532. All of these models outperformed RDP5's decision tree in terms of AUC with RDP achieving 0.7387.

In terms of precision and recall metrics, all of the binary problem models showed stronger performance in identifying parent sequences (precision ranging from 0.8354 to 0.8435) compared to recombinants (precision ranging from 0.6709 to 0.6869). These results demonstrate that whilst modern machine learning approaches can improve upon RDP5's baseline performance, the margins of improvement are modest, suggesting that the existing RDP5's algorithm has nearly reached the limit of what the statistics these models are based on can achieve.

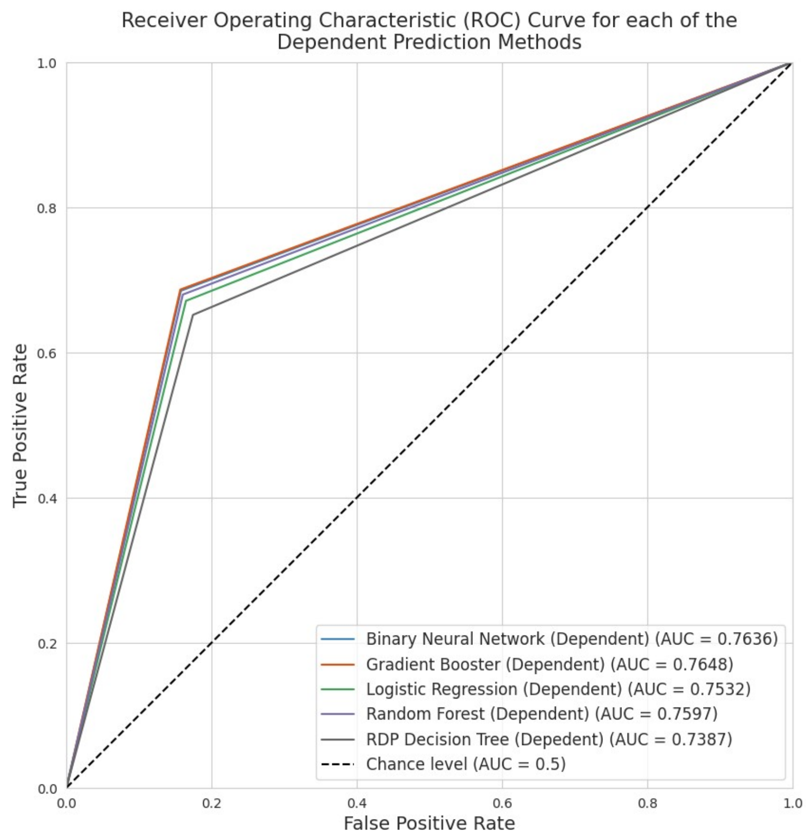


Figure 2.6: *Shows the receiver operating characteristicscurve for all of binary models utilising the dependent prediction techniques. This figure shows very similar results across all of the models with the AUCs ranging from 0.7387 for RDPs decision tree, to 0.7648 for the gradient booster classifier.*

Neural Network (Position Selection Method)

The position selection neural network represents a different approach to recombinant identification compared to the previously described models. Rather than treating recombinant identification as a binary classification problem, this architecture was designed to simultaneously analyse all three sequences in a potential recombination event (the putative recombinant and its two potential parents). By processing the complete set of 96 features from all three sequences concurrently, the model aimed to capture complex relationships and patterns that might be missed when examining sequences in isolation.

This approach utilises sparse categorical cross-entropy as its loss function, which is well-suited for multi-class classification problems where the goal is to select one option from multiple possibilities. In this case, the model must determine *which* of the three sequences in a triplet is most likely to be the recombinant. This thought process is similar to the 'dependent' prediction method utilised in the previous models taking into account their binary predictions. The 'sparse' implementation was chosen for computational efficiency, as it allows the model to work with integer class labels rather than one-hot encoded vectors.

The neural network's architecture was specifically designed to handle the increased complexity of processing all sequences simultaneously. Through a series of dense layers with decreasing dimensionality (90, 60, 30 and 3 nodes), interspersed with batch normalisation and dropout layers for regularisation, the network progressively refines its feature representations before making a final three-way classification decision. This graduated reduction in layer size aims to distil the most relevant information for recombinant identification whilst minimising noise and overfitting.

Position selection method on the test dataset

The triplet method neural network performed moderately well on the unseen test dataset, achieving an overall accuracy of 71.17%, this accuracy should not be compared to the overall accuracy from the previous models as they take into account parents, this accuracy figure only represents recombinants correctly identified. With this frame of reference the model’s performance is superior to all of the other models when making dependent decisions, as this model inherently is doing so.

Analysis of the confusion matrices (*figures 2.7 A&B*) reveals the model's predictive capabilities across the three positions. Of the 1 849 recombinants occurring in position 1, 1102 (59.60%) were correctly identified from the triplet provided to the model. Similarly, for position 2, 1326 of 1759 recombinants (75.38%) were correctly classified, while position 3 saw 1514 correct identifications from 1931 recombinants (78.40%). This resulted in a weighted average detection rate of 71.17% across all positions.

Table 2.9 showing performance metrics for the position selection neural network (utilising sparse categorical cross entropy) on the test dataset.

	Precision	Recall	f1-score	Support
Recombinant in Position 1	0.6146	0.6236	0.6191	1849
Recombinant in Position 2	0.738	0.7328	0.7355	1759
Recombinant in Position 3	0.7600	0.7545	0.7573	1931

The confusion matrices (*figures 2.6 A&B*) show the position selection neural network's varying effectiveness at identifying recombinant positions within the test dataset. The model demonstrates strongest performance for Position 3, correctly identifying 1514 cases (78.40%), followed by Position 2 with 1326 correct identifications (75.38%), whilst struggling most with Position 1, accurately identifying only 1102 cases (59.60%).

When making errors with Position 1 recombinants, the model shows nearly equal likelihood of misclassifying to Position 2 (367 cases, 19.85%) or Position 3 (380 cases, 20.55%). For Position 2 recombinants, there is a notable bias toward Position 1 misclassifications (312 cases, 17.74%) over Position 3 (121 cases, 6.88%). Position 3 recombinants saw the fewest errors overall, but still with a preponderance towards selecting position 1 with 281 cases (14.55%) misclassified as Position 1 and 136 cases (7.04%) as Position 2.

Figure 2.6 C shows the ROC curve for the model, with each position plotted as a separate class. The weighted AUC for this model is 0.784, with the constituent AUCs being 0.718, 0.810, 0.823 for positions 1 through 3 respectively. Reviewing the curve also emphasises the weaker performance in position 1 with a significantly worse true positive rate (~0.60) at a false positive rate of 20%, compared to position 2 and 3 (~0.80%).

This behaviour is peculiar as steps were taken to prevent positional biases. Consider that the model is predicting the position of the recombinant from within the triplet of parent, parent, recombinant. When given these three choices, with the assumption that a recombinant is present, then the recombinant has to be in position 1, 2, or 3 in that triplet. Therefore, the model is always attempting to identify the recombinant and pick which position the recombinant is in. The model is not interested in the position of the parents.

In order to prevent positional biases steps were taken to shuffle, and stratify the dataset such that there was an equal distribution of recombinants across the positions (33% in position 1, 2, and 3) in the training, validation, and test sets. Furthermore, during training the model was shown an equal distribution of training samples in each batch so that it would not learn that a certain position was favoured.

It is possible that amongst the recombinants in position 1 there exist particularly difficult examples of recombination. During the shuffling step, samples were moved out of position 1, along with their relevant statistics, into position 2 and 3 until there was an equal balance. Samples were moved out of position 1, where there was originally a majority, as the data was originally ingested from RDP5's decision tree which ordered the samples according to its predictions (~65% correct), hence 65% of the recombinants were in position 1. Thus, it could be that the samples left over (despite random sampling) in position 1 are complex examples of recombination which the model is struggling to correctly place.

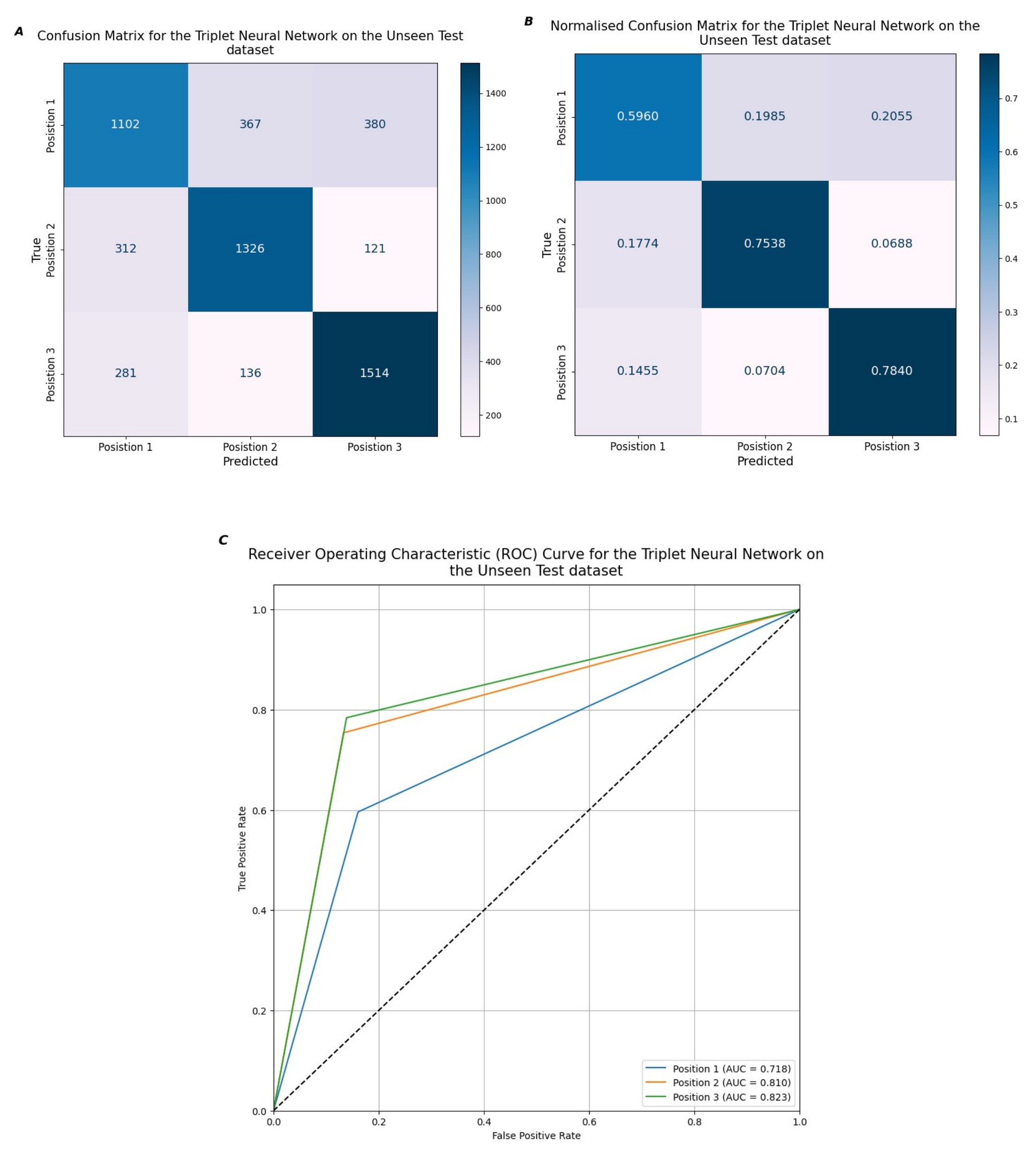


Figure 2.6: *showing the performance of the position selection neural network on the test set. This figure shows the confusion matrices for the model and the ROC curve plotted against RDP5's decision tree for comparison.*

A: *showing the confusion matrix for the position selectionneural network on the test data.*

B: *showing the normalised confusion matrix for theposition selection neural network on the test data.*

C: *showing the receiver operating characteristic curve for the position selection neural network on the test dataset, with AUCs of 0.718, 0.810, 0.823 for positions 1, 2, 3 respectively.*

Feature Contributions

Understanding feature importance in machine learning models is crucial for both model interpretation and improvement of underlying methodologies. In the context of viral recombinant detection, feature importance analysis allows us to understand which statistical measures are most effective at identifying recombinant sequences, potentially leading to more efficient and accurate inference. This type of analysis is particularly valuable in bioinformatics, where computational efficiency and biological interpretability are often as important as raw performance metrics.

Figures 2.7 and 2.8 present the relative importance of the statistics used by the logistic regression model. These visualisations provide insight into which statistical measures are most informative for detecting recombination events. Logistic regression was used for this analysis as, unlike more complex models such as neural networks, it provides clear, interpretable weights for each feature (*Figure 2.7*).

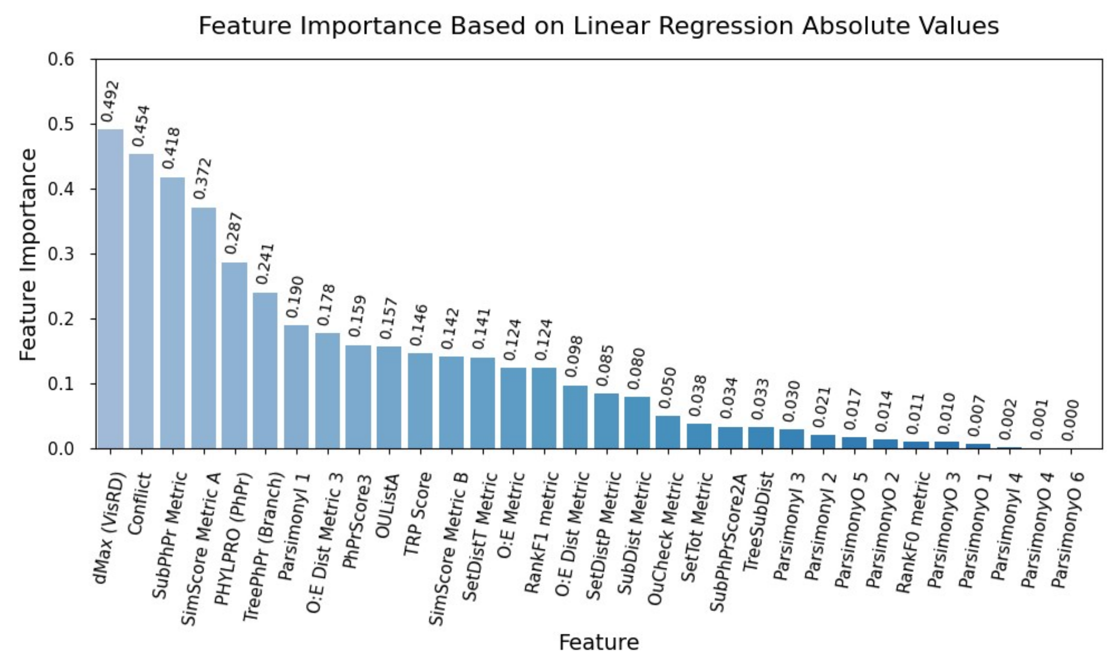


Figure 2.7: *Shows the relative feature importance of each statistic outputted from RDP5. The values are based on the logistic regression model. They are calculated by taking the absolute value of the weight applied to each feature by the logistic regression, indicating the magnitude of each feature's contribution regardless of whether it positively or negatively correlates with the correct identification of recombinants.*

The most influential features are the dMax (VisRD) and Conflict methods. Indicating that these features capture particularly meaningful signals with respect to recombinant identification. dMax refers to the VisRD method described in the methods section that utilises maximum parsimony trees. The Conflict statistic looks at the degree to which distances are smaller between potential co-recombinant sequences than with other sequences in the alignment.

Figure 2.8 provides additional context by showing the actual standardised weight values (including their signs) assigned by the logistic regression model, revealing whether features positively or negatively correlate with recombination probability. Since these features were standardised (zero mean, unit variance), the weights directly indicate the relative importance of each feature in the model's decision-making process - a weight of 2 indicates that a one standard deviation increase in that feature doubles the log-odds of predicting a recombinant, while a weight of -2 indicates that a one standard deviation increase halves the log-odds.

The standardisation of features ensures that the weights are comparable across different statistics, regardless of their original scales. This is particularly important as the raw statistics from RDP5 can have vastly different ranges and units. For example, dMax and PhPr scores naturally exist on different scales, but after standardisation, their weights can be directly compared to assess their relative importance in the model.

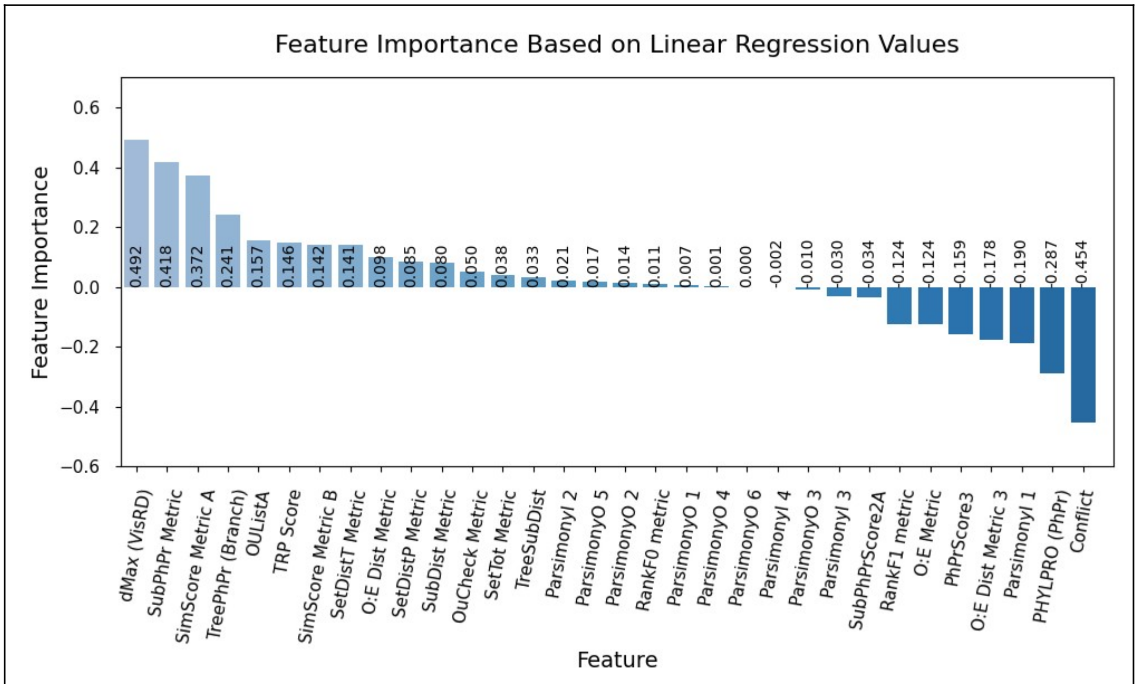


Figure 2.8: *Shows the feature importance of each statisticoutputted from RDP5. The values are based on the logistic regression model. They are calculated by taking the weight applied to each feature by the logistic regression.*

Several statistics show strong negative correlations with recombination probability, suggesting their values tend to decrease in the presence of recombination. However, interpretation must be careful - these negative weights don't necessarily indicate poor performance of these statistics, but rather that lower values of these statistics are predictive of recombination events when considered alongside other features in the model. I.e. a low Conflict value is weighted heavily by the model to be associated with recombination identification.

This information is particularly useful for future projects aiming to utilise these statistics. For instance, the inclusion of features with near zero importance in the decision tree and other methods used by RDP5 to identify recombinants, is questionable: these features are clearly failing to provide any substantially useful information with respect to the accurate identification of recombinant sequences in the datasets analysed. Although it is possible that these features might have greater utility in niche situations that the analysed dataset does not capture, it would perhaps be better to improve computational efficiency by simply not calculating most of the subtree-prune and regraft statistics that are represented by the ParsimonyO features.

Examination of Misclassified Samples

The analysis of misclassified samples reveals important patterns in how RDP5's various recombinant identification statistics relate to classification errors. The effect size analysis (*Figure 2.9*) identifies which statistics differ most substantially between correctly and incorrectly classified samples using Cohen’s d. Cohen's d measures the standardised difference in means between two groups - in this case, the difference in statistical values between correctly and incorrectly classified recombinant samples. The effect size is calculated by dividing the difference between the means by their pooled standard deviation. Cohen’s d values around 0.2 indicate small effect, 0.4 medium effects, and above 0.8 indicate large effects.

Reviewing *figure 2.9* shows how different statistics vary between correctly and incorrectly classified samples. The PHYLPRO metric has the largest positive effect ($d \approx 1.6$). This means that when a recombinant is misclassified as a parent, its PHYLPRO value is typically 1.6 standard deviations higher than the PHYLPRO values of correctly classified recombinants. Similarly, SimScore Metric A shows the largest negative effect ($d \approx -1.8$), indicating that when recombinants are misclassified, their SimScore A values are typically 1.8 standard deviations lower than those of correctly classified recombinants. These substantial differences in values between correct and incorrect classifications suggest that these statistics show strong systematic biases in cases where the model fails to correctly identify recombinants.

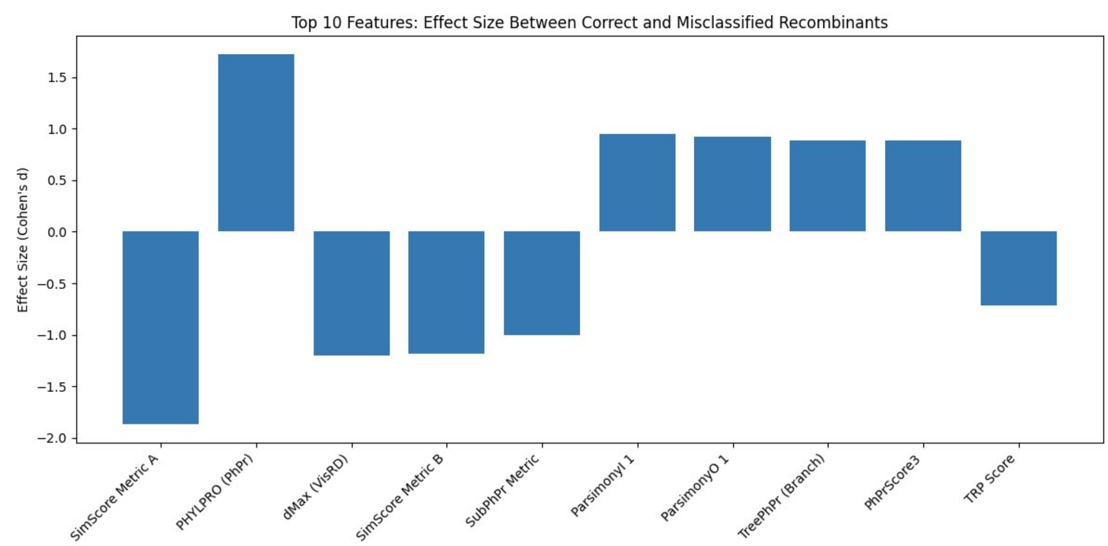


Figure 2.9: *Effect size analysis showing the relative importance of the 10 most divergent statistics for classification errors. The y-axis shows Cohen's d effect size, measuring the standardised difference in feature values between correctly and incorrectly classified samples. Positive values indicate the statistic tends to be higher in misclassified samples, whilst negative values indicate lower values in misclassified samples.*

The correlation matrix (*Figure 2.10*) provides insight into how these statistics interact - specifically in cases where recombinants were misclassified as parents. Notable patterns emerge when examining the relationships between different statistics:

1. dMaxA (VisRD) shows moderate negative correlations with both TreePhPr (Branch) (-0.48) and SubPhPr Metric (-0.25), suggesting these statistics often provide contradictory signals during misclassification.

2. SimScore Metric A exhibits a moderate positive correlation (0.40) with SimScore Metric B , indicating these related statistics tend to agree in misclassified cases. This relationship is expected given that both statistics are derived from similar underlying calculations, but the moderate strength suggests they still capture somewhat distinct aspects of the sequence relationships.

3. SubPhPr Metric's negative correlations with other statistics, ranging from -0.07 with SimScore Metric A to -0.25 with dMaxA, is particularly interesting from a biological perspective. SubPhPr examines phylogenetic profiles using genetic distances; it operates on the principle that correlations between the difference in distances

between the recombinant and other sequences should be lower than those of parental sequences when comparing genome regions between breakpoints to the remainder of genome regions . Its contrasting behaviour might indicate:

a. Cases where a very high proportion (such as >~30%) of the sequences in an alignment all share evidence of the same recombination event.
b. Cases where recombination has occurred between closely related sequences
c. Situations where multiple recombination events have partially overwritten each other
d. Instances where selection pressure has caused convergent recombination events (i.e. independent recombination events involving similar parental lineages and similar genome regions.

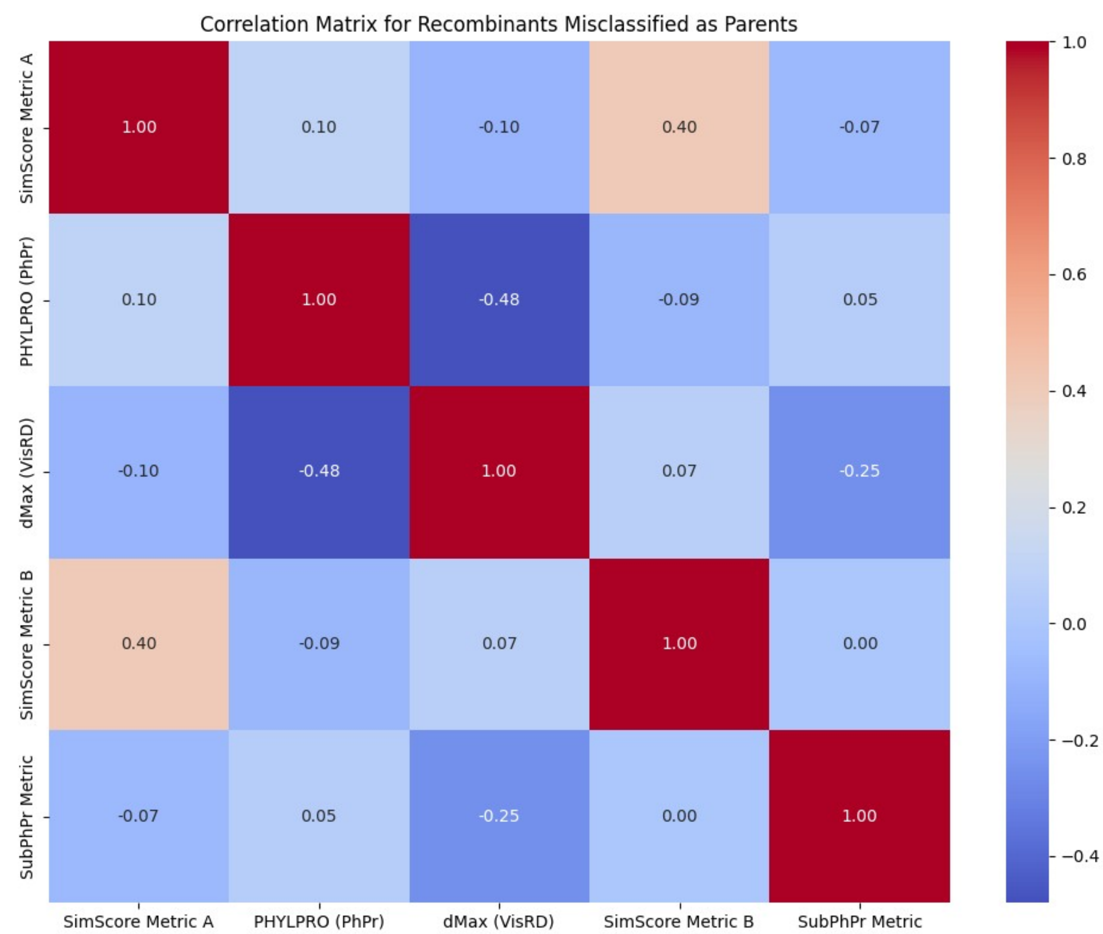


Figure 2.10: *Correlation matrix showing the relationships between the 10 most divergent statistics in misclassified samples. Red colours indicate positive correlations, whilst blue indicates negative correlations. The intensity of the colour and the number in each cell indicates the strength of correlation, ranging from -1 to 1.*

Many pairs of statistics show very weak correlations (near 0), such as SimScoreBA with SubPhPrScoreA (0.00) and PhPrScoreA (-0.09). These weak correlations suggest that different statistics are capturing independent aspects of sequence relationships, which could be valuable for improving classification accuracy if properly combined. However, this independence also reveals that even when examining only misclassified cases, different statistics can provide conflicting signals about the underlying sequence relationships. This complexity in the statistical interactions may help explain why accurate recombinant identification remains challenging, as different methods of measuring sequence relationships can yield contradictory evidence about which sequences are recombinant.

Integration of the Algorithms into RDP5/6

The integration of these machine learning models into RDP5 has already begun. Specifically an early version of the logistic regression model was trialled as an addition to RDP5. This was done by creating a simple logistic regression pseudo code model with the correct weights and then ensuring the correct pre-processing was applied.

Further models can be connected to RDP5 via a python-based machine learning module interfacing with the existing codebase. This module will primarily utilise the position selection neural network, with the gradient booster or logistic regression available as a fallback option. The enhancement would be implemented as an optional detection mode within RDP5, allowing users to leverage the potentially improved recombination detection capabilities whilst maintaining access to traditional consensus scores. By implementing the machine learning enhancement as an optional feature, RDP5's core functionality would be preserved whilst providing a pathway for gradual adoption by users. The modular nature of this implementation would also facilitate future updates as machine learning techniques advance, ensuring the system remains current and continues to provide optimal recombination detection accuracy.

Discussion & Conclusions

This thesis demonstrates the potential of machine learning approaches to enhance the detection of viral recombination events compared to existing methods. Through training models on large, simulated datasets spanning diverse viral evolution scenarios, we achieved modest but consistent improvements in recombinant identification accuracy over traditional approaches.

The position selection neural network demonstrated particularly strong performance, achieving a weighted AUC of 0.784 on the test dataset, outperforming RDP5's baseline AUC of 0.739. This model's success likely stems from its ability to simultaneously analyse all three sequences in potential recombination events, allowing it to capture complex relationships between putative recombinants and their potential parents. However, the model showed varying performance across different sequence positions, with notably weaker performance for position 1 (AUC 0.718) compared to positions 2 and 3 (AUCs 0.810 and 0.823 respectively). This positional bias persisted despite careful dataset balancing, suggesting underlying complexities in the recombination signals that warrant further investigation.

The gradient boosting classifier also showed promising results, achieving an AUC of 0.765, whilst the binary neural network achieved 0.763. These performances, though modest, represent meaningful improvements over RDP5's baseline. The logistic regression model achieved an AUC of 0.753, demonstrating that even simpler machine learning approaches can provide comparable results to existing methods.

A key strength of this work lies in its use of simulated data to generate a comprehensive training dataset encompassing diverse viral recombination scenarios. By varying factors such as recombination rate, mutation rate, genome size, population dynamics, and selection pressures, the simulated data captured a broad range of evolutionary conditions. This diversity helps ensure the trained models' robustness and generalisability to real-world viral datasets.

The incorporation of challenging test cases dissimilar from the training data provided a more stringent and realistic evaluation of model performance. In real-world applications, recombinant identification tools frequently encounter complex and atypical cases. The models' ability to maintain performance improvements on this adversarial test set suggests they are equipped to handle such outliers.

Feature importance analysis revealed that certain RDP5 statistics, particularly dMax and BadDist methods, contribute substantially more to accurate recombination detection than others. Notably, some of the statistics (such as Rcompat features) showed minimal importance, suggesting potential computational efficiencies could be gained by eliminating these calculations without significantly impacting detection accuracy.

The use of RDP5's existing recombination identification methods to generate input features represents another strength of this project. By leveraging these established methods, this approach builds upon and enhances on stalwarts in the field. The models effectively learn to interpret the combined signals from multiple recombination detection methods, allowing them to make more accurate predictions than any single method alone.

However, there are also some limitations to consider. While the simulated training data covered a wide range of scenarios, it may not capture all the complexities and nuances of real evolutionary processes (92). In real viral populations, selection pressures operate at multiple levels simultaneously. Viruses must maintain functional proteins, preserve RNA secondary structures, and optimise codon usage whilst evading host immune responses. These selective forces create intricate patterns of sequence conservation and variation that are difficult to model accurately in simulations.

Additionally, the interpretability of complex machine learning models, particularly neural networks, presents a challenge. Unlike simpler methods, such as logistic regression, with clear decision-making criteria, the reasoning behind neural network predictions can be opaque. This 'black box' nature can make it difficult to diagnose errors or unexpected behaviour. Future work could explore techniques for interpreting and visualising neural network decision-making, such as attribution maps and saliency analysis, to improve transparency.

The performance ceiling encountered in this study, where even the most sophisticated models achieved only modest improvements over RDP5's baseline, suggests that the limiting factor may not be the machine learning approaches themselves, but rather the underlying statistics they rely upon. The statistics generated by RDP5's existing methods, while proven effective, may not capture all available signals of recombination events. Developing new statistical measures that can detect previously unexploited signatures of recombination could provide additional discriminatory power, potentially enabling more substantial improvements in recombinant identification accuracy. This highlights the importance of continuing to innovate not just in the computational methods used to analyse sequence data, but also in the fundamental approaches we use to characterise recombination events.

Despite the limitations, the integration of these machine learning models into existing recombination detection pipelines represents a promising direction for future development. The successful implementation of an early version of the logistic regression model into RDP5 demonstrates the feasibility of such integration. Future work could focus on incorporating the more sophisticated models, particularly the position selection neural network, whilst maintaining the software's accessibility and ease of use.

In conclusion, this thesis demonstrates that machine learning models, particularly neural networks designed to consider entire sequence triplets simultaneously, can enhance viral recombinant identification accuracy. While the improvements are modest, they represent meaningful progress in automated recombination detection. The successful integration of these approaches into RDP5 provides a foundation for future developments in this critical area of viral genomics research.

References

1. Baltimore D. Expression of animal virus genomes. Bacteriol Rev. 1971 Sep;35(3):235–41.

2. Upadhyay S. Evolution and transmission of viruses. In: Viral Infections and Antiviral Therapies [Internet]. Elsevier; 2023 [cited 2023 Oct 24]. p. 39–54. Available from: https://linkinghub.elsevier.com/retrieve/pii/B9780323918145000143

3. Holmes EC. Error thresholds and the constraints to RNA virus evolution. Trends Microbiol. 2003 Dec;11(12):543–6.

4. Pérez-Losada M, Arenas M, Galán JC, Palero F, González-Candelas F. Recombination in viruses: Mechanisms, methods of study, and evolutionary consequences. Infect Genet Evol. 2015 Mar;30:296–307.

5. Scheel TKH, Galli A, Li YP, Mikkelsen LS, Gottwein JM, Bukh J. Productive Homologous and Non-homologous Recombination of Hepatitis C Virus in Cell Culture. PLOS Pathog. 2013 Mar 28;9(3):e1003228.

6. Austermann-Busch S, Becher P. RNA structural elements determine frequency and sites of nonhomologous recombination in an animal plus-strand RNA virus. J Virol. 2012 Jul;86(13):7393–402.

7. Bujarski JJ. RECOMBINATION OF VIRUSES. Encycl Virol. 1999;1446–53.

8. Simon-Loriere E, Holmes EC. Why do RNA viruses recombine? Nat Rev Microbiol. 2011 Jul 4;9(8):617–26.

9. Stedman KM. Deep Recombination: RNA and ssDNA Virus Genes in DNA Virus and Host Genomes. Annu Rev Virol. 2015;2(1):203–17.

10. Lander ES, Linton LM, Birren B, Nusbaum C, Zody MC, Baldwin J, et al. Initial sequencing and analysis of the human genome. Nature. 2001 Feb;409(6822):860–921.

11. Casjens S. Prophages and bacterial genomics: what have we learned so far? Mol Microbiol. 2003;49(2):277–300.

12. Frischmuth T, Stanley J. Recombination between viral DNA and the transgenic coat protein gene of African cassava mosaic geminivirus. J Gen Virol. 1998;79(5):1265–71.

13. Fraser MJ, Smith GE, Summers MD. Acquisition of Host Cell DNA Sequences by Baculoviruses: Relationship Between Host DNA Insertions and FP Mutants of Autographa californica and Galleria mellonella Nuclear Polyhedrosis Viruses. J Virol. 1983 Aug;47(2):287–300.

14. Lai MM. RNA recombination in animal and plant viruses. Microbiol Rev. 1992 Mar;56(1):61–79.

15. Grant HE, Hodcroft EB, Ssemwanga D, Kitayimbwa JM, Yebra G, Esquivel Gomez LR, et al. Pervasive and non-random recombination in near full-length HIV genomes from Uganda. Virus Evol. 2020 Jan 1;6(1):veaa004.

16. Baird HA, Galetto R, Gao Y, Simon-Loriere E, Abreha M, Archer J, et al. Sequence determinants of breakpoint location during HIV-1 intersubtype recombination. Nucleic

Acids Res. 2006 Oct 1;34(18):5203–16.

17. Galetto R, Giacomoni V, Véron M, Negroni M. Dissection of a Circumscribed Recombination Hot Spot in HIV-1 after a Single Infectious Cycle *. J Biol Chem. 2006 Feb 3;281(5):2711–20.

18. Zhang J, Temin HM. Retrovirus recombination depends on the length of sequence identity and is not error prone. J Virol. 1994 Apr;68(4):2409–14.

19. Jetzt AE, Yu H, Klarmann GJ, Ron Y, Preston BD, Dougherty JP. High Rate of Recombination throughout the Human Immunodeficiency Virus Type 1 Genome. J Virol. 2000 Feb;74(3):1234–40.

20. Shriner D, Rodrigo AG, Nickle DC, Mullins JI. Pervasive Genomic Recombination of HIV-1 in VivoSequence data from this article have been deposited with the EMBL/GenBank Data Libraries under accession nos. AY496645. Genetics. 2004 Aug 1;167(4):1573–83.

21. Neher RA, Leitner T. Recombination Rate and Selection Strength in HIV Intra-patient Evolution. PLOS Comput Biol. 2010 Jan 29;6(1):e1000660.

22. Williams A, Menon S, Crowe M, Agarwal N, Biccler J, Bbosa N, et al. Geographic and Population Distributions of Human Immunodeficiency Virus (HIV)–1 and HIV-2 Circulating Subtypes: A Systematic Literature Review and Meta-analysis (2010–2021). J Infect Dis. 2023 Nov 28;228(11):1583–91.

23. Hwang CK, Svarovskaia ES, Pathak VK. Dynamic copy choice: Steady state between murine leukemia virus polymerase and polymerase-dependent RNase H activity determines frequency of in vivo template switching. Proc Natl Acad Sci. 2001 Oct 9;98(21):12209–14.

24. Junghans RP, Boone LR, Skalka AM. Products of reverse transcription in avian retrovirus analyzed by electron microscopy. J Virol. 1982 Aug;43(2):544–54.

25. Negroni M, Buc H. Retroviral recombination: what drives the switch? Nat Rev Mol Cell Biol. 2001 Feb;2(2):151–5.

26. Nelson MI, Viboud C, Simonsen L, Bennett RT, Griesemer SB, George KS, et al. Multiple Reassortment Events in the Evolutionary History of H1N1 Influenza A Virus Since 1918. PLOS Pathog. 2008 Feb 29;4(2):e1000012.

27. Diaz A, Allerson M, Culhane M, Sreevatsan S, Torremorell M. Antigenic drift of H1N1 influenza A virus in pigs with and without passive immunity. Influenza Other Respir Viruses. 2013 Dec;7(Suppl 4):52–60.

28. Lucía-Sanz A, Manrubia S. Multipartite viruses: adaptive trick or evolutionary treat? Npj Syst Biol Appl. 2017 Nov 9;3(1):1–11.

29. McDonald SM, Nelson MI, Turner PE, Patton JT. Reassortment in segmented RNA viruses: mechanisms and outcomes. Nat Rev Microbiol. 2016 Jul;14(7):448–60.

30. McDonald SM, Patton JT. Assortment and packaging of the segmented rotavirus genome. Trends Microbiol. 2011 Mar 1;19(3):136–44.

31. Nibert ML, Margraf RL, Coombs KM. Nonrandom segregation of parental alleles in

reovirus reassortants. J Virol. 1996 Oct;70(10):7295–300.

32. Urquidi V, Bishop DHL. Non-random reassortment between the tripartite RNA genomes of La Crosse and snowshoe hare viruses. J Gen Virol. 1992;73(9):2255–65.

33. Qiu WP, Geske SM, Hickey CM, Moyer JW. Tomato Spotted WiltTospovirusGenome Reassortment and Genome Segment-Specific Adaptation. Virology. 1998 Apr 25;244(1):186–94.

34. Li X, Gu M, Zheng Q, Gao R, Liu X. Packaging signal of influenza A virus. Virol J. 2021 Feb 17;18(1):36.

35. Zeldovich KB, Liu P, Renzette N, Foll M, Pham ST, Venev SV, et al. Positive Selection Drives Preferred Segment Combinations during Influenza Virus Reassortment. Mol Biol Evol. 2015 Jun;32(6):1519–32.

36. Lowen AC. It's in the mix: Reassortment of segmented viral genomes. Spindler KR, editor. PLOS Pathog. 2018 Sep 13;14(9):e1007200.

37. Yoo E. Conformation and Linkage Studies of Specific Oligosaccharides Related to H1N1, H5N1, and Human Flu for Developing the Second Tamiflu. Biomol Ther. 2014 Feb;22:93–9.

38. Zimmer SM, Burke DS. Historical perspective--Emergence of influenza A (H1N1) viruses. N Engl J Med. 2009 Jul 16;361(3):279–85.

39. Cromie GA, Connelly JC, Leach DR. Recombination at double-strand breaks and DNA ends: conserved mechanisms from phage to humans. Mol Cell. 2001 Dec;8(6):1163–74.

40. Weller SK, Coen DM. Herpes simplex viruses: mechanisms of DNA replication. Cold Spring Harb Perspect Biol. 2012 Sep 1;4(9):a013011.

41. Muylaert I, Tang KW, Elias P. Replication and recombination of herpes simplex virus DNA. J Biol Chem. 2011 May 6;286(18):15619–24.

42. Wilkinson DE, Weller SK. The role of DNA recombination in herpes simplex virus DNA replication. IUBMB Life. 2003 Aug;55(8):451–8.

43. Weller SK, Sawitzke JA. Recombination promoted by DNA viruses: phage λ to herpes simplex virus. Annu Rev Microbiol. 2014;68:237–58.

44. Yu L, Majerciak V, Xue XY, Uberoi A, Lobanov A, Chen X, et al. Mouse papillomavirus type 1 (MmuPV1) DNA is frequently integrated in benign tumors by microhomology-mediated end-joining. PLoS Pathog. 2021 Aug;17(8):e1009812.

45. Truong LN, Li Y, Shi LZ, Hwang PYH, He J, Wang H, et al. Microhomology-mediated End Joining and Homologous Recombination share the initial end resection step to repair DNA double-strand breaks in mammalian cells. Proc Natl Acad Sci U S A. 2013 May 7;110(19):7720–5.

46. Schreiner S, Nassal M. A Role for the Host DNA Damage Response in Hepatitis B Virus cccDNA Formation—and Beyond? Viruses. 2017 May 22;9(5):125.

47. Weller SK, Schumacher AJ, Mohni KN. Herpes Simplex Virus: Manipulating DNA Damage Response Pathways. FASEB J. 2012;26(S1):932.2-932.2.

48. Muylaert I, Tang KW, Elias P. Replication and Recombination of Herpes Simplex Virus DNA*. J Biol Chem. 2011 May 6;286(18):15619–24.

49. Floyd R, Sharp DG. Aggregation of poliovirus and reovirus by dilution in water. Appl Environ Microbiol. 1977 Jan;33(1):159–67.

50. Floyd R, Sharp DG. Viral aggregation: effects of salts on the aggregation of poliovirus and reovirus at low pH. Appl Environ Microbiol. 1978 Jun;35(6):1084–94.

51. Floyd R, Sharp DG. Viral aggregation: buffer effects in the aggregation of poliovirus and reovirus at low and high pH. Appl Environ Microbiol. 1979 Sep;38(3):395–401.

52. Combe M, Garijo R, Geller R, Cuevas JM, Sanjuán R. Single-Cell Analysis of RNA Virus Infection Identifies Multiple Genetically Diverse Viral Genomes within Single Infectious Units. Cell Host Microbe. 2015 Oct;18(4):424–32.

53. Aguilera ER, Erickson AK, Jesudhasan PR, Robinson CM, Pfeiffer JK. Plaques Formed by Mutagenized Viral Populations Have Elevated Coinfection Frequencies. mBio. 2017 Mar 14;8(2):10.1128/mbio.02020-16.

54. Aguilera ER, Pfeiffer JK. Strength in Numbers: Mechanisms of Viral Co-infection. Virus Res. 2019 May;265:43–6.

55. Erickson AK, Jesudhasan PR, Mayer MJ, Narbad A, Winter SE, Pfeiffer JK. Bacteria Facilitate Enteric Virus Co-infection of Mammalian Cells and Promote Genetic Recombination. Cell Host Microbe. 2018 Jan;23(1):77-88.e5.

56. Jones MK, Watanabe M, Zhu S, Graves CL, Keyes LR, Grau KR, et al. Enteric bacteria promote human and mouse norovirus infection of B cells. Science. 2014 Nov 7;346(6210):755–9.

57. Makimaa H, Ingle H, Baldridge MT. Enteric Viral Co-Infections: Pathogenesis and Perspective. Viruses. 2020 Aug 18;12(8):904.

58. Chen YH, Du W, Hagemeijer MC, Takvorian PM, Pau C, Cali A, et al. Phosphatidylserine Vesicles Enable Efficient En Bloc Transmission of Enteroviruses. Cell. 2015 Feb;160(4):619–30.

59. Dreux M, Garaigorta U, Boyd B, Décembre E, Chung J, Whitten-Bauer C, et al. Short-Range Exosomal Transfer of Viral RNA from Infected Cells to Plasmacytoid Dendritic Cells Triggers Innate Immunity. Cell Host Microbe. 2012 Oct;12(4):558–70.

60. Cosset FL, Dreux M. HCV transmission by hepatic exosomes establishes a productive infection. J Hepatol. 2014 Mar 1;60(3):674–5.

61. Pérez-Losada M, Arenas M, Galán JC, Palero F, González-Candelas F. Recombination in viruses: Mechanisms, methods of study, and evolutionary consequences. Infect Genet Evol. 2015 Mar;30:296–307.

62. Yao XD, Evans DH. Effects of DNA structure and homology length on vaccinia virus recombination. J Virol. 2001 Aug;75(15):6923–32.

63. Patiño-Galindo JÁ, Filip I, Rabadan R. Global Patterns of Recombination across Human Viruses. Mol Biol Evol. 2021 Jun 1;38(6):2520–31.

64. Li C, Wang H, Shi J, Yang D, Zhou G, Chang J, et al. Senecavirus-Specific Recombination Assays Reveal the Intimate Link between Polymerase Fidelity and RNA Recombination. J Virol. 2019 Jun 14;93(13):10.1128/jvi.00576-19.

65. Roberts JD, Bebenek K, Kunkel TA. The Accuracy of Reverse Transcriptase from HIV-1. Science. 1988 Nov 25;242(4882):1171–3.

66. Lemey P, Salemi M, Vandamme AM, editors. The Phylogenetic Handbook: A Practical Approach to Phylogenetic Analysis and Hypothesis Testing [Internet]. 2nd ed. Cambridge: Cambridge University Press; 2009. Available from: https://www.cambridge.org/core/books/phylogenetic-handbook/A9D63A454E76A5EBCCF1119B3C56D766

67. Gibbs MJ, Weiller GF. Evidence that a plant virus switched hosts to infect a vertebrate and then recombined with a vertebrate-infecting virus. Proc Natl Acad Sci. 1999 Jul 6;96(14):8022–7.

68. Suarez DL, Senne DA, Banks J, Brown IH, Essen SC, Lee CW, et al. Recombination Resulting in Virulence Shift in Avian Influenza Outbreak, Chile. Emerg Infect Dis. 2004 Apr;10(4):693–9.

69. Sievers F, Wilm A, Dineen D, Gibson TJ, Karplus K, Li W, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. Mol Syst Biol. 2011 Jan;7(1):539.

70. Katoh K, Standley DM. MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. Mol Biol Evol. 2013 Apr 1;30(4):772–80.

71. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 2004 Mar 1;32(5):1792–7.

72. Martin DP, Varsani A, Roumagnac P, Botha G, Maslamoney S, Schwab T, et al. RDP5: a computer program for analyzing recombination in, and removing signals of recombination from, nucleotide sequence datasets. Virus Evol. 2021 Jan 20;7(1):veaa087.

73. SA S. GENECONV: a computer package for the statistical detection of gene conversion. Httpwww Math Wustl Edu Sawyer. 1999;

74. Statistical tests for detecting gene conversion. Mol Biol Evol [Internet]. 1989 Sep [cited 2024 May 20]; Available from: https://academic.oup.com/mbe/article/6/5/526/1088566/Statistical-tests-for-detecting-gene-conversion

75. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990 Oct 5;215(3):403–10.

76. Smith J. Analyzing the mosaic structure of genes. J Mol Evol [Internet]. 1992 Feb [cited 2024 May 23];34(2). Available from: http://link.springer.com/10.1007/BF00182389

77. Posada D, Crandall KA. Evaluation of methods for detecting recombination from DNA sequences: Computer simulations. Proc Natl Acad Sci. 2001 Nov 20;98(24):13757–62.

78. Gibbs MJ, Armstrong JS, Gibbs AJ. Sister-Scanning: a Monte Carlo procedure for assessing signals in recombinant sequences. Bioinformatics. 2000 Jul 1;16(7):573–82.

79. SALMINEN MO, CARR JK, BURKE DS, McCUTCHAN FE. Identification of Breakpoints in Intergenotypic Recombinants of HIV Type 1 by Bootscanning. AIDS Res Hum Retroviruses. 1995 Nov;11(11):1423–5.

80. Martin DP, Williamson C, Posada D. RDP2: recombination detection and analysis from sequence alignments. Bioinformatics. 2005 Jan 15;21(2):260–2.

81. Boni MF, Posada D, Feldman MW. An Exact Nonparametric Method for Inferring Mosaic Structure in Sequence Triplets. Genetics. 2007 Jun 1;176(2):1035–47.

82. Weiller GF. Phylogenetic profiles: a graphical method for detecting genetic recombinations in homologous sequences. Mol Biol Evol. 1998;15(3):326–35.

83. Beiko RG, Hamilton N. Phylogenetic identification of lateral genetic transfer events. BMC Evol Biol. 2006 Dec;6(1):15.

84. Lemey P, Lott M, Martin DP, Moulton V. Identifying recombinants in human and primate immunodeficiency virus sequence alignments using quartet scanning. BMC Bioinformatics. 2009 Dec;10(1):126.

85. Jaya FR, Brito BP, Darling AE. Evaluation of recombination detection methods for viral sequencing. Virus Evol. 2023 Jan 1;9(2):vead066.

86. Shikov AE, Malovichko YV, Nizhnikov AA, Antonets KS. Current Methods for Recombination Detection in Bacteria. Int J Mol Sci. 2022 Jan;23(11):6257.

87. Bay RA, Bielawski JP. Recombination Detection Under Evolutionary Scenarios Relevant to Functional Divergence. J Mol Evol. 2011 Dec;73(5–6):273–86.

88. Jariani A, Warth C, Deforche K, Libin P, Drummond AJ, Rambaut A, et al. SANTA-SIM: simulating viral sequence evolution dynamics under selection and recombination. Virus Evol [Internet]. 2019 Jan 1 [cited 2024 May 23];5(1). Available from: https://dx.doi.org/10.1093/ve/vez003

89. Brown T, Didelot X, Wilson DJ, Maio ND. SimBac: simulation of whole bacterial genomes with homologous recombination. Microb Genomics. 2016 Jan 19;2(1):e000044.

90. Sipola A, Marttinen P, Corander J. Bacmeta: simulator for genomic evolution in bacterial metapopulations. Bioinformatics. 2018 Jul 1;34(13):2308–10.

91. Bobay LM. CoreSimul: a forward-in-time simulator of genome evolution for prokaryotes modeling homologous recombination. BMC Bioinformatics. 2020 Jun 24;21(1):264.

92. Trost J, Haag J, Höhler D, Jacob L, Stamatakis A, Boussau B. Simulations of Sequence Evolution: How (Un)realistic They Are and Why. Crandall K, editor. Mol Biol Evol. 2024 Jan 3;41(1):msad277.

93. Cox DR. The Regression Analysis of Binary Sequences. J R Stat Soc Ser B Stat Methodol. 1959 Jan 1;21(1):238–238.

94. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, et al., editors. Advances in Neural Information Processing Systems [Internet]. Curran Associates, Inc.; 2017. Available from:

https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

95. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychol Rev. 1958;65(6):386–408.

96. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. J Mach Learn Res. 2011;12:2825–30.

97. TensorFlow Developers. TensorFlow [Internet]. Zenodo; 2023 [cited 2023 Jun 20]. Available from: https://zenodo.org/record/4724125

98. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library [Internet]. arXiv; 2019 [cited 2025 Jan 24]. Available from: https://arxiv.org/abs/1912.01703

99. Posada D. Evaluation of Methods for Detecting Recombination from DNA Sequences: Empirical Data. Mol Biol Evol. 2002 May 1;19(5):708–17.

100. Swanepoel P. Simulating recombinant sequence data to evaluate and improve computation methods of multiple sequence alignment and recombinant identification [Internet]. [Division of Computational Biology]: University of Cape Town; Available from: http://hdl.handle.net/11427/39870

101. Schmidt M, Le Roux N, Bach F. Minimizing finite sums with the stochastic average gradient. Math Program. 2017 Mar;162(1–2):83–112.

102. Breiman L. Random forests. Mach Learn. 2001;45:5–32.

103. Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal Loss for Dense Object Detection [Internet]. arXiv; 2018 [cited 2025 Jan 24]. Available from: http://arxiv.org/abs/1708.02002

104. Loshchilov I, Hutter F. Decoupled Weight Decay Regularization [Internet]. arXiv; 2019 [cited 2025 Jan 24]. Available from: http://arxiv.org/abs/1711.05101

Appendix

Explanation of Recombinant Event Tracking in SANTA-SIM

The recombination tracking system in the forked version of SANTA-SIM works through several interconnected components:

1. RecombinantTracker (Singleton) provides global access to:
    1.1. A counter for total recombination events
    1.2. A centralized list of all RecombinationEvent objects
2. RecombinationEvent function stores complete information about each recombination even including:
    2.1. Parent genomes that recombined
    2.2. The resulting recombinant genome
    2.3. Generation when recombination occurred
    2.4. Breakpoints (positions where genetic material switches between parents)
    2.5. Original sequences from both parents and the recombinant
3. Virus Class Extensions track recombination history:
    3.1. Each Virus maintains a LinkedHashSet of indices pointing to recombination events.
    3.2. When recombination occurs, the new virus inherits recombination history from both parents.
    3.3. New recombination events are added to the global tracker and referenced in the virus.
4. Propagation of History ensures tracking across generations:
    4.1. When viruses replicate with recombination, their offspring maintain pointers to parent events.
    4.2. The system carefully updates breakpoint positions when indels (insertions/deletions) occur.
    4.3. Recombination only occurs at codon boundaries to preserve reading frames.

This tracking system enables the alignment feature to correctly represent genetic relationships between sequences, maintaining a complete history of recombination events through the simulation.

Creating Perfect Alignments in SANTA-SIM

The process of creating perfect genomic alignments in this forked version of SANTA-SIM is sophisticated and handles recombination, insertions, and deletions through a multi-step process:

1. Indel Tracking During Simulation:
   1.1. Each genome maintains a chronological list of all insertion/deletion events
   1.2. Each indel records: position, size (positive for insertions, negative for deletions), coordinate translation, and a unique ID
   1.3. During recombination, indel histories from both parents are merged based on breakpoint positions
2. Alignment Construction Process:
   2.1. The writeFastaFormat method in AlignmentSampler.java implements the alignment algorithm
   2.2. For each sampled virus:
      2.2.1. Indels are processed in reverse chronological order using remove_indels
      2.2.2. Deletions are replaced with gap characters
      2.2.3. Insertions are removed but stored in an Insert_Event collection for later reintroduction
3. Insertion Management:
   3.1. The Insert_Event class tracks:
      3.1.1. Original position of each insertion
      3.1.2. Inserted sequence
      3.1.3. Which viruses contain this insertion
      3.1.4. Size of the insertion (may vary between viruses)
      3.1.5. Insertions at the same genomic location across viruses are grouped together
      3.1.6. This handles cases where recombination creates complex insertion patterns
4. Reintroduction of Insertions:
   4.1. The add_insertions method reinserts all stored insertions into the sequences
   4.2. Insertions are added in order from start to end of the genome
   4.3. All sequences receive either:
      4.3.1. The original inserted sequence (if the virus had that insertion)

        4.3.2. Gap characters of the same length (if the virus didn't have that insertion). This ensures all sequences maintain the same length and proper homology

5. Gap-Only Column Removal:
    5.1. After all insertions are reintroduced, columns containing only gaps are removed
    5.2. This produces cleaner alignments without affecting sequence homology
6. Recombination History Records:
    6.1. The system also outputs auxiliary files tracking recombination:
    6.2. recombination_events_*.txt - Details each recombination event and its breakpoints
    6.3. sequence_events_map_*.txt - Links each sequence to its recombination history

This approach ensures that even with complex recombination patterns and nested indel events, the resulting alignments maintain perfect positional homology between all sequences, allowing for accurate phylogenetic and evolutionary analyses.

Example of a Sequence Events Map File

Taken from XML1-2500-0.01-4E-5-50-1.txt

Sequence*Events
1*[19444, 13657, 15059, 18091, 19950, 20901]
2*[17465]
3*[10929]
4*[10929]
5*[]
6*[]
7*[20216, 20819]
8*[10929]
9*[]
10*[]
11*[10929, 19693]
12*[]
13*[10929, 19693]
14*[]
15*[]
16*[]
17*[15104, 13584, 20376]
18*[]
19*[2227, 1551, 2324, 13446, 15930]
20*[]
21*[]
22*[]
23*[10929, 19693]
24*[10929]
25*[2227, 1551, 2324, 5972, 10989, 12783, 17938, 19957]
26*[]
27*[10929, 20087]
28*[13657, 15059]
29*[]

Continued in file.

Example of Recombination Events File

Taken from Recombination_events_XML1-2500-0.01-4E-5-50-1.txt

EventNum * Breakpoints * Generation * Recombinant * Parents

1551 * [0, 747] * 229 * santa.simulator.genomes.SimpleGenome@971d0d8 * [santa.simulator.genomes.SimpleGenome@51931956, santa.simulator.genomes.SimpleGenome@2b4a2ec7]

2227 * [1095, 3189] * 304 * santa.simulator.genomes.SimpleGenome@564718df * [santa.simulator.genomes.SimpleGenome@51b7e5df, santa.simulator.genomes.SimpleGenome@18a70f16]

Continued in file.

Example of SimVsRealCompare File

Taken from alignment_XML1-2500-0.01-4e-5-50-1.faSimVsRealCompare.csv

```
RDPEvent,ActualRecomb,PredictRecomb,MatchNo,Matchscore,SimEVentNo,SimBP
Start,SimBPEnd, PredBPStart, PredBPEnd
 1, 27, 27, 7, .9968553, 20087, 1269, 3495, 3496, 1276
 3, 43, 43, 15, .9903978, 19693, 1596, 2325, 1601, 2322
 6, 2, 2, 21, .9492862, 17465, 510, 1530, 509, 1479
 8, 17, 42, 5, .9936392, 20376, 2085, 3300, 2090, 3308
 11, 48, 48, 17, .9850793, 19248, 2340, 3012, 2296, 3014
 13, 48, 28, 4, .8965849, 20691, 0, 1743, 3798, 1933
 14, 1, 1, 2, .8993711, 20901, 3663, 3822, 3678, 3820
 15, 34, 34, 1, .8037634, 21246, 3450, 3822, 3485, 3783
```

Other Models Tested

For the sake of reproducibility the other models trialled but not included in the write up are briefly described here with their relative performance metrics.

Naive Gaussian Bayes

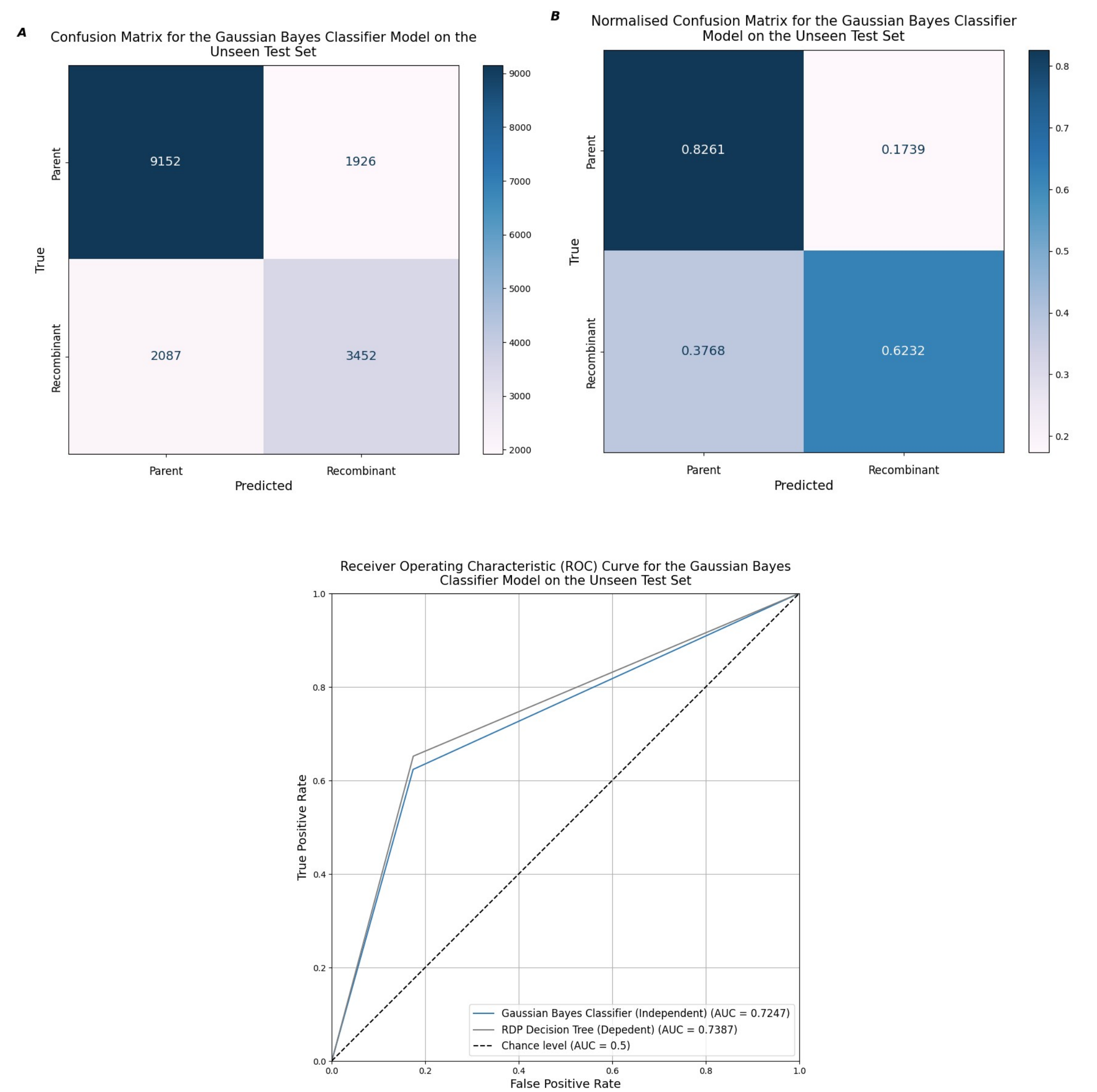


Figure 3.1: *showing the performance of the Naive Gaussian Bayes on the test set. This figure shows the confusion matrices for the model and the ROC curve plotted against RDP5's decision tree for comparison.*
A: *showing the confusion matrix for the classifieron the test data.*
B: *showing the normalised confusion matrix for theclassifier on the test data.*
C: *showing the receiver operating characteristic curvefor the classifier on the test dataset, with an AUC of 0.7247.*

The key hyperparameters for this model were kept as the defaults suggested by the SKLearn team. NGBs have few hyperparameters to tune. The key parameters were variance smoothing, set to 1e-9, and the Priors variable set to None, resulting in the priors being calculated from the training data for each class. The performance of the model was inferior to other models as shown in figure 3.1. Hence, this model was not included in the main write up.

K-Nearest Neighbours

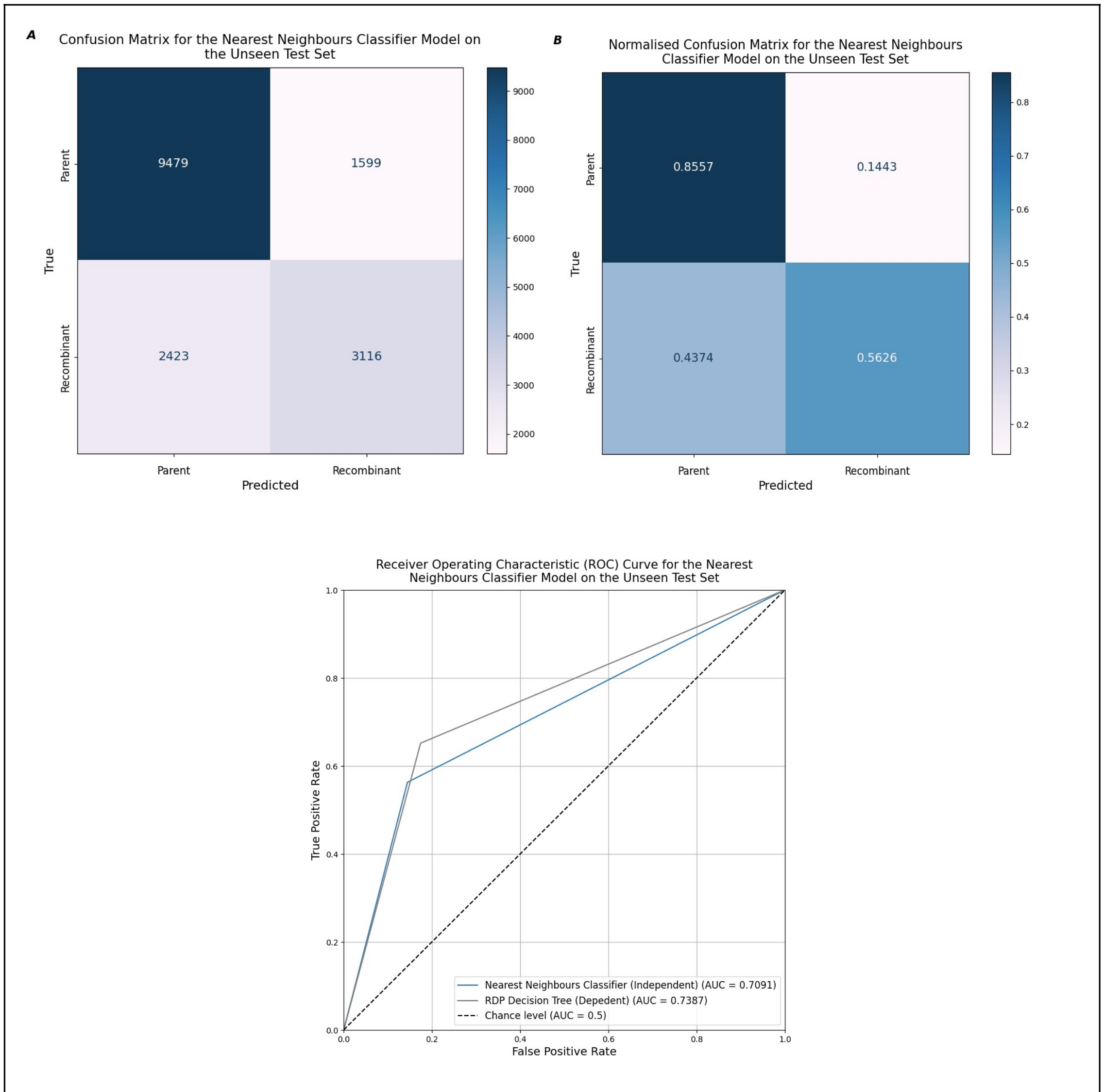


Figure 3.1: *showing the performance of the Nearest Neighbours classifiers on the test set. This figure shows the confusion matrices for the model and the ROC curve plotted against RDP5's decision tree for comparison.*
A: *showing the confusion matrix for the classifieron the test data.*
B: *showing the normalised confusion matrix for theclassifier on the test data.*
C: *showing the receiver operating characteristic curvefor the classifier on the test dataset, with an AUC of 0.7091.*

The key hyperparameters for this model were kept as the defaults suggested by the SKLearn team. The key parameters for this model include the number of neighbours which was kept as the default of 5, the weights parameter which was kept as uniform, the leaf size kept as 30, and the algorithm was set as ‘auto’. This model also underperformed to other models in the main paper which is why it was not included in the main write up.