Studying a Virus Assembly Fitness Landscape with Data Science

Rohini Krishna Preetha
201669138

Supervised by Pierre-Philippe Dechant

Submitted in accordance with the requirements for the
module MATH5872M: Dissertation in Data Science and Analytics
as part of the degree of

Master of Science in Data Science and Analytics

The University of Leeds, School of Mathematics

September 2023

The candidate confirms that the work submitted is his/her own and that appropriate credit has been given where reference has been made to the work of others.

UNIVERSITY OF LEEDS

School of Mathematics

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

Academic integrity statement

I am aware that the University defines plagiarism as presenting someone else's work, in whole or in part, as your own. Work means any intellectual output, and typically includes text, data, images, sound or performance.

I promise that in the attached submission I have not presented anyone else's work, in whole or in part, as my own and I have not colluded with others in the preparation of this work. Where I have taken advantage of the work of others, I have given full acknowledgement. I have not resubmitted my own work or part thereof without specific written permission to do so from the University staff concerned when any of this work has been or is being submitted for marks or credits even if in a different module or for a different qualification or completed prior to entry to the University. I have read and understood the University's published rules on plagiarism and also any more detailed rules specified at School or module level. I know that if I commit plagiarism I can be expelled from the University and that it is my responsibility to be aware of the University's regulations on plagiarism and their importance.

I re-confirm my consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to monitor breaches of regulations, to verify whether my work contains plagiarised material, and for quality assurance purposes. I confirm that I have declared all mitigating circumstances that may be relevant to the assessment of this piece of work and that I wish to have taken into account. I am aware of the University's policy on mitigation and the School's procedures for the submission of statements and evidence of mitigation. I am aware of the penalties imposed for the late submission of coursework.

Name Rohini Krishna Preetha

Student ID 201669138

Abstract

Treatment and comprehension of viral infections depend heavily on the knowledge of viral assembly. Using genomic data analysis and machine learning, this work explores the fitness landscape of viral assembly. The data set was taken from (Twarock et al., 2018). It reveals distinctive packaging signal patterns using PCA and UMAP. Strong affinities of the packaging signals at the ends of the genomes provide high fitness, while low binding affinities provide low fitness. In contrast, the strength of the binding affinities in the middle parts of the genomes exhibits an inverse relationship with the fitness values. The work comprises three models: a Random Forest Model, a two-layered Recurrent Neural Network and a six-layered Recurrent Neural Network. The six-layered Recurrent Neural Network emerges as the best model among the three with a high R^2 value (0.9904), low MAE (0.0057) and RMSE (0.01) values. The model has a good $\mathcal{F}$-statistic of 116.8454 and a low p-value of $3.1374 \cdot 10^{-27}$ showing that the model accurately learns the patterns present in the data. The analysis of the feature importance of the model using SHAP values reveals that the packaging signals in the middle part of the genomes have more importance than the ones at the ends. Moreover, the analysis using the SHAP values agrees with the findings obtained using UMAP. Notably, this study assists in the decoding of assembly dynamics. This study provides insights into viral assembly and illuminates key elements affecting fitness. Furthermore, it lays the groundwork for future initiatives that support the UN Sustainable Development Goals, notably in increasing global health and well-being through a better understanding of virus assembly and control methods.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human health and agricultural systems are both threatened by viral diseases. The presence of viral genetic material in the human genome is the result of a lengthy history of co-evolution. The persistent threat of viral epidemics, as brought out by COVID-19, Zika and Ebola outbreaks, as well as the introduction of new zoonotic illnesses, highlights the critical need for novel antiviral therapies. However, due to the enormous diversity of viruses and their capacity to rapidly change, medicinal therapies face severe challenges. During infections, RNA viruses in particular exhibit rapid mutation rates and exist as related variants known as quasispecies. As a result, therapeutic and preventative options remain restricted. Emerging developments notably in comprehending how certain viruses with single-stranded, positive-sense RNA genomes assemble, have begun to fill this knowledge gap (Twarock and Stockley, 2019).

A completely developed infectious virus is called a virion. In their simplest form, virions are made up of two parts: genetic material (single- or double-stranded RNA or DNA) and a protective protein shell known as the capsid. The capsid functions as a shell, protecting the viral DNA from nucleases, and it attaches the virion to particular receptors on the surface of prospective host cells during infection. The viral genome contains the instructions for the capsid proteins and encodes a small number of structural proteins due to its small size (Gelderblom, 1996).

Capsids are made up of one or a few unique structural protein types and are made up of either a single or double protein shell. As a result, many copies of these proteins must self-assemble to form the capsid's continuous three-dimensional structure. Virus capsids self-assemble in two ways: helical symmetry, in which protein subunits and nucleic acids are organised in a helical pattern, and icosahedral symmetry, in which protein subunits come together to build a symmetric shell wrapping the core comprising the viral genetic material. Viruses having helical symmetry are distinguished by their elongated tube-like form with capsomers, which are the building segments of the capsid, organised helically around the coiled RNA [e.g.: Tobacco Mosaic Virus]. Whereas viruses featuring icosahedral symmetry have a spherical structure consisting of 20 triangular faces and 12 vertices with five, three or two-fold rotational symmetry. Axes of fivefold rotational symmetry are defined by lines through opposing vertices: all structural aspects of the

polyhedron repeat five times inside each 360° rotation about any of the five-fold axes. Axes of threefold rotational symmetry are formed by lines passing through the centres of opposing triangle faces. Lines through the midpoints of opposite edges produce a two-fold rotational symmetry axis. The simplest icosahedral capsid has three capsomers per triangle face, giving the virus a total of 60 capsomers. These capsomers form the fundamental units of the capsid (Gelderblom, 1996).

Most of the viruses have icosahedral symmetry in their capsid containers (Dechant and Twarock, 2021). Moreover, in viruses with single-stranded genomes, encapsidation is frequently aided by packaging signals that correspond to certain secondary structural patterns within the viral RNA (Dechant and He, 2021a). Because of the intrinsic symmetry of the capsid, the packaging signal mechanism functions through the interaction between viral RNA with the coat protein (CP). The packaging signals appear as stem-loop structures within single-stranded RNA, giving a recognisable pattern capable of binding to CP. In some cases, the RNA-CP contact causes a structural alteration in the CP, allowing it to participate in assembly. The stability and binding strength, of these packaging signals, generate a unique profile for viral assembly, thus defining the phenotype important for the assembly process. The fitness of this phenotype can contribute to the overall assembly efficiency of the virus as shown in Figure 1.1A. Recent discoveries suggest that the occurrence of numerous packaging signals distributed throughout the genome is quite probable (Twarock and Stockley, 2019; Dechant and Twarock, 2021). This configuration reduces the search space for CP during capsid assembly, resulting in substantially improved assembly efficiency. As a result, the viral genome is subject to numerous levels of constraints, acting as both a gene repository and the instruction manual for packaging signals. This adaptability might lead to a variety of applications, including the creation of vaccines, medication and novel anti-viral techniques (Patel et al., 2017; Dechant and He, 2021a).

In 1994, a dodecahedron with 12 pentagonal faces was used as the equilibrium model to analyse the assembly of a simplified icosahedral virus by Zlotnick. The model was examined using a framework for ordinary differential equations (ODE) (Zlotnick, 1994).

Twelve packaging signals (PSs) could attach to coat proteins (CPs) in this model as shown in Figure 1.1B and subsequently detach, indicating reversible and equilibrium behaviour. The linked CPs can then bind to more CPs as shown in Figure 1.1C, ultimately building a capsid. These PSs have three binding strengths: weak, medium, and strong. Based on the TR (Translational Repressor) sequence in MS2 (Male Specific 2 bacteriophage), which contains roughly 12 kcal/mol of energy, this translates to energy levels of 4/8/12 kcal/mol. The energy required for CP-to-CP binding is substantially lower, around 2 kcal/mol. This change in binding strength influences assembly speed. It can, for example, be used as a starting point for assembly or to allow for mistake correction via weaker bindings elsewhere. Assembly efficiency is determined by the thermodynamics of PS binding and the quantity of CP bonds produced. This efficiency is thought to be a measure of fitness, or at the very least its contribution as a result of assembly considerations. This model takes into account important features of viral genetics, geometry,

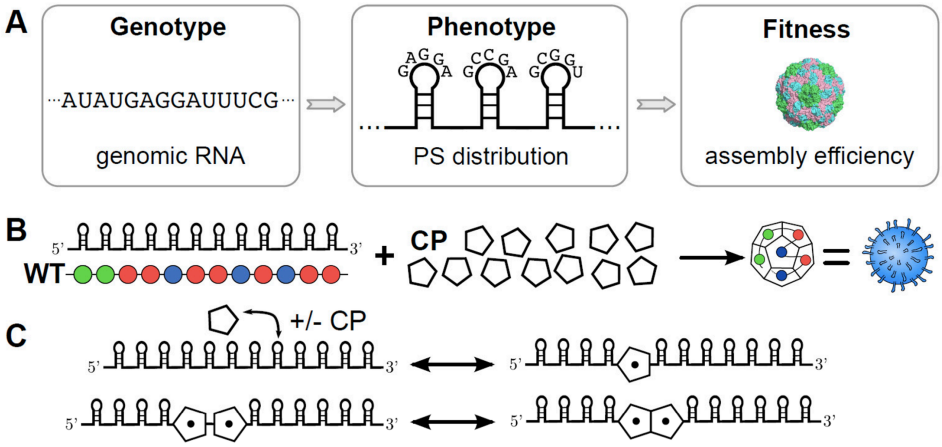

Figure 1.1: A packaging signal-mediated assembly modelling approach. Figure taken from (Dechant and He, 2021b)

and assembly as well as provides a discrete fitness space but requires extensive computations (Twarock et al., 2018; Dechant and He, 2021a).

The ability of neural networks to predict assembly efficiency, using the dataset from (Twarock et al., 2018), which contains the phenotype-fitness map, with exceptional precision was demonstrated in (Dechant and He, 2021a). The work suggested that stochastic simulations could be employed to partly investigate more complicated or realistic genome spaces, estimating assembly fitness to generate a dataset for training a neural network. The remaining aspects of the fitness landscape could then be predicted using neural network models (Dechant and He, 2021a).

This project aims to explore different machine learning techniques to make a more precise model for predicting virus assembly fitness. The work also aims to find out the patterns in features of the genomic data that are used to predict fitness values.

The structure of this report is as follows:

- **Literature Review**: This chapter presents the details about the key findings from several research papers that helped to understand and analyse the data and build the machine learning models.
- **Materials and Methods**: This chapter explains the dataset, the methods used for doing the data preparation, exploratory data analysis, building the machine learning models, and doing further analysis on the models.
- **Results**: The main goal of this chapter is to present the results and key findings from the methods used through plots and charts.
- **Conclusion**: The overall summary of the project work along with the future scope of the work is presented in this chapter.

Chapter 2

Literature Review

This chapter captures studies conducted on the diverse and dynamic landscape of viral capsid assembly and the role of Packaging Signals (PS) to harnessing the power of machine learning and deep learning in understanding and predicting the fitness landscapes. In addition, it explores the domain of dimensionality reduction techniques, with a special emphasis on the effectiveness of UMAP in computational biology. These publications highlight the value of multidisciplinary cooperation, mathematical modelling, and sophisticated data-driven methodologies in expanding our knowledge of complex viruses.

2.1 Twarock et al. (2018)

The research goes into a thorough analysis of how widely distributed Packaging Signal (PS) locations in viruses aid capsid formation. To address this, the authors create a mathematical model that captures the overall impact of PS sites on viral assembly efficiency. They take a geometric approach, concentrating on the dodecahedral capsid model, which is made up of twelve pentagonal capsid building components. This model is based on the architecture of Picornaviruses and numerous tiny plant viruses. The authors divide PS affinities for Coat Protein (CP) into three categories: mild, moderate, and strong.

The Key Findings are:

- **PS-Mediated Assembly Efficiency**: The model shows that PS-mediated assembly efficiency is critical in viral capsid development. It enables finer segmentation of the fitness space, improving computing efficiency while capturing key components of the fitness landscape.
- **Protein Ramp in Viral Assembly**: The study emphasises the importance of the "protein ramp" phenomena seen in viral infections. Instead of injecting the whole CP allotment at the start, the model incorporates a steady build-up of CP concentration. This change corresponds to the in-vivo settings of viral infections, resulting in a more realistic description of the assembly process.

- **PS Distribution Cooperative Effects**: The presence of the protein ramp demonstrates that the cooperative activity of PSs considerably improves assembly efficiency. This cooperative behaviour is best observed in conditions similar to the protein ramp, which may explain why PSs were not considered in in-vitro tests.

- **Addressing Levinthal's Paradox**: The study addresses a viral equivalent of Levinthal's Paradox in protein folding. It demonstrates how the PS distribution navigates the terrain of alternative assembly intermediates efficiently, biasing assembly towards the most efficient paths and reducing the process's overall complexity.

- **Hamiltonian Paths Analysis**: The authors provide Hamiltonian Paths Analysis, a mathematical technique that encodes alternative assembly possibilities based on the sequence of PS binding to CP. This technique correctly predicts PS sites in bacteriophage MS2, indicating its relevance in finding significant elements in viral genome organisation.

The modelling of PS-mediated assembly not only gives important mechanistic insights but also opens up new paths for anti-viral treatment and bionanotechnology applications. Understanding the features and functions of PS dispersion allows for the creation of stable virus-like particles with increased assembly efficiency, highlighting potential applications in sectors such as gene transfer and vaccination techniques.

Overall, this study provides a thorough examination of the PS-mediated assembly process, shedding insight into the complicated processes underpinning viral capsid formation. The mathematical models produced in this work will be useful for future research into viral infections, anti-viral techniques, and bionanotechnology applications.

2.2 Dechant and He (2021a)

In this paper, the authors investigate the use of machine learning, specifically neural networks, to predict the fitness landscape pertaining to viral assembly. The dataset taken from (Twarock et al., 2018) contains a mapping of 12-dimensional vector inputs to numerical output values, making it suitable for machine learning. The neural network is trained on a portion of the genomic space and verified on the remaining data. This method displays neural networks' quick capacity to learn sophisticated patterns within the vast degeneracy of thorough stochastic modelling, predicting assembly efficiency fitness with amazing precision. When compared to typical stochastic modelling techniques, this methodology delivers computational efficiency while capturing key elements of the fitness landscape.

The authors demonstrate that, while certain nuances of stochastic modelling are lost, the computing performance gained is significant. The neural network's flexibility to discontinuities in the stochastic technique allows for a more refined segmentation of fitness space, offering applications in complicated models such as virus-like particles and carbon fullerene assembly in biomedical and nanoscience.

The fundamental challenge includes converting 12-dimensional vector inputs to a normalised real number output between 0 and 1. The computational complexity of this mapping requires substantial processing resources, with individual genome runtimes ranging from 20 minutes to 12 hours. Despite its inherent complexity, supervised machine learning proved excellent in learning and predicting fitness values. Although the authors' selected neural network design was not substantially optimised, it produced impressive results, beating other machine learning methods analysed.

To validate their approach, the authors split the dataset into training and validation sets, with the training set comprising only about 5.6% of the total data. The neural network efficiently learned the dataset, providing near-perfect predictions when tested on the validation set. Furthermore, the authors explored learning curves by varying the training set size, demonstrating the neural network's ability to adapt to data volume. Overall, the results showcased in this research highlight the potential of machine learning, particularly neural networks, in capturing intricate fitness landscapes efficiently.

2.3 Becht et al. (2019)

The authors compared the UMAP algorithm against various recent dimensionality reduction approaches, including t-SNE, FIt-SNE, and scvis. The study aims to offer a thorough knowledge of these algorithms' computational features, ability to distinguish cell populations, reproducibility of embeddings, and ability to retain distances.

The authors timed each approach as it processed data subsamples ranging in size from 20 to 200,000 cells. The findings revealed some intriguing tendencies. UMAP emerged as the quickest method for smaller datasets, whereas scvis consistently performs slower across all dataset sizes. UMAP competes with FIt-SNE and surpasses other approaches in terms of runtime for bigger datasets (100,000 events and beyond).

The authors assessed the algorithms' capacity to differentiate diverse cell populations within two-dimensional embeddings by using random forests to estimate Phenograph cluster labels based on data point locations in the embeddings. In this challenge, non-linear dimensionality reduction approaches such as UMAP, t-SNE, and FIt-SNE beat PCA. UMAP, t-SNE, and FIt-SNE with or without late exaggeration separated cell populations with good accuracy, with random-forest accuracies hovering around 95%.

The authors also investigated distance preservation in embeddings. Distances on a small scale are effectively preserved by all algorithms, while distances on a moderate size are ignored by t-SNE-based approaches. Surprisingly, scvis beats UMAP in this aspect.

In conclusion, UMAP performs similarly to t-SNE in isolating cell subgroups while preserving global distances akin to scvis. UMAP is a moderately fast algorithm, with runtimes just slightly longer than those of FIt-SNE. This quantitative evaluation validates UMAP's resilience by proving its capacity to effectively retain both local and global data structures without

considerable compromise, all while maintaining excellent computing efficiency.

2.4 Sarker (2021)

The author investigates deep learning algorithms for supervised or discriminative learning, concentrating on multi-layer perceptrons (MLPs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). MLPs are a fundamental deep learning architecture that operates as feedforward artificial neural networks (ANN) with input and output layers and hidden layers as the computational engine. The author emphasises the necessity of setting hyperparameters such as the number of hidden layers, neurons, and iterations in model construction, but also the benefits of MLP in real-time or online learning.

The paper mentions that RNNs introduce the idea of "memory" and maintain the potential to influence current inputs and outputs by using knowledge from previous inputs. However, vanishing gradients might affect the learning when dealing with long data sets. As a remedy, the author mentions innovative remedies:

- **Long short-term memory (LSTM)**: The author explains LSTM as an innovative remedy to the vanishing gradient problem, explaining the activities of its memory cells and gates in storing, maintaining, and extracting pertinent information.

- **Bidirectional RNN/LSTM**: The author introduces bidirectional RNNs, which can process data from both the past and future directions at the same time, with Bidirectional LSTM standing out for improving sequence classification.

- **Gated recurrent units (GRUs)**: The author provides GRUs as a streamlined alternative with fewer parameters than LSTM, with equivalent performance and computational efficiency. They emphasise GRU's ability to extract dependencies from large data sets.

The author delves into common RNN modifications such as Long short-term memory (LSTM), bidirectional RNN/LSTM, and gated recurrent units (GRUs). LSTM is a novel solution to the vanishing gradient problem, whereas bidirectional RNNs can process data from both the past and future directions at the same time. GRUs are simplified variations with fewer parameters that provide equivalent performance and computational efficiency.

In essence, the authors offer useful insights into deep learning algorithms used for supervised learning. The paper explains the difference between MLP, CNN, and RNN and provides a comprehensive grasp of the architectures and their applicability across several application areas. This detailed overview assists readers in understanding the advantages and uses of discriminative deep learning algorithms.

2.5 Jing et al. (2022)

This research proposes an ensemble model for forecasting COVID-19 infection rates using geographical and temporal variables. The study is organised into three stages: data collecting, cross-sectional analysis, and the building of a Dynamic Attention Recurrent Neural Network (DA-RNN) for forecasting. The authors emphasise the need for robust feature selection and model interpretability in pandemic predictions. The dataset includes geographical characteristics for 50 U.S. states and the District of Columbia, as well as 17 static features and the infection rate as the goal variable. The authors employ ensemble algorithms and SHAP to completely evaluate feature importance, revealing which characteristics contribute the most to infection rate forecasts. The paper offers SHAP as a technique for assessing feature significance, assisting in finding dominating features while retaining model interpretability.

The study uses machine learning methods such as XGBoost, Random Forest, and LightGBM, as well as SHAP analysis, to identify numerous critical parameters influencing the propagation of COVID-19. In addition, the study employs Dual-stage Attention-based Recurrent Neural Networks (DA-RNNs) for forecasting verified COVID-19 cases in particular U.S. states, surpassing baseline approaches such as Support Vector Regression (SVR) and an encoder-decoder network. The study gathers and analyses daily COVID-19 time-series data, assessing model performance with measures such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). The study uses SHAP analysis to analyse the model output. It computes feature importance using two datasets from distinct time periods throughout the epidemic. The relevance of features fluctuates with time, showing the changing nature of the pandemic's contributing forces.

The paper acknowledges that the forecasting accuracy of the models can be affected by changes in the virus, and future research could consider incorporating additional characteristics of COVID-19 to improve forecasting accuracy.

2.6 Summary

Twarock et al. (2018) provide a mathematical model that investigates the function of Packaging Signals (PS) in viral capsid development, with an emphasis on the dodecahedral capsid model observed in Picornaviruses. They emphasise the significance of PS-mediated assembly efficiency, the "protein ramp" phenomena, the cooperative effects of PS distribution, and how PS solves a viral counterpart of Levinthal's Paradox. Dechant and He (2021a) investigate machine learning, primarily neural networks, for forecasting viral assembly fitness landscapes. Becht et al. (2019) analyse dimensionality reduction techniques, with UMAP being a rapid and effective tool for separating cell types based on single-cell RNA sequencing data.

Sarker (2021) presents a detailed review of supervised learning deep learning methods such as multi-layer perceptrons, convolutional neural networks, and recurrent neural networks, LSTM, Bidirectional RNN/LSTM and GRUs. Jing et al. (2022) use an ensemble model that in-

tegrates geographical and temporal characteristics to anticipate COVID-19 infection rates. The research emphasises feature selection using SHAP values and interpretability in pandemic forecasts, and it presents a forecasting Dynamic Attention Recurrent Neural Network (DA-RNN).

Overall, these papers contribute valuable insights to diverse fields, including viral assembly modelling, machine learning for complex landscapes, dimensionality reduction in computational biology, deep learning algorithms for supervised learning, and feature selection. They underscore the significance of mathematical modelling, machine learning, and data-driven approaches in advancing our understanding of complex biological and epidemiological phenomena.

Chapter 3

Materials and Methods

3.1 Dataset

This project uses the data generated from (Twarock et al., 2018). The dataset consists of 3^{12} (531,441) genomes and their assembly efficiencies measured in terms of fitness. Each genome is represented as a 12-dimensional vector which constitutes twelve packaging signals (PS) with varied degrees of binding affinities. The genome consists of combinations of '1', '2' and '3' values. The values '1', '2' and '3' stand for weak, medium and strong binding affinities respectively. The number of well-assembled capsids out of a potential total of 2,000 determines assembly efficiency. In the 12-dimensional genomic space, this efficiency describes a fitness landscape. The fitness values range from 0 to 2000.

3.2 Data Preparation

The original data was in the form of a single column with rows alternating between 'Fitness' data and associated 'Genome' information as shown in Figure 3.1. Therefore this column was sliced to form separate columns representing pairs of related information. The index was reset to ensure that the indices started from 0. Thereafter the two separate columns were concatenated to form a single dataframe and renamed the columns to 'Genome' and 'Fitness'. Each of the entries in the 'Genome' column is considered a space-separated string of numbers. These strings are broken down into individual digits and then transformed into integers. Finally, the 'Fitness' column was converted into numeric from string datatype, as shown in Figure 3.2. The 'Genome' column was then turned into an array for machine learning purposes as shown in Figure 3.3.

```
                0
0  111111 111111
1             200
2  111111 111112
3            1393
4  111111 111113
5            1869
6  111111 111121
7            1597
8  111111 111122
9            1896
```

Figure 3.1: Structure of the raw data

	Genome	Fitness
0	111111111111	200
1	111111111112	1393
2	111111111113	1869
3	111111111121	1597
4	111111111122	1896
5	111111111123	1960
6	111111111131	1875
7	111111111132	1959
8	111111111133	1961
9	111111111211	1639

Figure 3.2: Structure of the data after modification

```
array([[1, 1, 1, ..., 1, 1, 1],
       [1, 1, 1, ..., 1, 1, 2],
       [1, 1, 1, ..., 1, 1, 3],
       ...,
       [3, 3, 3, ..., 3, 3, 1],
       [3, 3, 3, ..., 3, 3, 2],
       [3, 3, 3, ..., 3, 3, 3]])
```

Figure 3.3: Genome data after converting to array

3.3 Exploratory Data Analysis

To understand the patterns present in the data, exploratory data analysis was carried out using histogram, and boxplot. Furthermore, due to the high dimensionality exhibited by the data

Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP) were also done to visualise the data.

Figure 3.4 depicts the distribution of the 'Fitness' data using a histogram. It follows a normal distribution, although with a tiny positive skew. Notably, the majority of data points fall between 1500 and 1750.

A closer look at the data with a box plot, as shown in Figure 3.5, confirms the presence of a modest positive skew. The bottom quartile (25th percentile) of the data is roughly 1520, whereas the upper quartile (75th percentile) is around 1730. The dataset's lowest value is approximately 1230, while the largest value is around 1980. It is worth noting that the dataset contains an outlier, detected with a value of 200. This outlier deviates from the normal data distribution.

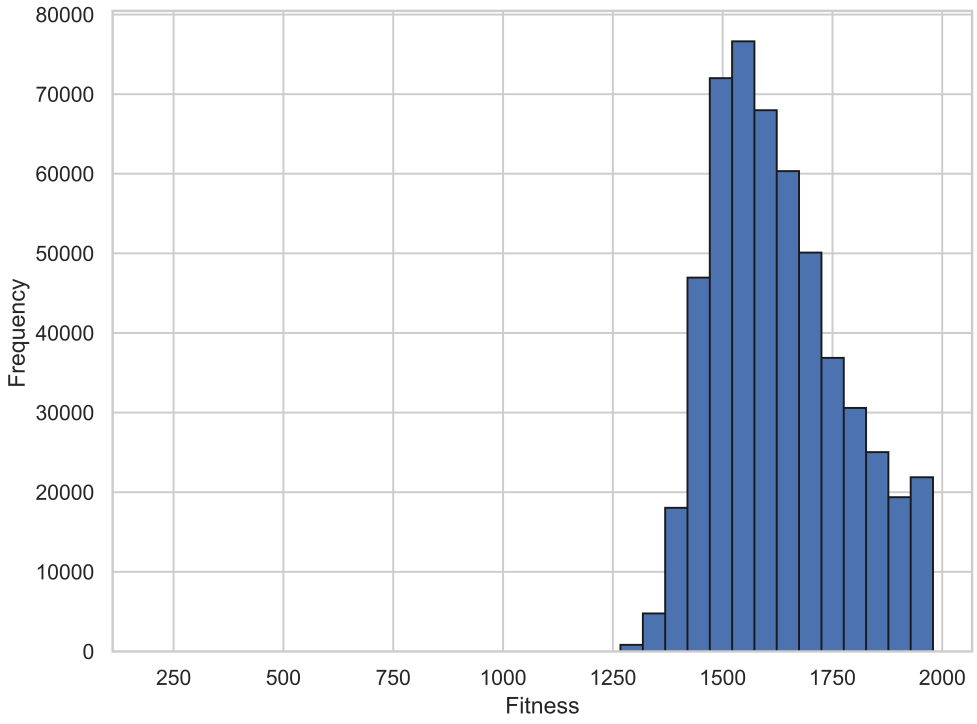

Figure 3.4: Distribution of Fitness data

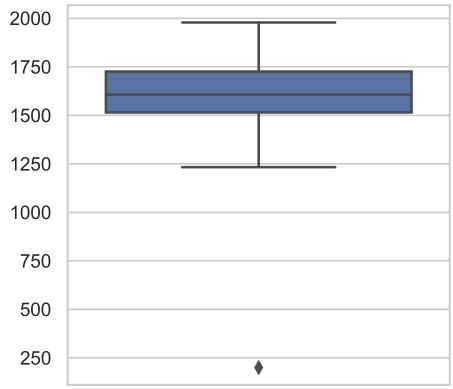

Figure 3.5: Boxplot of Fitness data

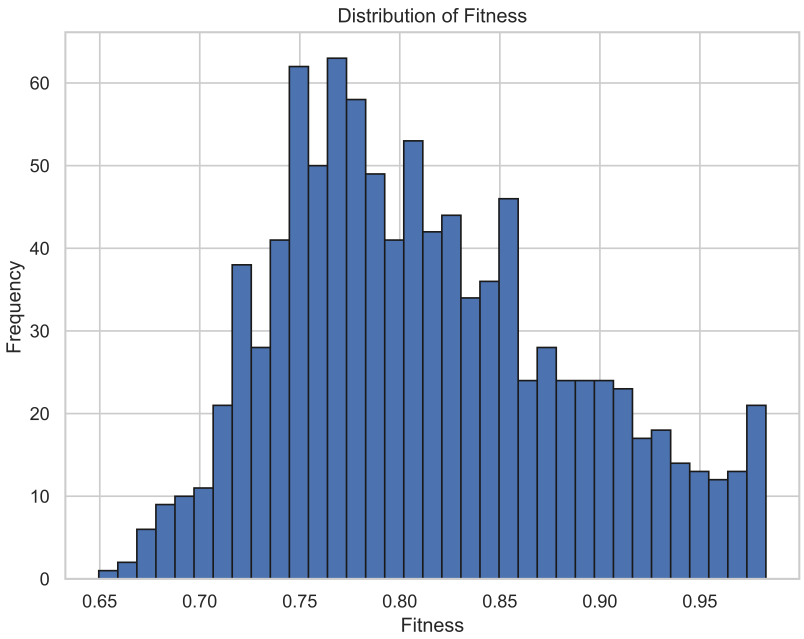

Figure 3.6: Histogram showing the distribution of the subset data

Since there are 531,441 records in the dataset, it is really difficult to visualise the data and get meaningful insights from the data. Therefore 1000 records were picked at random from the data in such a manner that the subset represents the entire range of fitness values and mostly resembles the actual distribution of the data. Moreover, the data was normalised as discussed in section 3.4 before performing PCA and UMAP analysis. The distribution of the subset is shown in Figure 3.6.

3.3.1 PCA

Principal Component Analysis (PCA) is commonly regarded as one of the most famous dimensionality reduction approaches. While it may be interpreted from multiple viewpoints, its core notion is a linear optimisation process that removes the axes with the lowest variance. This optimisation is often represented as the minimising of the axes defined by a vector w^*, as provided by the equation:

$$w^* = \underset{w^T w=1}{\operatorname{argmin}} w^T C w$$

(Draganov, 2021). C is the covariance matrix of the high-dimensional data points X, derived as the data's outer product. This PCA formulation is based on optimising the low-dimensional representation using distance metrics between the high-dimensional and low-dimensional distributions. In essence, the technique seeks to keep the dimensions with the largest variance, maximising the sum of squared distances of data points from the origin. PCA efficiently searches low-dimensional locations whose dot products, as measured by the sum of squared differences, are similar to those in high-dimensional space, thus creating the Principal Components (PCs) which are eigenvectors or the orthogonal axes along which the variances of the data are the highest (Draganov, 2021).

The PCA results give a set of eigenvalues. The amount of variance explained by each 'Principal Component' (PC) is quantified by these eigenvalues. They are normally organised in the order of decreasing magnitude, with the first eigenvalue indicating the highest variance and corresponds to the first PC. To compute the proportion of variance or the explained variance for each PC, the eigenvalue of each component is divided by the sum of all eigenvalues (Jaadi, 2021). The variance that can be explained by each PC can be easily identified using a scree plot. The number of components is decided at the point beyond which the rest of the eigenvalues are all quite low and of similar magnitude (Kassambara and Mundt, 2017) . In other words, if the number of PCs used can cumulatively explain atleast a majority proportion of the variance, then the dimensionality reduction is successful since it can capture most of the relevant information in the data (Schmalen, 2020).

PCA was carried out by utilising three main components to visualise the data. The distribution of all data points throughout the plot is uniform, as seen in Figure 3.7. It seems that none of the components fully expresses the fundamental structure or direction of the data. This might be because the capsid has a dodecahedral structure with a uniform arrangement of 12-dimensional data points (packaging signals). This non-linearity of the data may be the cause of this result, which suggests that more sophisticated methods may be required for a more accurate portrayal (Draganov, 2021).

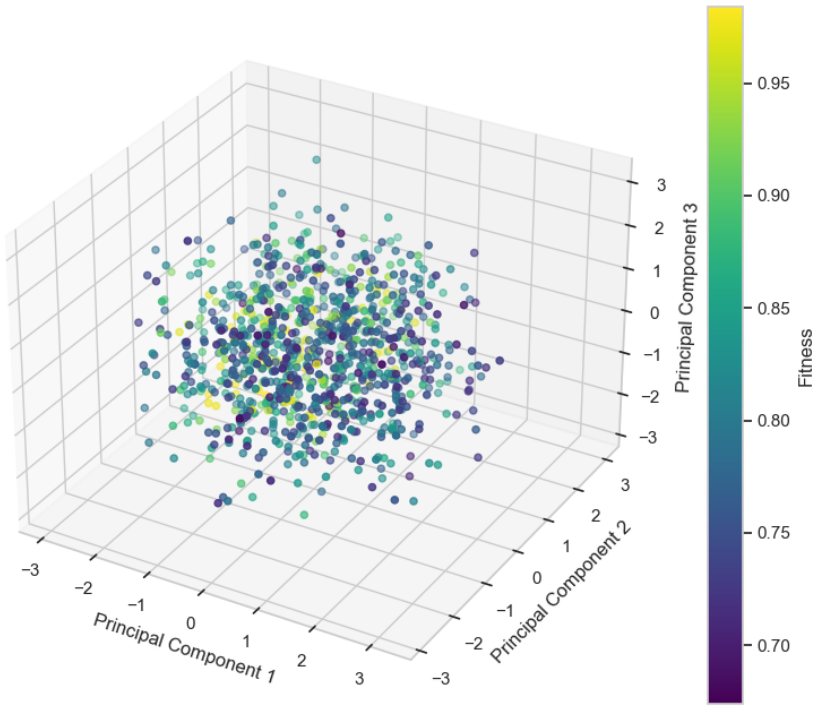

Figure 3.7: PCA plot for subset of data

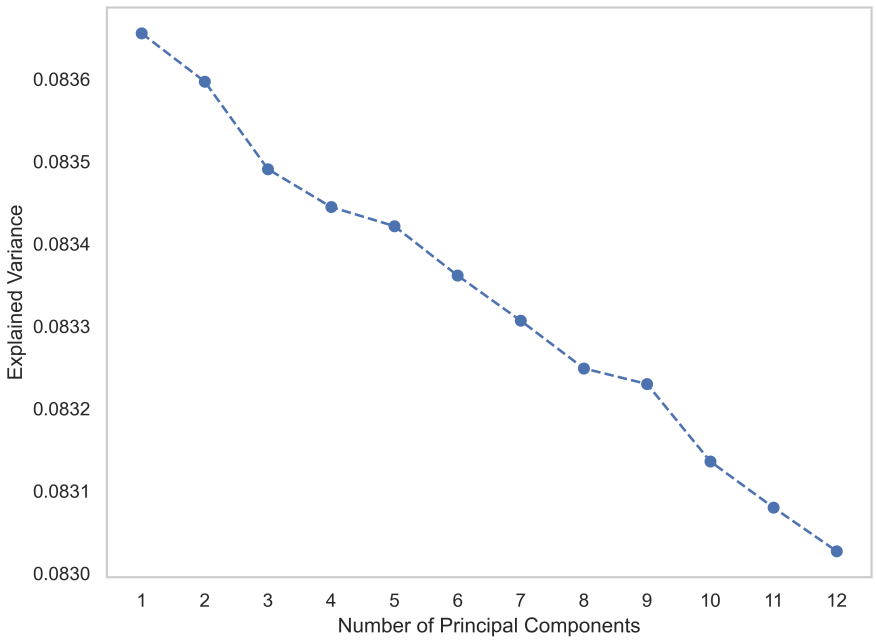

Figure 3.8: Scree plot showing the explained variance for different number of Principal Components

A scree plot (Kanyongo, 2005) was produced to see whether the predetermined number of components could explain the variance in the data as shown in Figure 3.8. The figure shows

that only about 0.0835 of the variance can be explained even with three components. Therefore, employing only three components may not adequately account for the data's variance, indicating that all 12 components may be necessary for a more accurate representation.

3.3.2 UMAP

UMAP (Uniform Manifold Approximation and Projection) is a method that minimises pairwise similarities via gradient descent. UMAP takes a more theoretical approach, attempting to develop a manifold-based representation of both high and low-dimensional spaces. The foundation of UMAP's method is the definition of the pseudo-distance metric $D(x_i, x_j) = d(x_i, x_j)(x_i) - p(x_i)$, where d represents a regular distance metric and is the shortest distance to any other point. This pseudo-metric solves issues related with Euclidean closest neighbours in high-dimensional spaces. UMAP produces a uniform distribution over the space by using this pseudo-metric, allowing the use of topological tools to define the reduction criterion (Draganov, 2021).

UMAP constructs $v_{j|i}$ and v_{ij} using the equations:

$$v_{j|i} = \exp\left[\left(-d\left(x_i, x_j\right) - \rho_i\right)/\sigma_i\right]; v_{ij} = v_{j|i} + v_{i|j} - v_{j|i}v_{i|j},$$

which represent the likelihood of nearest-neighbour associations in high-dimensional space. Furthermore, as seen in the equation:

$$w_{ij} = \left(1 + a\left\|y_i - y_j\right\|_2^{2b}\right)^{-1},$$

UMAP introduces w_{ij} as the low-dimensional counterpart. UMAP interprets each nearest-neighbor connection as a separate Bernoulli random variable. This option guarantees that the perspective has no effect on the edge values. As a result, UMAP eliminates the necessity for normalisation terms over the whole dataset, instead concentrating on preserving the independence of probabilities for each edge (Draganov, 2021).

UMAP was used to get insights into the genetic landscape and the way fitness values are distributed. To make the visualisation easier to understand, the fitness ratings were divided into two groups: 'Fitness>=1800', which represents genomes with high fitness close to 2000 and 'Fitness<1800', which represents genomes with average fitness. The data points in the UMAP graphic were effectively colour-coded by using this categorisation. This method acts as a visual aid, enabling the simple differentiation of genomes with varying fitness levels. The resultant UMAP visualisation, shown in Figure 3.9, gives a picture of the genomic landscape, showing patterns and clusters linked with fitness values.

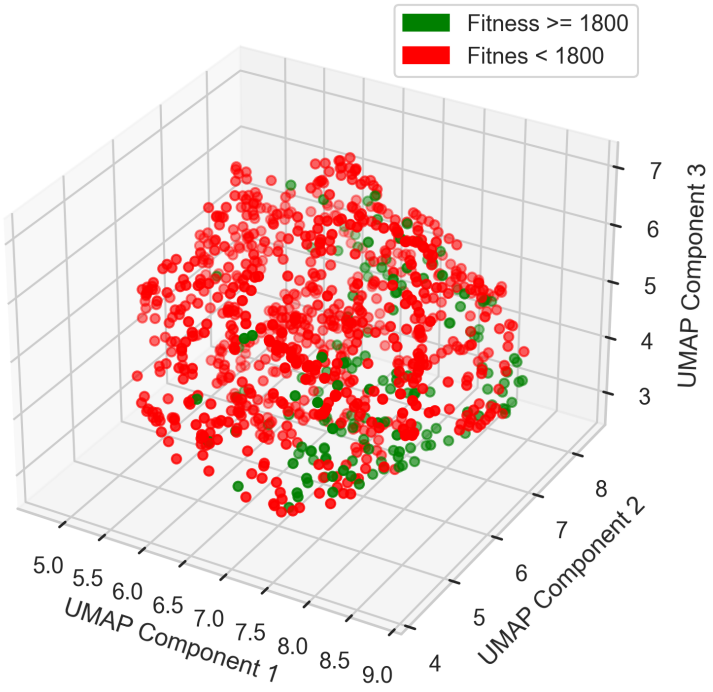

Figure 3.9: UMAP showing the patterns in the fitness landscape. The green points represent high fitness (fitness>=1800) and the red points indicate low fitness (fitness<1800)

The UMAP visualisation shows that genomes with high fitness are more likely to cluster together on one side, indicating that they are closer to each other than genomes with lower fitness scores. This implies that genomes with comparable high fitness values share a degree of similarity, as seen by their spatial grouping in the UMAP plot.

Since this is an interesting aspect, more analysis was conducted to find the pattern in the packaging signals that contributed to high and average fitness values. Tables 3.1 and 3.2 give an overview of genomic data, exhibiting information on packaging signals within genomes linked with low and high fitness values, respectively. These values correspond to 1000 records. Out of 1000 records, 840 genomes have low fitness values, and 160 have high fitness values. The minimum and maximum values for all the packaging signals are 1 and 3 respectively. The tables contain 25^{th} Percentile, 50^{th} Percentile and 75^{th} Percentile values of each packaging signal. Consider PS1 in Table 3.1 as an example. The '25^{th} Percentile', specified as '25%' and equal to 1.0000, indicates that a quarter (25%) of the data points corresponding to PS1 had values equal to or less than 1.0000. The 'Median', commonly known as the '50^{th} Percentile' (50%), is 2.0000 for PS1, reflecting the centre of the data distribution when ordered. Furthermore, the '75^{th} Percentile' (75%) which holds 3.0000 for PS1, denotes that about three-quarters (75%) of

the PS1 data points lie at or below this figure.

	PS1	PS2	PS3	PS4	PS5	PS6
count	840	840	840	840	840	840
mean	1.9655	1.9631	2.0571	2.0655	2.1143	2.0940
std	0.8155	0.8110	0.8280	0.8107	0.8119	0.8297
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000
75%	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	PS7	PS8	PS9	PS10	PS11	PS12
count	840	840	840	840	840	840
mean	2.0869	2.0143	2.0536	2.0560	1.9940	1.9714
std	0.8175	0.8154	0.8218	0.8173	0.8059	0.8179
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000
75%	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000

Table 3.1: Summary of genomic data having low fitness values for 840 out of 1000 records

Table 3.1 shows that packaging signals towards the ends of the genomes with low fitness values, notably 'PS1', 'PS2', 'PS11', and 'PS12', have lower mean values than packaging signals in the middle. This indicates that values '1' or '2', which reflect weak or medium binding affinities, respectively, rather than '3', which represents strong binding affinities, are more prevalent towards the genomic ends. The occurrence of weak binding affinities towards the ends of the genome may account for the low fitness values obtained. This inference is supported by Table 3.2, which shows that the packaging signals near the ends of the genomes having high fitness values have larger mean values than those in the central part. This might be due to a dominance of packaging signals with a greater amount of '2' and '3' values, showing medium and strong binding affinities, as compared to '1' values, indicating poor binding affinity. As a result, genomes with stronger packaging signals at the ends might have achieved greater fitness values. Since these findings were obtained for 1000 data points alone, the entire data was analysed using the same method.

Table 3.3 and Table 3.4 give an interesting perspective of the complete genomic landscape, taking into account both low and high fitness levels. Table 3.3 shows that the mean values of packaging signals 'PS1', 'PS2', 'PS11' and 'PS12' are low compared to the packaging signals in the central part of the genome. Whereas, Table 3.4 shows that the packaging signals at the ends of the genome have higher mean values than the ones at the middle part. A strong pattern emerges, just as it did in the study of the 1000 records. Genomes having packaging signals with low and medium binding affinities to their sequence ends have lower fitness scores. When

focusing on genomes with strong binding affinities for packaging signals at the ends, the pattern becomes apparent. Higher fitness values can be found in these genomes. This pattern shows that fitness outcomes are greatly influenced by the strength of these packaging signals at the genomic ends. In summary, these tables provide an intriguing story about how fitness outcomes are highly influenced by the genomic landscape, especially the affinity of packaging signals at the sequence endpoints.

	PS1	PS2	PS3	PS4	PS5	PS6
count	160	160	160	160	160	160
mean	2.1000	2.1938	1.8250	1.7500	1.6875	1.5312
std	0.8333	0.8048	0.7895	0.7004	0.6931	0.6337
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	2.0000	1.0000	1.0000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.0000	2.0000	1.0000
75%	3.0000	3.0000	2.0000	2.0000	2.0000	2.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	PS7	PS8	PS9	PS10	PS11	PS12
count	160	160	160	160	160	160
mean	1.5062	1.6562	1.7312	1.8625	2.2312	2.2125
std	0.6344	0.7358	0.7331	0.7647	0.7948	0.8271
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.0000	2.0000	1.0000
50%	1.0000	1.5000	2.0000	2.0000	2.0000	2.0000
75%	2.0000	2.0000	2.0000	2.0000	3.0000	3.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000

Table 3.2: Summary of genomic data having high fitness values for 160 out of 1000 records

	PS1	PS2	PS3	PS4	PS5	PS6
count	449235	449235	449235	449235	449235	449235
mean	1.9719	1.9699	2.0228	2.0542	2.0613	2.0860
std	0.8143	0.8157	0.8223	0.8232	0.8197	0.8176
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000
75%	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	PS7	PS8	PS9	PS10	PS11	PS12
count	449235	449235	449235	449235	449235	449235
mean	2.0860	2.0613	2.0542	2.0228	1.9699	1.9719
std	0.8176	0.8197	0.8232	0.8223	0.8157	0.8143
min	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000
75%	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
max	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000

Table 3.3: Summary of genomic data having low fitness values for entire data

	PS1	PS2	PS3	PS4	PS5	PS6
count	82206	82206	82206	82206	82206	82206
mean	2.1534	2.1647	1.8754	1.704	1.6652	1.5302
std	0.8114	0.8013	0.7723	0.709	0.7107	0.6293
min	1.0000	1.0000	1.0000	1.000	1.0000	1.0000
25%	1.0000	1.0000	1.0000	1.000	1.0000	1.0000
50%	2.0000	2.0000	2.0000	2.000	2.0000	1.0000
75%	3.0000	3.0000	2.0000	2.000	2.0000	2.0000
max	3.0000	3.0000	3.0000	3.000	3.0000	3.0000
	PS7	PS8	PS9	PS10	PS11	PS12
count	82206	82206	82206	82206	82206	82206
mean	1.5302	1.6652	1.704	1.8754	2.1647	2.1534
std	0.6293	0.7107	0.709	0.7723	0.8013	0.8114
min	1.0000	1.0000	1.000	1.0000	1.0000	1.0000
25%	1.0000	1.0000	1.000	1.0000	1.0000	1.0000
50%	1.0000	2.0000	2.000	2.0000	2.0000	2.0000
75%	2.0000	2.0000	2.000	2.0000	3.0000	3.0000
max	3.0000	3.0000	3.000	3.0000	3.0000	3.0000

Table 3.4: Summary of genomic data having high fitness values for entire data

3.4 Machine Learning Techniques

The main steps used here for machine learning and modelling the data include the following steps:

- **Splitting the dataset**: Both the input variable 'Genome' and output variable 'Fitness' were split into train and test sets in the ratio 70:30. The model uses the train set for fitting the model. The test set will be completely new to the trained model so that the model performance can be evaluated based on its performance on unseen data.

- **Normalisation of data**: The output variable 'Fitness' has values ranging from 0 to 2000. These values were normalised in the range 0 to 1 by dividing each value by 2000. Thus normalisation map 3^{12} genomes to values between 0 and 1 (Dechant and He, 2021a).

- **Hyperparameter tuning**: Hyperparameters are model parameters that are specified before training. It refers to the definition of a set of input parameters that influence the structure of the model and, as a result, the outputs (Xenochristou and Kapelan, 2020).

- **Gridsearch**: Grid search is a machine learning hyperparameter tuning approach used to determine the optimal combination of hyperparameters for the model being studied. Grid search entails setting a grid of probable values for each hyperparameter and then thoroughly assessing every possible combination of these values to obtain the optimum model performance (Malato, 2021).

- **K-fold Cross-validation**: Cross-validation is a technique that divides a learning set into k distinct groups of equal size with no overlap. The model is trained on the training set, which is represented by k-1 subsets, followed by the evaluation of the validation set. To determine cross-validated performance, the average of the k performance measurements on the validation sets is employed repeatedly (Berrar, 2019).

- **Model evaluation**: Finally the model evaluation can be done using performance metrics such as MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and Coefficient of Determination (R^2) (Yao, 2022).

  Mean Absolute Error (MAE): The average absolute difference between the predicted values and the actual values is measured using the metric known as Mean Absolute Error (MAE). Better model performance is shown by a lower MAE, which is particularly helpful when outliers are present in the data (Yao, 2022). MAE is calculated using the following equation (Chicco et al., 2021).

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|X_i - Y_i|$$

  .

Root Mean Square Error (RMSE): RMSE is the average difference between the predicted and actual values of a statistical model. It is the standard deviation of the residuals. The gap between the regression line and the data points is represented by residuals. The RMSE measures the dispersion of these residuals, demonstrating how closely the observed data clusters around the expected values. RMSE values can vary from zero to positive infinity and have the same units as the predicted variable. Lower RMSE values show better model performance (Frost, 2019). RMSE value is given by the following equation (Chicco et al., 2021).

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2}$$

Coefficient of Determination or R-squared (R^2) value : (R^2) value is a statistical indicator that shows how much of the variance in the dependent variable can be explained by the independent variables in a regression model. R-squared is sometimes referred to as the coefficient of determination. A better fit of the model to the data is indicated by higher values, which range from 0 to 1. An R-squared score of 1 denotes that the model accurately predicts the dependent variable's variance, whereas a value of 0 denotes that the model offers no benefit over a straightforward mean-based prediction (Yao, 2022). The R^2 value is given by the following equation (Chicco et al., 2021).

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(X_i - Y_i)^2}{\sum_{i=1}^{m}(\bar{Y} - Y_i)^2}$$

$\mathcal{F}$-statistic: The $\mathcal{F}$-statistic calculates the ratio of two variances: the unexplained variance found within the residuals and the model's explained variance between the predicted and actual values. In simple terminology, it assesses whether the model's predictions adequately account for the variance in the data. An improved model's ability to explain the data is shown by a higher $\mathcal{F}$-statistic (Frost, 2017).

***P*-value**: The probability of obtaining an $\mathcal{F}$-statistic as high as the one obtained can be determined by the *p*-value, which is calculated under the null hypothesis. When performing a regression analysis, the null hypothesis often says that the model's coefficients (parameters) have no significant impact, i.e., the model fails to account for the variance in the data. The model's performance is statistically significant when the *p*-value is low (often less than 0.05) (Frost, 2017).

3.4.1 Random Forest Regression

A Random Forest regression is a sophisticated machine learning approach that falls under the broader category of methods for ensemble learning. Ensembles integrate numerous separate

models to achieve a more stable and precise prediction. The "forest" in "Random Forest" refers to a collection of decision trees, which are a type of model applied in different machine learning applications. Random forest regression is specially developed for regression problems in which the aim is to predict a continuous numerical output. It is a modification of the more renowned Random Forest classification technique (Breiman, 2001).

During the training phase, the algorithm creates a large number of decision trees and then aggregates their predictions to generate a final prediction. Individual trees are constructed using bootstrap samples instead of the original data. This is known as bootstrap aggregation or bagging, and it reduces overfitting. Overfitting means that the model closely replicates the special properties of the training data, resulting in poor performance when given to the test data. A Random Forest's decision trees are built one at a time. The choice of which predictor variables to employ for splitting a node in a decision tree is decided by preset criteria that serve as optimisation problems in both classification and regression tasks. When dealing with regression problems, minimising the mean squared error calculation for each internal node is a widely used criterion for separating nodes, which is given by the equation

$$E_{X,Y}((Y - h(X)))^2,$$

where X, Y is a random vector, $E_{X,Y}$ is the generalisation error, $h(X)$ is the numerical predictor (Breiman, 2001; Schonlau and Zou, 2020).

The advantages of the Random Forest model are given below (Buskirk, 2018).

- Reduces overfitting by aggregating results from various decision trees.
- Capable of processing a range of predictor types, including continuous, categorical, skewed, and sparse data.
- Performs well when there exist complex relationships between predictor and target variables as well as with non-linear data.
- The feature importance can be obtained easily.

However, there are some disadvantages as well with Random Forest models (Buskirk, 2018). They are:

- Difficulty in visualising the model.
- Variables obtained as important may be biased if there exists a correlation between the variables.
- Missing data should be treated appropriately to avoid errors.

In this project, a Random Forest regressor having the parameters listed in Table 3.5 was used. These parameters were obtained as the best parameters when hyperparameter tuning was

done using grid search. To validate the model performance, 5-fold cross-validation was done. The grid search uses cross-validation to evaluate various combinations of parameters and based on the scoring metric, which is R^2 score by default, returns the best set of parameters.

Hyperparameter	Value
n_estimators	150
min_samples_split	10

Table 3.5: Hyperparameters used for the Random Forest model

3.4.2 Recurrent Neural Network

A Recurrent Neural Network (RNN) is a type of neural network that uses sequential or time-series data and passes the previous step's output as input to the present stage. Recurrent networks, like feedforward and Convolutional Neural Networks (CNN), learn from training input albeit vary from others since they have "memory", which allows them to influence current input and output by making use of information from prior inputs. Recurrent Neural Networks (RNNs) are made up of several layers that feature a feedback mechanism, giving them a powerful capacity to analyse data sequences. Each neuron in an RNN has some type of short-term memory, which allows it to transfer important information to itself in the following time steps. The integration of this internal memory with the feedback loop allows for links between current and previous outputs, enabling data flow from the past to the present. RNNs, in a nutshell, allow information to be moved over time, and connections between neuron weights are changed via a process known as backpropagation through time (BPTT) (Werbos, 1990). This technique guarantees that RNNs are well-equipped to deal with sequential data analysis. However, ordinary recurrent networks suffer from vanishing gradients, which makes learning large data sequences difficult (Sarker, 2021; Keshk et al., 2023).

Long Short-term memory (LSTM) is a kind of short-term memory. This is a common type of RNN design that employs special units to solve the vanishing gradient problem, which was proposed by (Hochreiter and Schmidhuber, 1997). An LSTM unit's memory cell can hold data for lengthy periods, and the flow of data into and out of the cell is regulated by three gates. Particularly, the 'Forget Gate' decides what information from the prior state of the cell has to be memorised while any data that has become irrelevant will be eliminated, whereas the 'Input Gate' chooses the data that is required for the cell state and the 'Output Gate' sets and regulates the outputs. The LSTM network is regarded as one of the best since it eliminates the problems associated with training a recurrent network (Sarker, 2021).

To begin, the forget gate is critical in selecting which information inside the cell state should be maintained or destroyed. To do this, a sigmoid activation function is used, which takes into consideration the prior hidden state (h_{t-1}), current input (x_t), and previous cell state (c_{t-1}) to create a value ranging from 0 to 1 for each of the elements in the cell state c_{t-1}, as shown in equation 3.1. A value of 0 indicates the decision to "discard all information" whereas a value of

1 indicates the decision to "preserve all information" (Keshk et al., 2023).

Subsequently, the cell evaluates new input and chooses which information should be integrated into the cell state. This is performed using a sigmoid layer, which decides the data to be updated, as specified by equation 3.2. Meanwhile, the tanh activation function creates a new vector of values. Merging these two values yields an updated cell state that includes both the recorded information from the previous cell state c_{t-1} and the vector of new values created by the tanh activation function (Keshk et al., 2023).

Following the completion of the computations in the preceding stages, a new cell state c_t is formed using equation 3.4. This new state keeps certain components from c_{t-1}, a decision driven by f_t (the forget gate output), and incorporates newly calculated information from the second phase. Finally, equation 3.5 regulates the flow of information in the form of output. The output gate, which similarly uses a sigmoid activation function, runs the cell state c_t through the tanh function before multiplying it by the sigmoid layer's result (Keshk et al., 2023).

The variables m_t, f_t, and o_t in the given equations indicate the memory, forget, and output gate values at time t, respectively. Meanwhile, x_t, h_t, and c_t represent the input layer, hidden layer, and cell state at time t, respectively. The variables beginning with "b" represent bias vectors, whereas those beginning with "W" represent weight matrices. Furthermore, σ denotes the sigmoid activation function, whereas tanh denotes the hyperbolic tangent function. The symbol $\odot$ represents element-wise multiplication, which is used to perform certain operations in various areas of the equations (Keshk et al., 2023).

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \tag{3.1}$$

$$m_t = \sigma\left(W_{xm}x_t + W_{hm}h_{t-1} + W_{cm}c_{t-1} + b_m\right) \tag{3.2}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \tag{3.3}$$

$$c_t = f_t \odot c_{t-1} + m_t \odot \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{3.4}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{3.5}$$

In this project, two Recurrent Neural Network Models were created. To train the Recurrent Neural Network (RNN), three critical hyperparameters were used in the training approach. First, the hyperparameter 'epochs', which indicates the number of full passes the training dataset should go through during the training, was set to 50. This means that the complete dataset was used repeatedly 50 times to improve the model's performance by modifying its internal weights. Second, the 'batch size' was set to 10, representing the number of training instances handled in each optimisation cycle. Smaller batch sizes were used to save memory utilisation and encourage more frequent weight changes, potentially improving convergence. Finally, a 'validation split' of 0.2 was used, which set aside 20% of the training data for validation. This enables continuous assessment of how the model performs on independent data to avoid overfitting and guarantee robust generalisation to previously unknown datasets. The architecture of both of

these models is explained below.

Model 1

A recurrent neural network (RNN) with two fundamental layers was used as the first model in this study. The first layer is made up of a Long Short-Term Memory (LSTM) architecture, which is specially built for processing sequences of 12 time steps, with each time step corresponding to a specific feature or variable inside the sequence. The LSTM layer is particularly good at detecting complicated patterns and connections in sequential data.

The second layer is a Dense layer with a single neuron that is completely connected. This layer acts as the output layer, predicting a single numerical result. The major goal of the training procedure was to minimise the mean squared error (MSE) loss, which is a statistic that evaluates the average squared discrepancy between the model's predictions and the actual target values. The Adam optimizer, known for its performance, was used to maximise training efficiency.

Model 2

The second model is a multi-layered recurrent neural network (RNN) designed to analyse sequential data. The model is made up of multiple discrete layers, each with its own set of properties. The first layer is a 64-unit LSTM (Long Short-Term Memory) layer. This layer accepts input sequences of length 12 with a single feature. The activation function utilised in this case is 'linear', which means it does not add non-linearities to the model. This layer is intended to return sequences for additional processing.

The initial LSTM layer is followed by a second LSTM layer with 32 units and a 'sigmoid' activation function. This layer also yields sequences, allowing the model to capture the data's complicated temporal patterns. Following that, a third LSTM layer with 16 units and a 'linear' activation function was added. This layer, unlike the preceding layers, does not return sequences, implying that it summarises the sequential data and feeds it to the following layers.

The output design of the network consists of three dense layers, each with 64, 32, and 1 neurons. Two of these layers use 'sigmoid' activation functions and the last layer uses 'linear' activation function to obtain a numerical output. Throughout the training phase, the model strives to minimise the mean squared error (MSE) loss, which is a common choice in regression problems, with optimisation handled by the Adam optimizer.

3.4.3 Feature Importance from Random Forest Model

Machine learning techniques usually lack explicit equations for explaining the working mechanism behind the model. To understand the 'black box' behind the models, variable importance was developed (Grömping, 2015).

The Random Forest algorithm includes two approaches for calculating feature importance:

- **Gini Importance (Mean Decrease Impurity)**: A collection of Decision Trees is built in the Random Forest structure. Internal nodes and leaves are present in each Decision Tree. The internal nodes divide the dataset into discrete groups with comparable answers by using particular features. Internal node features are selected based on parameters such as Gini impurity or information gain for classification jobs and variance reduction for regression. The impurity drop produced by each feature for a split is assessed, and the feature with the greatest decrease is chosen for the internal node. The relevance of each characteristic is determined by the average decrease in impurity produced by each feature across all trees in the forest. This method is provided in Scikit-learn's Random Forest implementation for both classification and regression applications. It is critical to look at relative relevance levels rather than absolute values. Because all relevant parameters are calculated during Random Forest training, this approach has a high computational speed. It does, however, favour numerical and categorical characteristics with large cardinality. Furthermore, in the event of associated features, it may choose one feature while overlooking the significance of another, thus leading to inaccurate findings (Płoński, 2020).

- **Permutation Based Feature Importance (with scikit-learn)**: The scikit-learn API in python has 'permutation_importance' method. This method takes in the trained model and test data as input parameters. This function shuffles each feature at random and computes the change in model performance. The most significant features are those that have the greatest influence on performance (Płoński, 2020).

Feature importance based on 'Gini Importance (Mean Decrease Impurity)' (Płoński, 2020) was used in this project to analyse the role played by each feature in training the model and prediction.

3.4.4 SHapley Additive exPlanation (SHAP)

The ability to effectively comprehend the output of a prediction model is crucial. It fosters adequate confidence among users, offers insight into how to develop a model, and aids comprehension of the process being modelled. Simple models tend to be chosen in various applications due to their ease of understanding, even if they are less accurate than complicated ones. However, as the amount of data to be processed grows, more the benefits of applying complicated models have grown (Lundberg and Lee, 2017).

SHAP's key idea is to apply Shapley values, a concept derived from cooperative game theory, to evenly distribute each feature's contribution among all the other features. In cases of multicollinearity, Shapley regression values are used to determine the relevance of features in linear models. Retraining the model on all subsets of features is required for this strategy. It assigns a significance value to each feature, which shows the influence of incorporating that feature on model prediction. To measure this effect, a model with the feature contained is trained, and another model with the feature removed is trained. The two models' predictions are com-

pared based on the present input. Due to the effect of removing a feature relying on the model's other features, previous differences are estimated for all potential subsets. The Shapley values are then calculated and adopted to attribute features. They are the weighted mean of all potential differences (Lundberg and Lee, 2017).

The Shapley value is given by the equation:

$$\phi_j(val) = \sum_{S \subseteq M \setminus \{j\}} \frac{|S|!(m - |S| - 1)!}{m!} [\text{val}(S + j) - \text{val}(S)], \quad j = 1 \dots m,$$

where:
S is the chosen subset of features, and $|S|$ is the number of features in this subset. M denotes the whole selection of features, while m denotes the total number of features. The j^{th} feature in the vector representing a feature's values of the instance which requires explanation is denoted by j. The functions $val(S)$ and $val(S + j)$ give values to subsets of features S and S with the addition of the j^{th} feature, respectively. The Shapley value has four important properties: efficiency, symmetry, dummy, and additivity (Lundberg and Lee, 2017; Keshk et al., 2023).

- **Efficiency**: According to the Efficiency property, the contributions of features should total up to the prediction for a single instance, which is equal to the difference between the actual output and the average prediction (Lundberg and Lee, 2017; Keshk et al., 2023).
- **Symmetry**: When two unique features, i and k, are considered inside the feature subset S, their contributions should be equal to each other for all feasible combinations. As a result, the function describing the Shapley value should be symmetric (Lundberg and Lee, 2017; Keshk et al., 2023).
- **Dummy**: If the presence of a certain feature has no effect on the prediction, its Shapley value should be zero, suggesting that it does not contribute to the model's choice (Lundberg and Lee, 2017; Keshk et al., 2023).
- **Additivity**: When two distinct functions that provide feature values are combined, the gain that results from the combination should be equal to the total of the gains from each function taken individually for each feature (Lundberg and Lee, 2017; Keshk et al., 2023).

Chapter 4

Results

The machine learning approaches described in Section 3.4 were used to build different machine learning models. These models were trained using the given dataset and then tested with the test dataset. To compare the models, performance metric scores were used. Furthermore, important features were investigated using the best-performing model. This forms the framework for a comprehensive exploration of the results in the following section.

4.1 Random Forest Model

The results obtained from the Random Forest Regression model are shown below. The scatter plot in Figure 4.1 depicts the positioning of predicted and actual values of the target variable 'Fitness', which was normalised to a range between 0 and 1. The orange line in the graph represents points where predicted values exactly match the actual values, and it serves as the benchmark line for perfect predictions. Looking at the figure, it's clear that the majority of data points cluster around this line. This finding shows that the model's predictions are fairly close to the actual values, suggesting its accuracy.

The metric scores of the model were obtained to see the performance of the model and are given in Table 4.1. In this case, an MAE of 0.0061 indicates that the model's predictions differ from the actual values by around 0.0061 units on average. Lower MAE values suggest that the model is more accurate. RMSE is similar to MAE, but it emphasises larger errors by calculating the square root of the differences before averaging. A root mean square error of 0.01 indicates that the usual prediction error once squared and averaged, results in a root mean square error of 0.01. Lower RMSE values, like MAE, imply more accuracy in prediction. With an R^2 of 0.9883, the model explains roughly 98.83% of the variance in the Fitness values. In simplified terms, it means that the predictions of the model closely match the real fitness values, and it's an effective measure of model performance.

Based on these criteria, the Random Forest Model appears to perform very well. It has a low MAE and RMSE, suggesting that it makes accurate predictions with little error. Furthermore, the high R^2 value indicates that the model explains a considerable percentage of the fitness

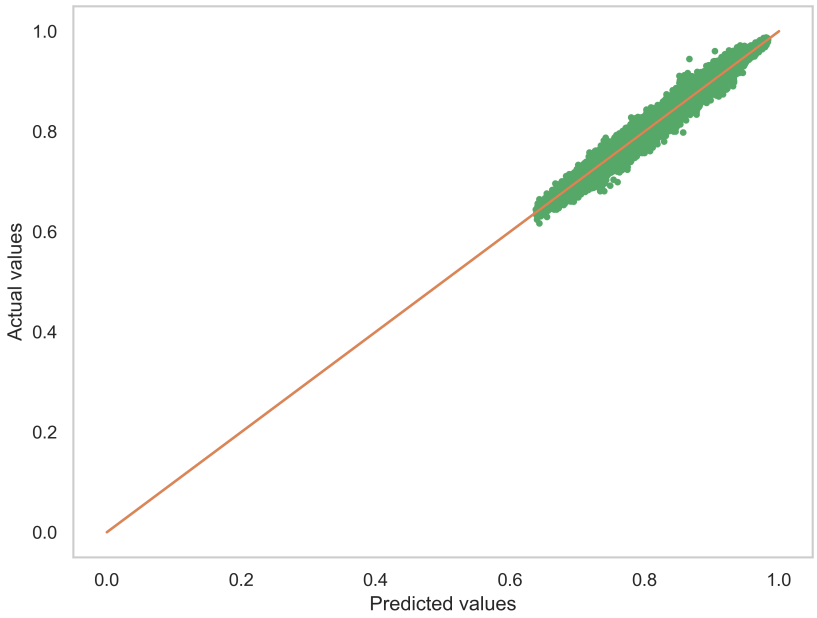

Figure 4.1: Actual versus predicted values from Random Forest regressor model

variation, showing a strong fit to the data. This combination of low error metrics and a high R^2 score demonstrates the model's ability to predict fitness values.

	MAE	RMSE	R^2
Random Forest Model	0.0061	0.01	0.9883

Table 4.1: Metric scores of Random Forest model

Understanding how the model performs as the quantity of training data varies can be done by charting the R^2 value versus the percent of training data used. Figure 4.2 shows the R^2 value for varying percentages of training data. The plot shows that the R^2 value steadily grows as the proportion of data given into training increases.

Based on the Random Forest model, a feature importance study was performed, and the outcomes are represented in the bar chart in Figure 4.3. The chart demonstrates that some packaging signals in the middle part of the genome, especially 'PS7' and 'PS6' but also 'PS4', 'PS3', 'PS9', 'PS5', 'PS10', and 'PS8', have a greater influence on the model's predictions than do signals near the ends of the genome. As a result, it may be inferred that signals positioned closer to the middle of the genome have a greater influence on the model's predictive abilities than do signals located at the ends. This finding shows that some packaging signals, particularly those in the centre, are more important for the model's capacity to produce reliable predictions and need more research into the biological implications of these signals.

Furthermore, the best line of fit was obtained for the model as shown in Figure 4.4. The equation of the line is $y = 0.99x + 0.01$. This model's $\mathcal{F}$-statistic and p-value were, respectively,

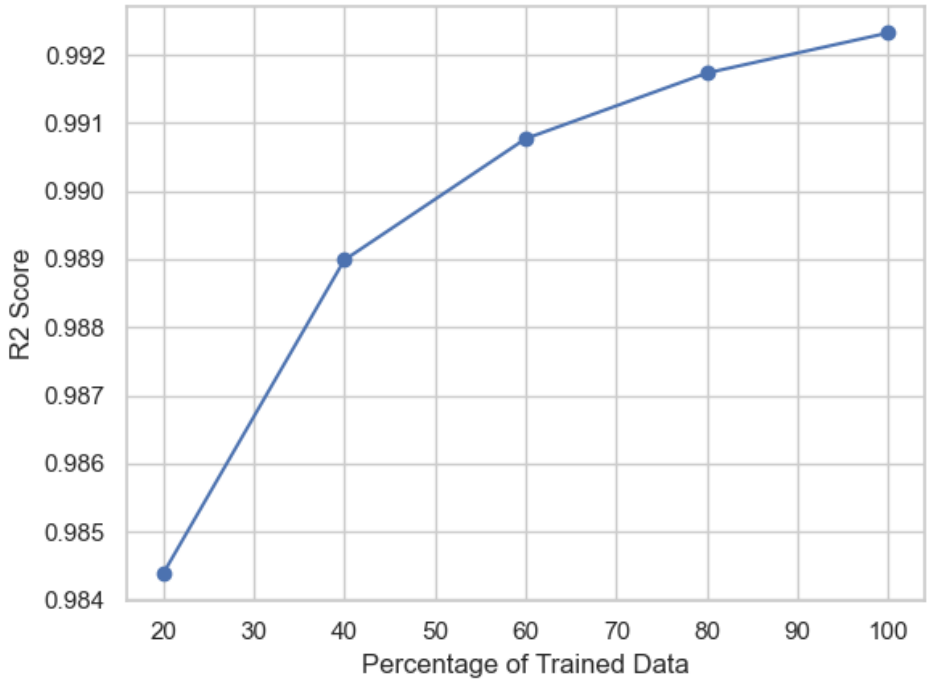

Figure 4.2: R^2 values versus percent of training data curve for Random Forest Model

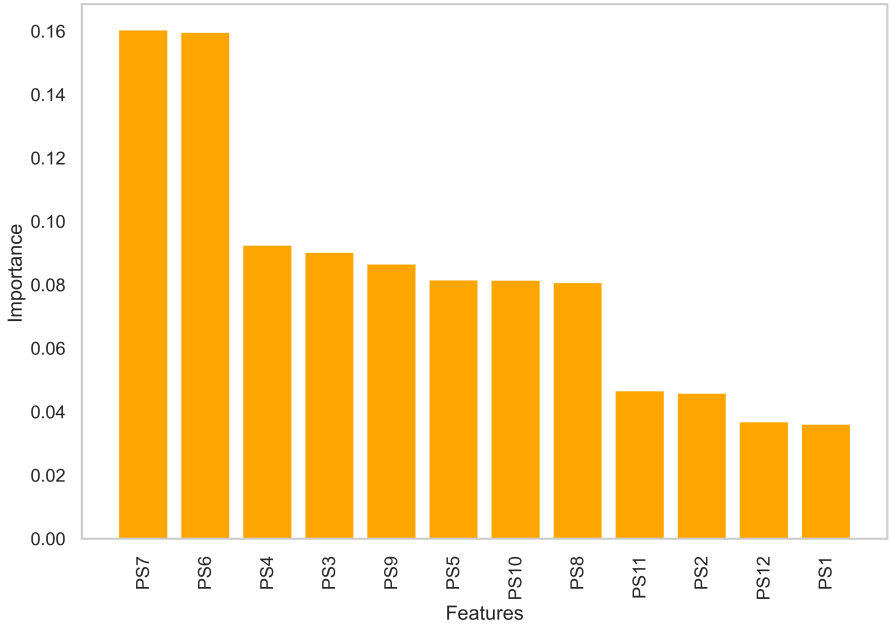

Figure 4.3: Feature importance from Random Forest regressor model

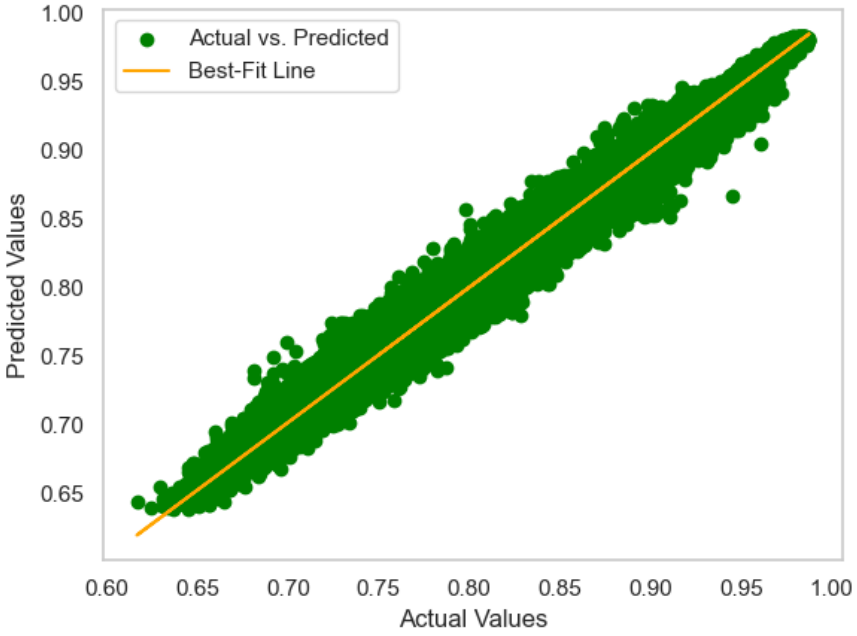

Figure 4.4: Best line of for Random Forest regressor model

0.0118 and 0.9133. These statistical measurements are essential for evaluating the effectiveness of the model. In this case, the model's performance is statistically insignificant as evidenced by the high p-value, which surpasses 0.05, and the poor $\mathcal{F}$-statistic. This implies that the predictions of the model do not significantly deviate from random chance. The RF model's low $\mathcal{F}$-statistic and high p-value indicates that it has trouble identifying significant patterns in the data, limiting its predictions to random guesses. Therefore a better model is required to identify the patterns underlying in the data.

4.2 Recurrent Neural Network Model 1

The results of the first recurrent neural network (RNN) model, which consists of an LSTM layer and a dense layer, are presented below. The scatter plot from Figure 4.5 shows a comparison between the actual and predicted values based on how well the model performed on the test dataset. The optimum scenario, when actual and predicted values fully match, is shown by the black line. The fact that the data points cluster tightly around this line highlights the significant agreement between the model's predictions and the actual values. In addition, the relationship between the predicted and actual values consequently led to the line $y = 1.02x - 0.01$ being the line of best fit for this data as shown in Figure 4.6.

The $\mathcal{F}$-statistic and p-value obtained for the recurrent neural network are 13.1365 and $2.8964 \cdot 10^{-4}$. The $\mathcal{F}$-statistic of 13.1365 indicates that the RNN model performs a fine job

describing the variance in the dataset, meaning that its predictions align well with real values. This is supported by the significantly low p-value of $2.8964 \cdot 10^{-4}$, indicating a very high degree of statistical significance. This suggests that there is substantial evidence that the RNN model's predictive powers are not the consequence of random chance, but rather of its capacity to detect meaningful patterns in data. These results indicate that this RNN model is a helpful tool for making precise predictions based on the available data and that its performance evaluation is statistically significant.

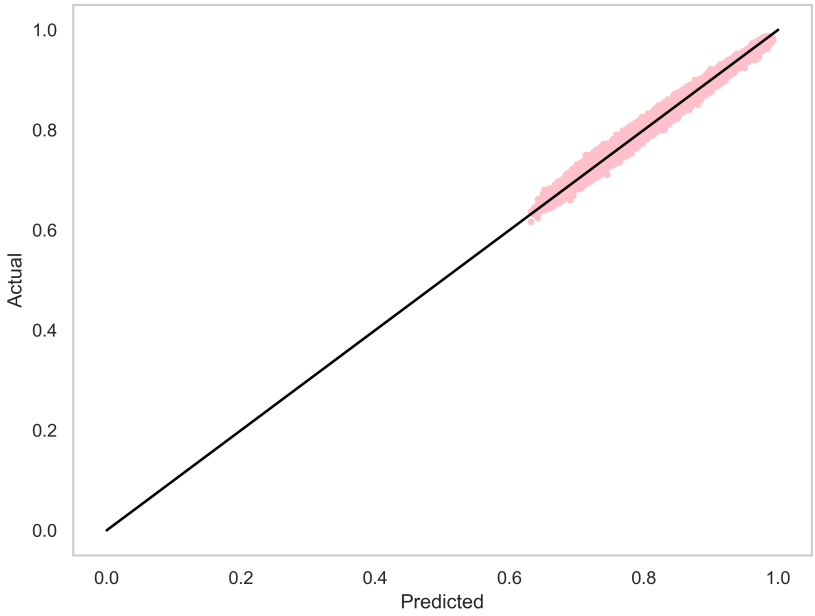

Figure 4.5: Actual versus predicted values from recurrent neural network 'model 1'

The MAE, RMSE and R^2 values of this model are given in Table 4.2. The evaluation of RNN Model 1 reveals a high level of prediction accuracy. The MAE value of 0.0057 indicates a really small average difference between the model's predictions and the actual observations. Additionally, an RMSE of 0.01 highlights the model's constant ability to sustain minimal prediction disparities. The model's ability to adequately explain 99.06% of the variance in the dataset is demonstrated by the model's R^2 score of 0.9906. These findings support RNN Model 1's prediction ability, proving it is a reliable analytical tool for this particular dataset.

	MAE	RMSE	R^2
RNN Model 1	0.0057	0.01	0.9906

Table 4.2: Metric scores of RNN model 1

To examine the model's possible overfitting, a learning curve was produced to visualise the

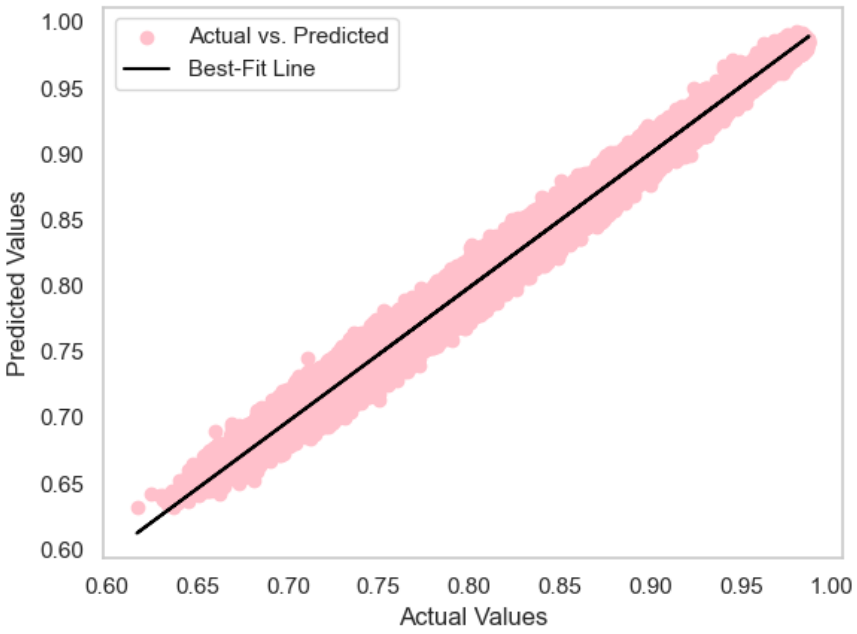

Figure 4.6: Best line of fit for 'RNN model 1'

training loss and validation loss through each training iteration, as shown in Figure 4.7. The finding is particularly evident: as the training loss decreases, so does the validation loss, providing strong proof that the model is not overfitting. Minor oscillations in the validation loss are seen due to the use of a very small batch size, but these fluctuations do not disturb the overall trend of decreasing validation loss. Notably, the validation loss constantly remains smaller than the training loss all over the learning process, demonstrating the model's ability to generalise successfully to unknown data. A plot depicting the variation in R^2 values across different amounts of training data was generated to obtain more insights into the model's learning behaviour (see Figure 4.8). The figure illustrates an intriguing pattern: initially, the R^2 value rises steadily until it reaches 40% of the training data. However, there is a little drop at the 60% point. Despite this decrease, the model shows the potential to recover and improve its R^2 score, steadily increasing towards 100% of the training data. This pattern implies that the model performs best when given a large quantity of training data, while there may be some variations in its performance at intermediate data percentages. To solve the performance issues noticed at intermediate data percentages, an alternate model was devised.

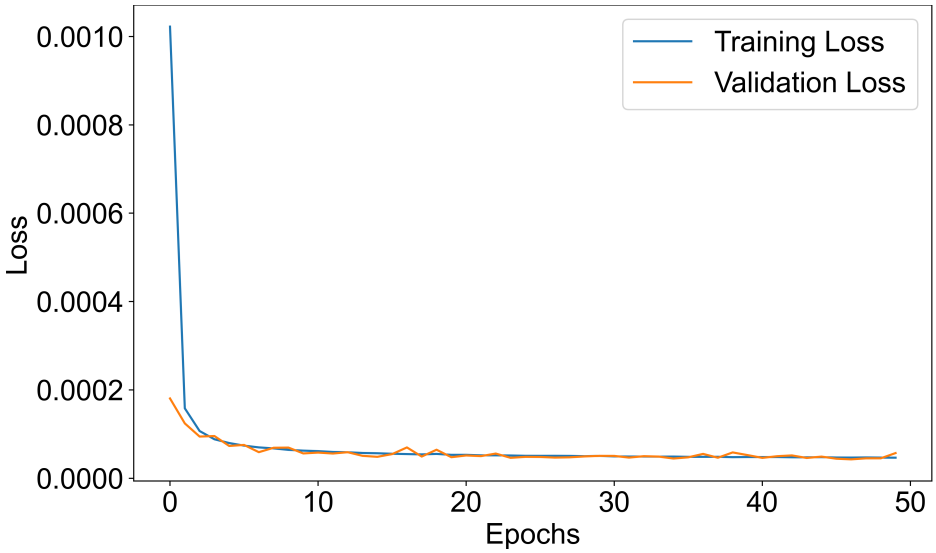

Figure 4.7: Learning curve with training and validation loss of 'RNN model 1'

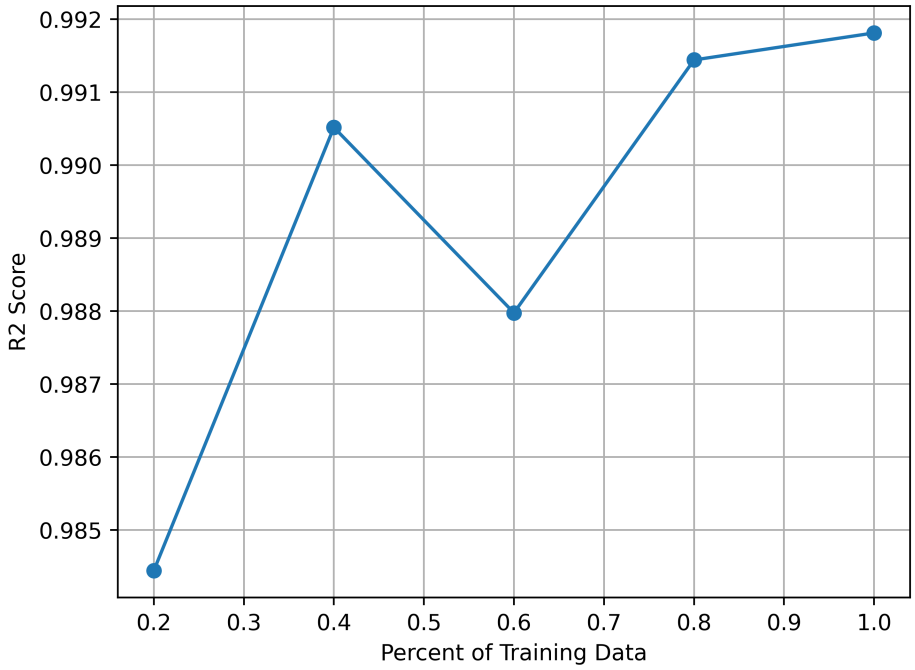

Figure 4.8: Learning curve: R^2 value versus percent of training data for 'RNN model 1'

4.3 Recurrent Neural Network Model 2

The findings of the second neural network model, which has six layers, are shown below. The scatter plot in Figure 4.9 depicts the model's alignment of actual and projected values for the test dataset. The bulk of data points here tightly concentrate around the ideal line when predicted values perfectly match the actual values. This arrangement shows that the model's predictions are remarkable. The line of best fit shown in Figure 4.10, represented by the equation $y = 1.01x - 0.01$, emphasises the relation between expected and actual values. This line falls really close to the ideal line which meets the $x = y$ condition. Further quantitative analysis suggests a strong performance. The model has a very low p-value of $3.1374 \cdot 10^{-27}$ and a remarkably high $\mathcal{F}$-statistic of 116.8454. These statistical measures confirm the model's impressive predictive accuracy and ability to determine the target variable consistently and reliably.

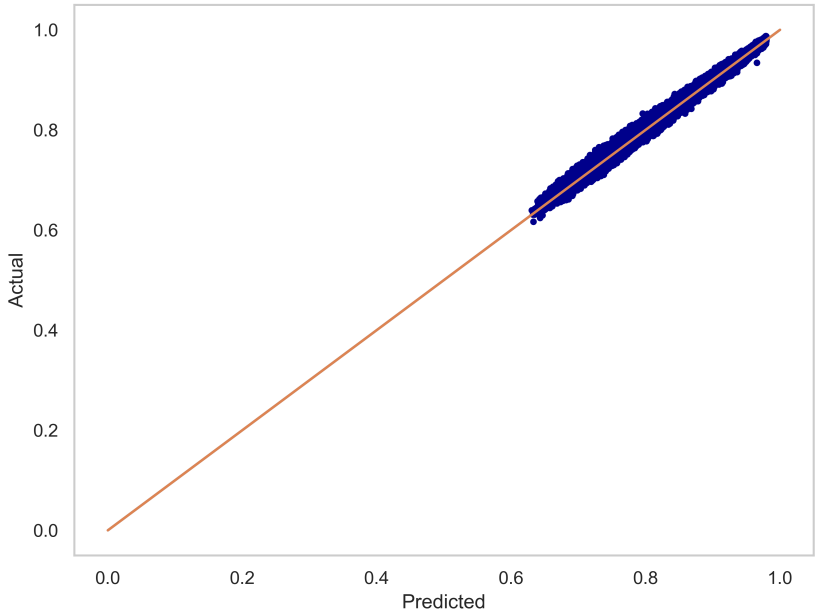

Figure 4.9: Actual versus predicted values from recurrent neural network 'model 2'

Table 4.3 shows the performance metrics of the 'RNN model 2'. The very low MAE score of 0.0057 for this model indicates that it makes precise predictions with minor average prediction deviations. The RMSE score of 0.01 confirms its accuracy by demonstrating that prediction errors fall in close proximity to real values. An R^2 score of 0.9904 indicates that this model successfully accounts for almost 99.04% of the variance in the target variable. These results indicate this model's exceptional predictive capacity as well as its potential to consistently predict the target variable with great precision.

A learning curve was developed to identify potential overfitting by showing the training and validation loss patterns over iterations as shown in Figure 4.11. With each repetition, both

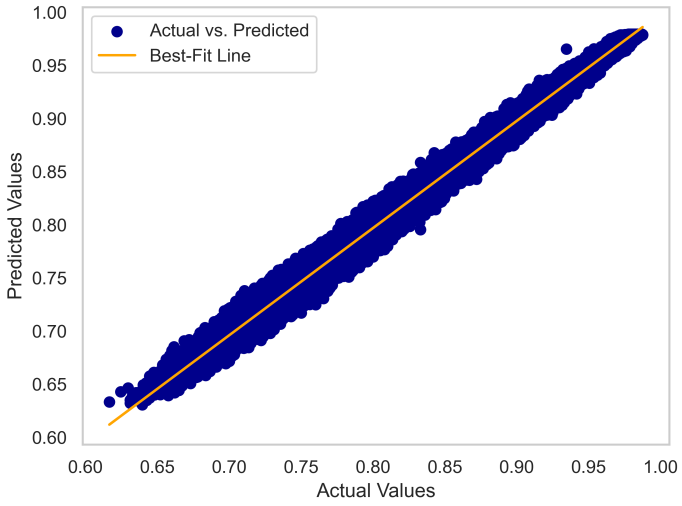

Figure 4.10: Best line of fit for 'RNN model 2'

	MAE	RMSE	R^2
RNN Model 2	0.0057	0.01	0.9904

Table 4.3: Metric scores of RNN model 2

training and validation losses decreased significantly. Despite slight changes, mostly due to the small batch size, the validation loss continuously followed a downward trend. Notably, it stayed lower than the training loss throughout the training phase, demonstrating that the model can effectively generalise to unknown data.

The graph displaying the relationship between the R^2 value and the percentage of training data in Figure 4.12 shows a growing trend as more data is used for training. This result suggests that increasing the amount of training data typically improves the model's prediction accuracy. The absence of any noticeable decline or fall highlights the model's ability to efficiently use bigger datasets, resulting in persisting performance enhancements. Furthermore, the model exhibits constant and impressive performance even with a very small fraction of the training data, precisely 20%, where the R^2 score remains high at 0.98875. This discovery emphasises the model's capacity to use a small quantity of training data effectively while retaining prediction accuracy. It implies that the model has learnt useful patterns and correlations from the given data, allowing it to generate accurate predictions even when data resources are limited.

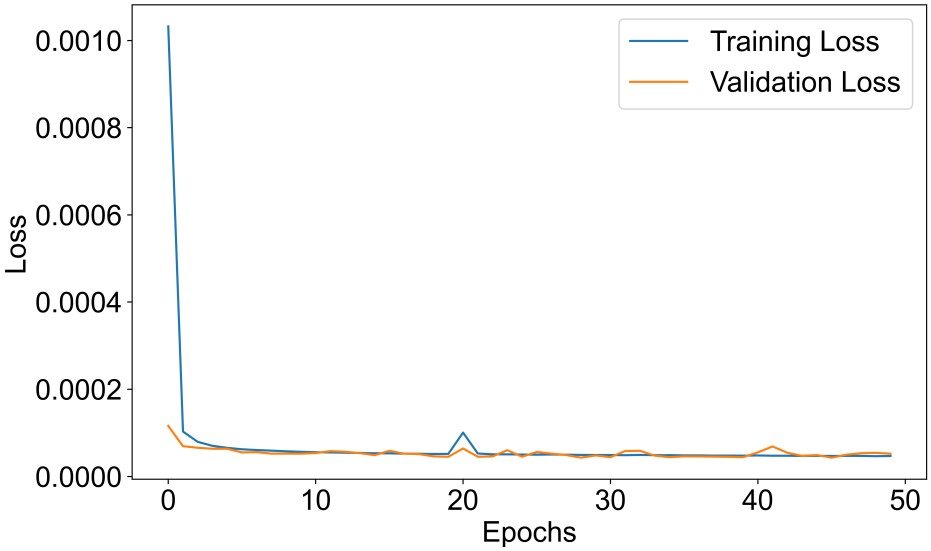

Figure 4.11: Learning curve with training and validation loss of 'RNN model 2'

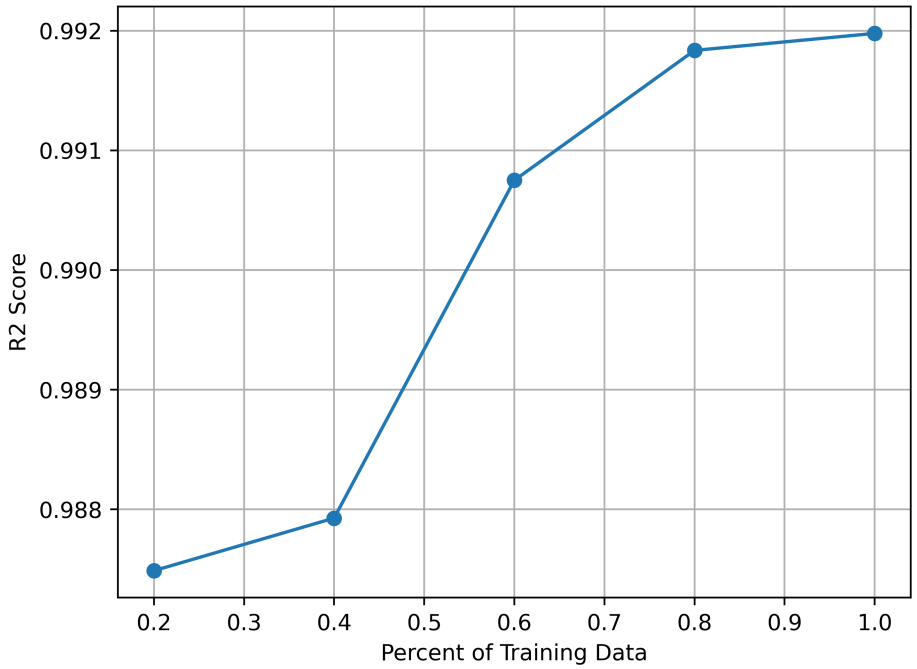

Figure 4.12: Learning curve: R^2 value versus percent of training data for 'RNN model 2'

4.4 Model Comparison

The comparison includes three independent models: the Random Forest Regressor, a Recurrent Neural Network with two layers (RNN Model 1), and a Recurrent Neural Network with six layers (RNN Model 2). Table 4.4 clearly depicts all of their different metric scores. Following an extensive analysis, it is clear that RNN Model 1 is the winner in terms of model performance evaluation. This claim is supported by the fact that it achieved the lowest Mean Absolute Error (MAE) score and the greatest R^2 score in the group, with RNN Model 2 following closely behind.

	MAE	RMSE	R^2
Random Forest Model	0.0061	0.01	0.9898
RNN Model 1	0.0057	0.01	0.9906
RNN Model 2	0.0057	0.01	0.9904

Table 4.4: Metrics scores of all the models

	$\mathcal{F}$-statistic	p-value
Random Forest Model	0.0118	9.133000e-01
RNN Model 1	13.1365	2.896400e-04
RNN Model 2	116.8454	3.137400e-27

Table 4.5: $\mathcal{F}$-statistic and p-values of models

However, ranking the model based on metrics score alone is not appropriate. Based on the relationship between R^2 scores and the percentage of training data, RNN Model 2 retains its excellent performance even when trained with a minimal proportion of data. The Random Forest model also shows promise in this regard, indicating its adaptability.

Despite this, while evaluating the $\mathcal{F}$-statistic and p-value measurements as given in 4.5, the Random Forest model falls short in terms of statistical significance. It has a significantly low $\mathcal{F}$-statistic and high p-value, raising uncertainties regarding the accuracy of its projections. RNN Model 2, on the other hand, not only outperforms the other two models in terms of R^2 scores but also in terms of statistical support, as evidenced by its better $\mathcal{F}$-statistic and p-value scores. This thorough evaluation points to the clear judgement that RNN Model 2 is the best choice among these models. Its combination of noteworthy prediction ability and strong statistical foundations qualifies it as the most reliable and efficient model among the competitors.

4.5 Feature importance using SHAP values

The SHAP summary plot, created using RNN Model 2 (see Figure 4.13), provides insight into the influence of packaging signals on fitness values. The x-axis indicates SHAP values, while

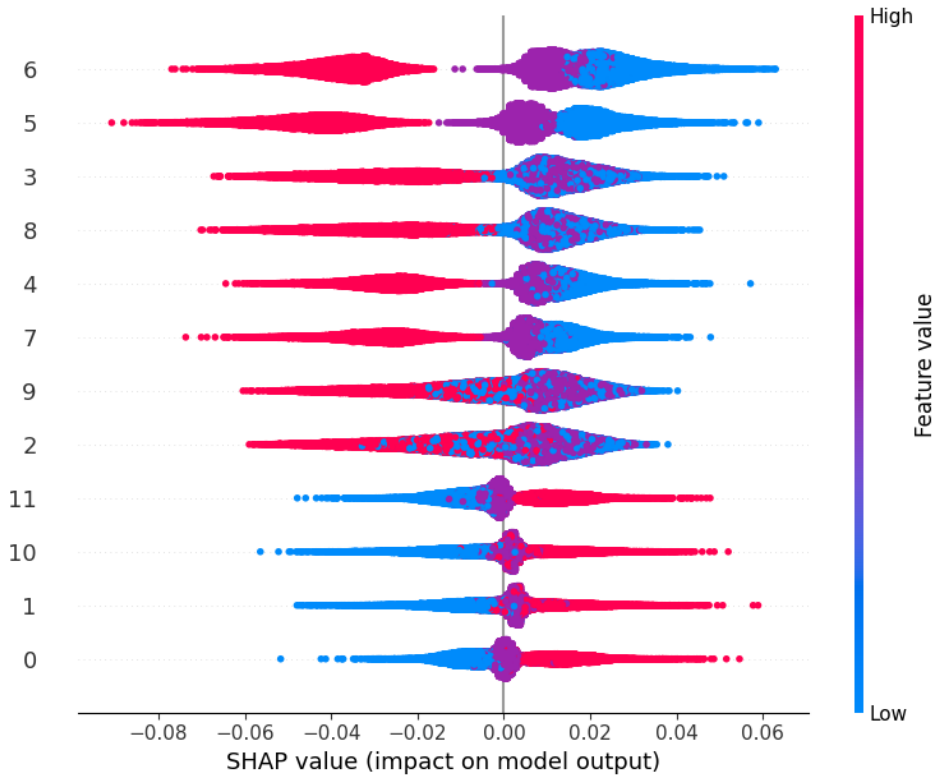

Figure 4.13: SHAP plot for RNN model

the y-axis depicts the decreasing order of significance of features (specifically, packaging signals) from top to bottom, as they influence the target variable 'Fitness'. The figure uses a colour scheme with three unique hues: blue, violet, and red, which represent the weak, medium, and high binding affinities of packaging signals, represented by values 1, 2, and 3 in the genomic sequence. An intriguing pattern emerges from a careful inspection of this plot.

Consider, for instance, the packaging signal at the seventh position, denoted as 'PS7' and indexed as 6. This specific packaging signal plays an essential role in the model's predictive capability. When this signal has weak or medium binding affinities, it has a positive impact on the model's predictions. On the other hand, the presence of a strong binding affinity has a negative influence on the model's performance. Essentially, this means that when the value of the packaging signal at the seventh position increases, the corresponding 'Fitness' value gets reduced and vice versa. Considering the genomic symmetry, the sixth packaging signal 'PS6' (indexed 5), also impacts the model but has a more severe negative effect. Similarly, the fourth (indexed as 3) and ninth (indexed as 8) positions, abbreviated as 'PS4' and 'PS9', as well as the fifth (indexed as 4) and eighth (indexed as 7) positions, abbreviated as 'PS5' and 'PS8', and the third (indexed as 2) and tenth (indexed as 9) positions, abbreviated as 'PS3' and 'PS10', were also found to have a negative impact on the model's fitness value prediction.

On the contrary, packaging signals located at the genome's ends, notably those in the first, second, eleventh, and twelfth places (indexed as 0, 1, 10, and 11, respectively), have a positive

effect on the 'Fitness' score. This indicates that when the binding affinity of the packaging signals at the ends of the genome increases, so does the 'Fitness' value. However, given the larger significance of the middle half of the genomes, obtaining a higher 'Fitness' score requires packaging signals with lower binding affinities in these parts of the genome.

This finding closely resembles the inferences from the UMAP. The 3.3 and 3.4 tables give an interesting perspective of the complete genomic landscape, taking into account both low and high fitness levels. A strong pattern emerges, just as it did in the study of the 1000 records. Genomes having packaging signals with low and medium binding affinities to their sequence ends have lower fitness scores. When focusing on genomes with strong binding affinities for packaging signals at the ends, the pattern becomes apparent. Higher fitness values can be found in these genomes. This pattern shows that fitness outcomes are greatly influenced by the strength of these packaging signals at the genomic ends.

The agreement between the outcomes of SHAP and UMAP analysis emphasises the robustness of these findings. Both techniques emphasise the importance of packaging signal binding affinities, particularly at the ends of the genome, in determining 'Fitness' results. This convergence of multiple analytical methodologies gives an improved understanding of the complicated connection between genomic features and 'Fitness' values, strengthening our capacity to make accurate predictions and optimise genomic sequences for the intended results.

Chapter 5

Conclusion

The initial part of the study aimed to understand the complex connections between genomic data structures and their influence on fitness values. 1000 samples were randomly selected to represent the complete range of fitness values. Principal Component Analysis (PCA) was used to visualize the 12-dimensional vectors, but its limitations with non-linear data having a lattice structure made it difficult to explain the majority of variance in the data. Therefore Uniform Manifold Approximation and Projection (UMAP) was used to visualise the data. Genomes with weak and medium binding affinities at their ends were linked to lower fitness scores, while those with stronger affinities were linked to higher fitness. On the other hand, in the middle portions of genomes, weaker binding affinities were connected with higher fitness scores and stronger affinities with lower fitness values. A recurrent pattern emerged when the dataset's 531,441 genomes were included.

With a better understanding of the relationship between the genomes and fitness values, machine learning models were used to predict the fitness values. Data was split into train and test data, normalized, and evaluated using the test data to understand how well the model generalises to unseen data. Random Forest Regressor and Recurrent Neural Networks were used due to the non-linear nature of the data. The Random Forest model showed good performance metrics, with a low MAE and RMSE value, and R^2 value close to 1. The model's learning curve showed a continuous increasing pattern, and the feature importance showed that the packaging signals in the middle part of the genomes played a significant role in predicting fitness scores. However, the model used random guesses rather than learning the dodecahedral structure of the genome and it was found using the poor $\mathcal{F}$-statistic and high p-value.

The Recurrent Neural Network model with two layers improved the R^2 value significantly, with the implementation of LSTM acting as a memory for sequential data. The learning curve confirmed the model is not overfitting, with a $\mathcal{F}$-statistic and p-value of 13.1365 and $2.8964 \cdot 10^{-4}$. However, the R^2 learning curve didn't show a continuous increase in the R^2 value as the percentage of training data increased.

The second Recurrent Neural Network Model with 6 layers came out as the best-performing model among the three with a high R^2 score of 0.9904, really low MAE and RMSE values of

0.0057 and 0.01. The learning curve with training and validation loss ensured that the model was not overfitting. Moreover, the R^2 curve increased continuously as the percentage of training data increased. A high $\mathcal{F}$-statistic (116.8454) and a very low p-value ($3.1374 \cdot 10^{-27}$) showed that the model learned the complex pattern of the 12-dimensional vector dataset to predict fitness values for unseen data. Thus the analysis showed that a Recurrent Neural Network can easily learn complex data with greater accuracy.

The SHAP values obtained from the best-performing RNN Model showed that the packaging signals in the middle part of the genome are more significant than the ones in the end. Additionally, the binding affinities of the packaging signals are inversely related to the fitness values for these packaging signals. The packaging signals at the ends of the genomes have the least significance in the fitness value prediction. In addition, they have a positive influence on the prediction of fitness scores. Higher values of binding affinities at the ends of the genomes yield good fitness scores. This finding agrees with the UMAP results which showed that the fitness score and binding affinities of the packaging signals at the middle part of the genome are inversely related, and the ones at the ends were directly related.

There are certain limitations to this analysis. The modelling and analysis were done using a toy model of virus assembly and in real-world scenarios, the viruses have a more complex structure and evolutionary pattern. The performance of these models on real-world data is still questionable. Even though the patterns found in the toy set were interesting, these may not emerge in a real virus sample. While machine learning models demonstrated predictive abilities, their intrinsic complexity may necessitate additional refining to gain an improved knowledge of genome-fitness relationships. This opens future scope for further work employing real-world virus data in machine learning models. Those models can be used to predict the fitness of viruses as well as understand the underlying patterns in the virus. Once the patterns are known, vaccines and medications can be developed to curb the growth and the spread of the virus. Thus by mitigating the virus at the earlier stages of the virus detection itself, the health and well-being of the community can be improved. Thus this project aligns well with the United Nations Sustainable Development Goals (UNSDGs) of 2023 'Good Heath and Well-being' (Division, 2023).

Bibliography

Becht, E., L. McInnes, J. Healy, C.-A. Dutertre, I. W. Kwok, L. G. Ng, F. Ginhoux, and E. W. Newell (2019). Dimensionality reduction for visualizing single-cell data using umap. *Nature biotechnology 37*(1), 38–44.

Berrar, D. (2019). Cross-validation. In S. Ranganathan, M. Gribskov, K. Nakai, and C. Schönbach (Eds.), *Encyclopedia of Bioinformatics and Computational Biology*, pp. 542–545. Oxford: Academic Press.

Breiman, L. (2001). Random forests. *Machine learning 45*, 5–32.

Buskirk, T. D. (2018). Surveying the forests and sampling the trees: An overview of classification and regression trees and random forests with applications in survey research. *Survey Practice 11*(1).

Chicco, D., M. J. Warrens, and G. Jurman (2021). The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. *PeerJ Computer Science 7*, e623.

Dechant, P.-P. and Y.-H. He (2021a). Machine-learning a virus assembly fitness landscape. *Plos one 16*(5), e0250227.

Dechant, P.-P. and Y.-H. He (2021b). Machine-learning a virus assembly fitness landscape. Volume 16, pp. e0250227. Public Library of Science San Francisco, CA USA. Figure adapted from Dechant and He, https://doi.org/10.1371/journal.pone.0250227.g001, p. 2.

Dechant, P.-P. and R. Twarock (2021). Models of viral capsid symmetry as a driver of discovery in virology and nanotechnology. *The Biochemist 43*(1), 20–24.

Division, U. N. S. (2023). The sustainable development goals report 2023: Special edition. `https://unstats.un.org/sdgs/report/2023/`. [Accessed 03-09-2023].

Draganov, A. (2021). *Towards a common dimensionality reduction approach; Comparing PCA, TSNE, and UMAP through a cohesive framework*. Ph. D. thesis, George Mason University.

Frost, J. (2017). How F-tests work in Analysis of Variance (ANOVA) — statisticsbyjim.com. https://statisticsbyjim.com/anova/f-tests-anova/. [Accessed 02-09-2023].

Frost, J. (2019). Root mean square error (RMSE) — statisticsbyjim.com. https://statisticsbyjim.com/regression/root-mean-square-error-rmse/. [Accessed 28-08-2023].

Gelderblom, H. R. (1996). Structure and classification of viruses. *Medical Microbiology. 4th edition*.

Grömping, U. (2015). Variable importance in regression models. *Wiley interdisciplinary reviews: Computational statistics 7*(2), 137–152.

Hochreiter, S. and J. Schmidhuber (1997). Long short-term memory. *Neural computation 9*(8), 1735–1780.

Jaadi, Z. (2021). A Step-by-Step Explanation of Principal Component Analysis (PCA) — builtin.com. https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed 24-08-2023].

Jing, N., Z. Shi, Y. Hu, and J. Yuan (2022). Cross-sectional analysis and data-driven forecasting of confirmed covid-19 cases. *Applied Intelligence*, 1–16.

Kanyongo, G. Y. (2005). Determining the correct number of components to extract from a principal components analysis: a monte carlo study of the accuracy of the scree plot. *Journal of modern applied statistical methods 4*(1), 13.

Kassambara, A. and F. Mundt (2017). PCA - Principal Component Analysis Essentials - Articles - STHDA — sthda.com. http://www.sthda.com/english/articles/31-principal-component-methods-in-r-practical-guide/112-pca-principal-component-analysis-essentials/. [Accessed 24-08-2023].

Keshk, M., N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya (2023). An explainable deep learning-enabled intrusion detection framework in IoT networks. *Information Sciences 639*, 119000.

Lundberg, S. M. and S.-I. Lee (2017). A unified approach to interpreting model predictions. *Advances in neural information processing systems 30*.

Malato, G. (2021). Hyperparameter tuning. Grid search and random search — Your Data Teacher — yourdatateacher.com. https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/. [Accessed 24-08-2023].

Patel, N., E. Wroblewski, G. Leonov, S. E. Phillips, R. Tuma, R. Twarock, and P. G. Stockley (2017). Rewriting nature's assembly manual for a ssRNA virus. *Proceedings of the National Academy of Sciences 114*(46), 12255–12260.

Płoński, P. (2020). Random Forest Feature Importance Computed in 3 Ways with Python — mljar.com. `https://mljar.com/blog/feature-importance-in-random-forest/`. [Accessed 04-09-2023].

Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science 2*(6), 420.

Schmalen, P. (2020). Understand your data with principal component analysis (PCA) and discover underlying patterns — towardsdatascience.com. `https://towardsdatascience.com/understand-your-data-with-principle-component-analysis-pca-and-discover-underlying-patterns-d6cadb020939`. [Accessed 24-08-2023].

Schonlau, M. and R. Y. Zou (2020). The random forest algorithm for statistical learning. *The Stata Journal 20*(1), 3–29.

Twarock, R., R. J. Bingham, E. C. Dykeman, and P. G. Stockley (2018). A modelling paradigm for RNA virus assembly. *Current opinion in virology 31*, 74–81.

Twarock, R. and P. G. Stockley (2019). RNA-mediated virus assembly: mechanisms and consequences for viral evolution and therapy. *Annual Review of Biophysics 48*, 495–514.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*(10), 1550–1560.

Xenochristou, M. and Z. Kapelan (2020). An ensemble stacked model with bias correction for improved water demand forecasting. *Urban Water Journal 17*(3), 212–223.

Yao, Y. (2022). Data analysis on the computer intelligent stock prediction model based on lstm rnn and algorithm optimization. In *2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, pp. 480–485. IEEE.

Zlotnick, A. (1994). To build a virus capsid: an equilibrium model of the self assembly of polyhedral protein complexes. *Journal of molecular biology 241*(1), 59–67.