Karlsruhe Institute of Technology

Investigating the relationship between tree distance and model distance

Master's Thesis of

Xinyi Zhang

at the Department of Informatics
Institute of Theoretical Informatics

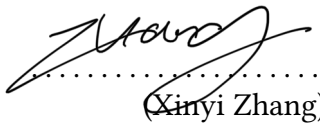
Reviewer:	Prof. Dr. Alexandros Stamatakis
Second reviewer:	Prof. Dr. Hartmut Prautzsch
Advisor:	M.Sc. Ben Bettisworth

01. September 2021 – 28. February 2022

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE



.....

(Xinyi Zhang)

Abstract

A Horizontal Gene Transfer (HGT) is the transfer of genetic information (i.e genes) between two unrelated species by means other than heredity (i.e. child parent inheritance). When investigating the evolutionary history of a species the presence of HGT events may lead to an incongruent gene tree and species tree, complicating phylogentic analysis. Therefore detecting HGT events is crucial in order to obtain a correct phylogenetic tree. An attempt to detect HGT events has been done in this work. We started by analyzing patristic tree distances and model-based evolutionary distances and we have not found any relation to HGT events. However, when we apply a convolutional method to the model-based evolutionary distances, we observe a distinct relationship between the data and a HGT event. In particular when there is a linear relationship between evolutionary distances and the position of convolution windows there may be HGT event happening. Using this observation we attempt to classify the convolution window distances with some machine learning methods. Nevertheless, the classification process is more challenging than anticipated. We present here our result and the difficulties we encountered.

Zusammenfassung

Ein horizontaler Gentransfer (HGT) bezeichnet die Übertragung genetischer Informationen (d.h. Gene) zwischen zwei nicht verwandten Arten, die auf andere Weise als durch Vererbung geschieht. Bei der Untersuchung der Evolutionsgeschichte kann das Vorhandensein von HGTs zu inkongruenten phylogenetischen Bäumen führen. Daher ist der Nachweis eines HGT entscheidend, um einen korrekten phylogenetischer Baum zu erhalten. In dieser Arbeit wurde ein Versuch unternommen, HGT-Ereignisse zu erkennen. Wir begannen mit der Analyse von patristischen Baumdistanzen und modellbasierten evolutionären Distanzen und haben keine Beziehung zu HGT-Ereignissen festgestellt. Wenn wir jedoch die Convolution-Window-Methode auf die modellbasierten evolutionären Distanzen anwenden, können wir eine gewisse Beziehung zum HGT beobachten. Insbesondere kann es zu HGTs kommen, wenn lineare Abhängigkeit zwischen den evolutionären Distanzen und der Positionen der Convolution-Windows besteht. Anhand dieser Beobachtung versuchen wir, die Convolution-Window-Distanzen mit einigen Methoden des maschinellen Lernens zu klassifizieren. Dennoch ist der Klassifizierungsprozess anspruchsvoller als erwartet. Hier stellen wir unser Ergebnis sowie die aufgetretenen Probleme vor.

Contents

List of Figures

List of Tables

1 Motivation and Contribution

The phylogenetic tree concept is used to illustrate the evolutionary relationships among different species. It is also known as the tree of life or simply called phylogeny. The first evolutionary tree was sketched by Charles Darwin in 1837 in his seminar work "On the Origin of Species. It was the first sketch of the tree of life and has ever since remained one of the central ideas of evolutionary biology. Today, the concept of a phylogenetic tree is widely used in studies on the evolution of species, including mapping the changes in the species' feature, which mught include DNA solutions, to analyze the historical path of evolution [1]. Additionally, this useful research tool also plays an important role when it comes to classification of metagenomics, especially for viruses and bacteria.

For historical and computational reasons, evolutionary relationships between species are usually assumed to be tree-like. This is to say, the tree does not contain any reticulations, or places where the two species merge again.

However, when phylogenetic trees are inferred from complete genome sequences, some discordance between the inferred phylogenetic tree and the actual true evolutionary history of the species can be observed. Apart from errors during the inference process, such discordance can also be caused by events such as Horizontal Gene Transfer (HGT) and Incomplete Lineage Sorting (ILS) [2]. Both, ILS and HGT events affect the evolutionary history of genes. For example the gene from other organisms can be passed to host organisms via HGT. The transferred gene has a different evolution history from the host. The Difference will cause confusion when we analyze the evolutionary history of the host organism leading to a incorrect phylogenetic tree.[3]

The goal of this thesis is to identify this process and detect incorrect trees by recognizing events such as HGT and ILS. Because the evolutionary history of genes and the evolutionary history of species are different in these cases, we check if the distance computed between taxa of a tree are in the line with the distance computed from the alignments only. The distance computed using the phylogenetic tree is denoted as tree distance or patristic distance.The model-based evolutionary distance represents then the distance between alignments. By comparing the two distances we may notice some anomalies which might represent evidence of the presence of ILS and HGT events. However, in our work we only simulate HGT events because ILS simulation is more complicated than HGT simulation. Furthermore ILS may also bring further complication to the simulated samples and sometimes it is very difficult to identify the impact of ILS events.

In this thesis, we develop HUGS(Horizontal event detection Using Generalized Statistics). HUGS is a tool that takes gene trees and per-gene alignments as inputs and infers

a prediction about the presence of HGT events. For the prediction algorithm, we deploy current state-of-the-art machine learning methods to attain better results. This tool will be expected to serve as guide to detect HGT events that can compromise the phylogenetic tree.

The next chapter (Chapter 2 introduces the background information and required definitions. Chapter 3 covers the comparison between tree distances and model-based evolutionary distances. It is followed by a naive sliding window approach (Chapter 4). Here the sliding window is applied to model-based evolutionary distances in order to obtain results with higher resolution. These results are then used for classification (Chapter 5). We attempt to classify the biological dataset into two groups. One group contains datasets where we observe one or more HGT events in the gene trees and the other comprises datasets is the opposite group where we do not observe any HGT events. Results are then presented in Chapter 6. Finally, we discuss the results and provide a brief outlook in Chapter 7.

2 Introduction

The patristic distance, this is to say the node distance which incorporates branch lengths, are derived directly from the phylogenetic tree, provided the tree has meaningful branch lengths. A model distance is calculated from from the comparison of 2 gene sequences. In the case that the model and the tree are correct, there should be a correlation between these two distances. In this thesis we will investigate these correlations and cases when this expected correlation is violated.

The remainder of this chapter is organized into two parts. In the first part (Section 2.1) we present the necessary background information and introduce the required definitions. In the second part (Section 2.2) we will discuss related work.

2.1 Theoretical background

In this section, a basic theoretical foundation is provided. We will first differentiate between gene trees and the species tree. This is followed by an introduction to Horizontal Gene Transfers and hybridization events and their consequences for phylogenetic inference. Finally, we will introduce tree distances and model-based evolutionary distances and how to calculate them.

2.1.1 Gene Trees and Species Trees

Both species trees and gene trees are specialized phylogenetic trees. Species trees describe the evolutionary relationships between species and the gene trees describe the evolutionary relationships for single genes. The species tree reflects the evolutionary history of a group of species. Each branch in a species tree represents an evolutionary step that relates the ancestor species and its offspring. On the other hand, a gene tree reflects the evolutionary history of a particular gene. The gene tree reflects the process of evolution at a local level. Events such as gene duplication, gene loss and gen transfer can influence the course of a gene's history making the evolutionary history represented by gene trees different from the species history. In fact, gene duplication is one of the most important mechanism of evolution [4]. Gene copies generated by duplication events are passed to the offspring creating numerous possibilities for the tree [5]. The relation between species tree and gene tree can be seen in Figure 2.1. In general, the history of a single gene is different from the history of the species and should not be treated as the species history. There are attempts that try to build a species tree from a set of gene trees and vice versa [6][7]. However, these methods always have very strict prerequisites that are in some cases difficult to achieve [7].

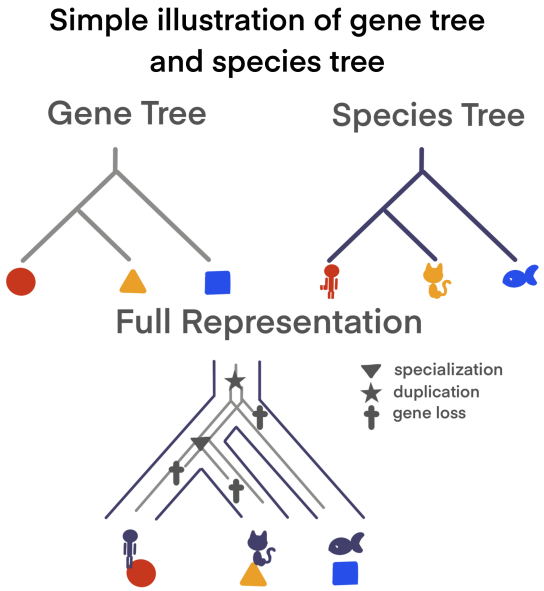


Figure 2.1: Simple illustration of gene tree and species tree: On the top left is a simple gene tree with three genes. On the top right is the corresponding species tree with three species: human, cat, and fish. On the bottom is the combination of the trees, the dark blue lines represent the species tree and the grey lines are the gene tree with some duplication, specialization, and gene loss events.

2.1.2 Horizontal Gene Transfer

Horizontal Gene Transfer (HGT) or Lateral Gene Transfer (here referred to as HGT) is the event that is used in this thesis to explain and simulate discordance between trees. The main source of HGT is the exchange of genetic material between different organisms of either the same or different species. Although HGT is more common in bacteria (e.g., transfer from one bacterial species to another), it can also have other more complex organisms as recipients, such as fungi, plants, and animals [8]. HGT is also important regarding human health as it affects the development, emergence and recurrence of diseases such as cancer, genetic-, metabolic-, and neurodegenerative disorders [9]. HGT often influence the inference of phylogenetic trees [10] as shown in Figure 2.2.

2.1.3 Hybridization

Apart from HGT, hybridization is another major cause of incongruence among gene and species trees. Both HGT and hybridization occurs in the nature, hybridization is a combination of two lineages that lead to a new species. When we include a node that was generated by hybridization event it do not follow the assumption of evolution from a common ancestor. In fact, some of its genes may not share any evolution history with other nodes of the phylogenetic tree.

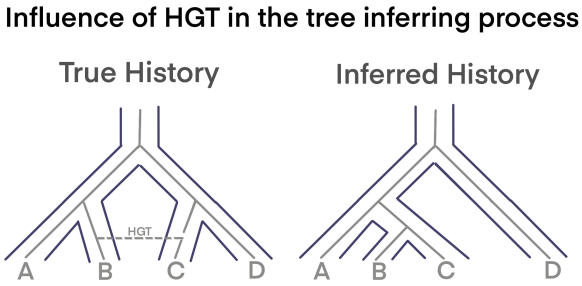


Figure 2.2: Influence of HGT in the tree inference process: Dark blue lines represent the species tree, grey lines represent the gene tree. On the left is the true evolutionary history. As we can see, a HGT event will yield the incorrectly inferred tree on the left.

2.1.4 Tree Distances

In this thesis, the tree or patristic distance is a pairwise distance defined as the length of the shortest path between pairs of taxa. The species trees and the gene trees which we use in this thesis have branch lengths that can be used to calculate tree or patristic distances. However, we are only interested in gene trees since species trees are in general, not observable. A species tree shows the evolutionary history of a set of species but we cannot use it to analyze the evolution history of genes and therefore we cannot use it for detecting HGT events that happened in gene level. Apart from the branch length we can also use the number of inner nodes to calculate a node distance, which is straightforward. The node distance may be sufficient for our needs because this thesis is focused on the structure of the tree rather than its other value.

2.1.5 Model-based Evolutionary Distances

The distance between sequences is normally used to compute distance matrices that are then used for inferring phylogenetic trees using method such as Maximum Likelihood (ML). It is defined as the mean expected number of substitutions per site. In our case, we only calculate the distance between two sequences. The simplest distance measure is to count the number of different sites and this distance is also referred to as the p-distance. However, this method usually underestimates the true number of substitutions. Even sites that remain invariant can be the result of multiple substitutions [11]. Time-continuous Markov Chain models can be used to address this issue. In these models, sites are assumed to evolve independently from each other and substitutions at each site are described by a Markov Chain. Each change of state of the Markov chain only depends on the current state, therefore each substitution depends only on the current state of the site. Besides the use of Markov Chain, other restrictions, such as setting relative substitution rates between nucleotides, may be added leading to different models. The distances that we obtain by using these models are then called model-based evolutionary distances. In this thesis we mainly use two models, the simplest one, JC69 (Jukes-Cantor 1969) and the most complex one in common use, GTR (General Time Reversible) .

2.1.5.1 JC69 (Jukes and Cantor 1969)

The JC69 model [12] is a very simple model with equal base frequencies $\pi = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$ and equal mutation rates λ for all nucleotide. Using q_{ij} to indicate the substitution rate from nucleotide i to nucleotide j, with $i, j = T, C, A$ or G the instantaneous substitution rate matrix is the following:

$$Q_{JC69} = q_{ij} = \begin{bmatrix} -3\lambda & \lambda & \lambda & \lambda \\ \lambda & -3\lambda & \lambda & \lambda \\ \lambda & \lambda & -3\lambda & \lambda \\ \lambda & \lambda & \lambda & -3\lambda \end{bmatrix} \tag{2.1}$$

The JC69 model distance is given by

$$d_{JC69} = -\frac{3}{4}(ln(1 - \frac{4}{3}p), \tag{2.2}$$

where p is the aforementioned p-distance. Note that if $p >= \frac{3}{4}$ the JC69 distance is not defined, as $1 - \frac{4}{3}p$ is negative in this case, and the logarithm of a negative number is not defined. This means that sufficiently different gene sequences may lead to errors in JC69 distance calculations.

2.1.5.2 GTR (Tavaré 1986)

The Generalised Time-Reversible (GTR) model of Tavaré 1986 [13] is a more complex model that uses variable base frequencies $\pi = (\pi_T, \pi_C, \pi_A, \pi_G)$. Additionally, the mutation rates are also different for each nucleotide transition pair and are denoted by a, b, c, d, e and f. Its substitution matrix is the following:

$$Q_{GTR} = q_{ij} = \begin{bmatrix} * & a\pi_C & b\pi_A & c\pi_G \\ a\pi_T & * & d\pi_A & e\pi_G \\ b\pi_T & d\pi_C & * & f\pi_G \\ c\pi_T & e\pi_C & f\pi_A & * \end{bmatrix} \tag{2.3}$$

Where * is whatever value such that each row should sum to 0. As GTR is more general model with more free parameters, it is suggested to estimate the GTR distance using the Maximum Likelihood method. However it is possible solve the problem analytically by diagonalizing Q_{GTR}: one eigenvalue of Q_{GTR} is 0, so the characteristic equation is a cubic equation which is solvable [11].

2.2 Related Work

In this section related works such as previous studies on HGT events are presented. In particular we will describe how HGT are detected in these works. Today the methods that identify HGT are divided into two groups, one sequence-composition related and the other one phylogenetic tree related [14].

The sequence-composition methods, or sometimes referred as parametric methods, search for sections of genome that are significantly different from the average genome regions. In particular, Guanine-Cytosine content and condon usage can be used as parameters [15]. Guanine-Cytosine content, or simply GC-content, is the percentage of Guanine (G) and Cytosine (C) in a DNA molecule. GC-content can be given for a section of the genome or for the entire genome. Condon usage refers to the difference of occurrence of synonymous condons in coding DNA. The major disadvantage of this method is that it relies too much on compositional pattern, this will lead to problem when the compared sequence share similar composition. Furthermore, the length of the transferred gene may be too short to reveal the differences between compared sequence [16].

The phylogenetic methods compare the different evolution history of involved genes and identify conflicting phylogenetic trees. Phylogenetic methods can be further divided into implicit methods and explicit methods. Explicit methods try to explicitly reconstruct and compare gene trees with their species tree [17]. On the other hand implicit methods compare evolutionary distances or sequence similarity [18]. In general implicit methods are simpler than explicit methods because they do not require tree inference. Explicit methods, on the other hand, can give more details for HGT events, such as when it happened. However, the conflicting evolution history can be the result of events other than HGT event. Phylogenetic methods also requires tree inference or Multiple Sequence Alignment which are relatively expensive[14]. Apart from the use of traditional phylogenetic tree, phylogenetic network can also be used in detecting HGT events [19]. Phylogenetic network is a modification of phylogenetic trees, where reticulation points are added. Reticulation points are points where two branches comes together to form a new species.

Parametric method and phylogenetic method can also be combined in order to get better results [20]. More related to our work, there are also previous attempt to detect HGT events using evolutionary distances such as DLIGHT [21]. Furthermore, there are approaches that uses machine learning methods, in particular deep residual neural network, to detect HGT events which gives good results [22].

3 Comparing Tree and Model-based Evolutionary Distances

In this chapter, we will describe in detail how we compared the patristic tree distance and model-based evolutionary distance. This is our first attempt in order to investigate the effect of HGT event on the tree structure. The patristic tree distance is calculated from the gene trees while the model-based evolutionary distance is calculated from the per-gene sequence alignments. We intend to find some connections that connect HGT events to the relation between the patristic tree distance and the model-based evolutionary distance. When there is a conflict between the distances, there may also be a HGT event taking place.

In Section 3.1 we describe the process of comparing patristic tree distance and model-based evolutionary distance. The following section (Section 3.2) covers the data simulation and the tools used during the simulation process. Section 3.3 describes how the patristic tree distance and the model-based evolutionary distance are calculated. Finally Section 3.4 describes how we represent the simulated data and the calculated distances.

3.1 Workflow

The section is divided into two parts. The first covers data simulation. We simulate the species tree and the gene trees using the software tool Simphy [23] and corresponding sequences using the simulator INDELible [24]. The second part describes the software that calculates and displays the distances on the simulated data. The main program, which is our contribution, is written in Python, while the simulation tools are implemented in C and C++.

Figure 3.1 shows the detailed workflow. There are five main modules:

1. Generate gene tree and species tree using Simphy
2. Generate sequences from the simulated tree via INDELible
3. Calculate patristic distances on the simulated gene trees
4. Calculate GTR or JC69 distances on the simulated sequences
5. Display the calculated distances as tables and diagrams

We start by separately simulating the tree and subsequently the sequence data. The simulators are invoked via the command line tool of the Python program, as sub-processes.

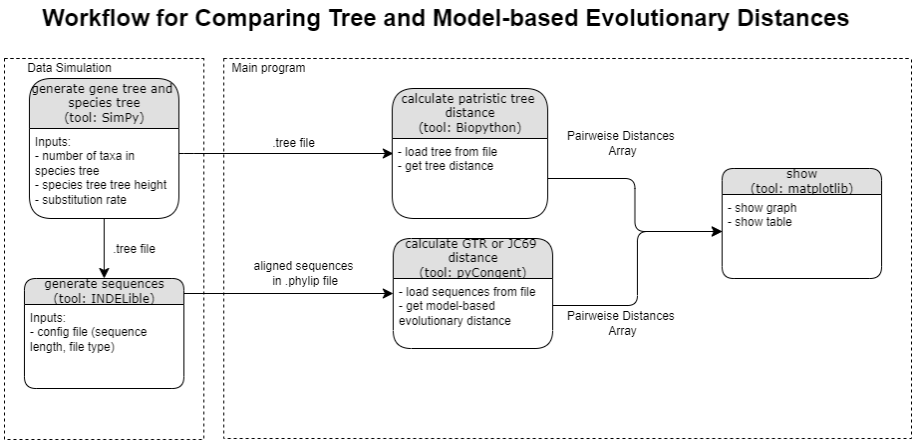


Figure 3.1: Workflow for comparing Tree and Model-based Evolutionary Distances

The simulated species tree and gene trees are in standard Newick tree format. The corresponding simulated sequences are in the standard PHYLIP format. These data are then used to calculate the patristic tree distances and the model-based evolutionary distances, respectively. The calculated distances are then displayed in the display module as tables or graphs. The display module also saves the calculated distance matrices and graphs in local files.

3.2 Data Simulation

Generating a sufficient amount of data is an important aspect of our work as a relatively large amount of data should be used to support our theory. Further, these data should be comparatively simple. Ideally each dataset should only have one HGT event. In this way, the influence of other events, such as gene duplication and ILS events, is reduced to a minimum and does not impact the results. For the sake of simplicity, we further only consider HGT events between two genes. In our work we only simulate two genes per species tree. Theoretically this should not affect our result, as we can also conduct the pairwise gene comparison for multiple genes. To reduce execution times of the analyses, we limit the length of our simulated sequences to 1000 nucleotides.

In the simulation process we first simulate one single species tree. The corresponding two gene trees are then simulated from this species tree. In the next step, these gene trees are used for simulating corresponding gene sequences. For the simulation of the species tree, the parameters used are: speciation rate, population size, substitution rate, generation time, number of taxa, and tree height. Number of taxa is set to five, population size should be as small as possible to avoid ILS events, for other parameters we consult the tree samples provided by the simulation tool in order to have reasonable values. The simulation of gene trees is conducted automatically with only the desired number of genes as input. The simulated trees are saved under the `src/data/Simphy/dataset_name` folder as `.TREE` files in Newick tree format.

The simulated gene trees are then used as inputs to the sequence simulator. Additional information such as sequence length and evolutionary models must also be provided to INDELible in order to correctly simulate gene sequences. This information is provided to the simulator via a configuration file. It is important to notice that INDELible only uses the gene trees to simulate the sequences, that is, it does not know their species tree. The simulated sequences are already aligned and are saved in separate PHYLIP files, one file per gene. The output sequences are saved in the same folder as the trees. In order to be able to use these sequences for distance calculations, corresponding gene alignment pairs must be concatenated into a single large PHYLIP file.

3.2.1 Simulation tools

For the simulation of trees we would like to have gene trees that are closely related to gene sequences. Furthermore, gene trees should also be related to their respective species tree. While the simulation of tree data can be conducted by manually generating simple trees with some realistic branch lengths, determining realistic branch length values was more difficult than expected. We started with widely used sequence simulators such as Seq-gen [25] and Dawg [26]. However, they do not support the simulation of tree shapes/topologies. Both simulators accept tree data as input, but they do not provide information on how a realistic tree should look like. Thus, in order to simulate tree shapes we used Simphy [23] which offers the possibility to simulate both, gene trees, and a species tree. For the sequence simulation we decided to use INDELible [26] because it offers a better integration with Simphy.

3.3 Distance Computation

Regarding computation of model-based evolutionary distance, the ability to deploy multi-processing is of great importance. During the computation of model-based evolutionary distance there is a large amount of data involved. Therefore the model-based evolutionary distance computation must support the possibility of multi-processing. We also must taken the possibility of future optimization into consideration. Thus, the model-based evolutionary distance computation must be easily extendable.

On the other hand, the computation of patristic tree distances is straight forward, as all information required is already contained in the Newick formatted tree. However, the patristic distance is not calculated directly from the original simulated newick tree. Instead, we load the Newick tree to construct a new datastructure that best fit the tree behaviour, using methods that are provided in the Biopython library. The distance is then calculated from that tree data structure. The pairwise patristic tree distances are saved in a symmetric distance matrix for future use. The distance matrix is defined as $D_{tree} = [d_{ij}]$ where $i, j \in \{1...n\}$, with n being the number of taxa. We also compute the node distances, as defined in Section 2.1, and save them in another symmetric matrix for future use. For more details on tree distances computation please refer to Section 2.1.

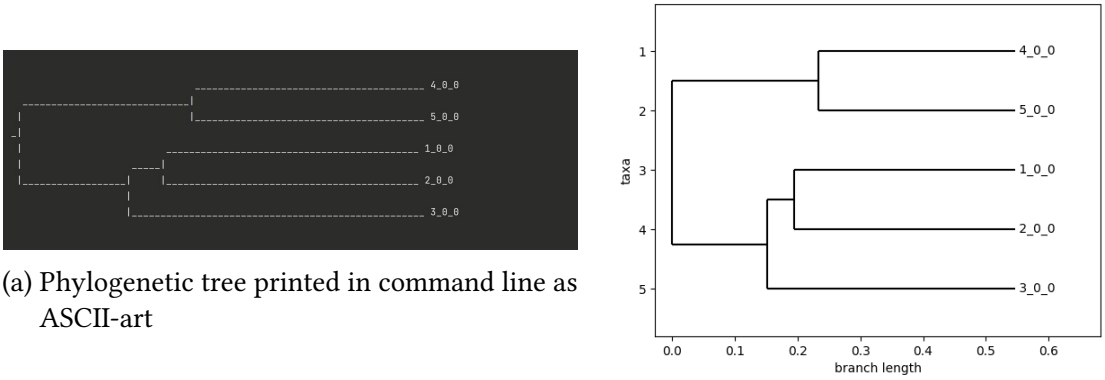
The model-based evolutionary distances are more difficult to compute. First we have to decide which evolutionary model to use. The first model we attempted was JC69 because of its simplicity and because of its computational efficiency. The alternative model we consider is the GTR model, as it is more general and might reveal details that we could miss under simple models. We use the already implemented methods in the PyCongent library to calculate these two model-based evolutionary distances. The calculations under the JC69 model are very fast. The GTR model requires about 1 minute for calculating all pairwise distance between five sequences each of length 1000. We decide to optimize the model-based evolutionary distance calculation using multi-threading. On a test hardware system using 4 threads will substantially reduce the computing time to under 1 minute for the distance calculation using GTR model, we have a latency speedup of about 3.13. Test hardware system has the following configuration: Inter(R) Core(TM) i7-4702HQ CPU (2.20 GHz) with 4 cores, 8 GiB of system memory. The results of the model-based evolutionary distance calculations are also saved in symmetric matrices. In addition, to facilitate the comparison of distances the ordering of the sequences corresponds to the ordering of the tree taxa, which are sorted alphabetically on their name. The distance matrix is $D_{model} = [d_{s_i s_j}]$ where s_i = is the sequence that corresponds to taxon i and s_j = is the sequence that corresponds to taxon j.

3.3.1 Distance calculation tools

The patristic tree distance calculation can be easily implemented from scratch. However, we need a parser which should be able to read and write tree and sequence related data formats. Biopython [27] is one of the most comprehensive python bioinformatics libraries. It has a well structured API that is well suited for our purpose. Therefore, we use it for all our basic data structures such as phylogenetic trees and aligned sequence data. For calculating model-based evolutionary distances, we need to search for other libraries. Biopython does not offer any functionality related to evolutionary models. To this end, we employ another library called PyCogent (Python COmparative GENomic Toolkit) [28] which implements numerous functions related to evolutionary models.

3.4 Visualization

Displaying is the last step of the workflow. After having computed the patristic tree distances and the model-based evolutionary distances, we need to compare and visualize them. The distance matrices can be visualized by using the tabulate module [29]. The tabulate module is not part of the standard Python library. However, it is widely used to display tables. An additional row and an additional column are added to specify the headers, which contain names of tree nodes or names of the sequences. For our data analysis, we decided to use a two dimensional histogram, where one column reflects one distance between pairs of taxa (for the patristic tree distance) or pair of sequences (for model-based evolutionary distances). To distinguish between the two types of distances we use different column colors. An example for the 2d histogram can be seen in Figure 3.3. In addition to the visualization of distances, we can also display the simulated tree. The tree



(a) Phylogenetic tree printed in command line as ASCII-art

(b) Phylogenetic tree as `.PNG` picture

Figure 3.2: Example of how the phylogenetic trees are shown

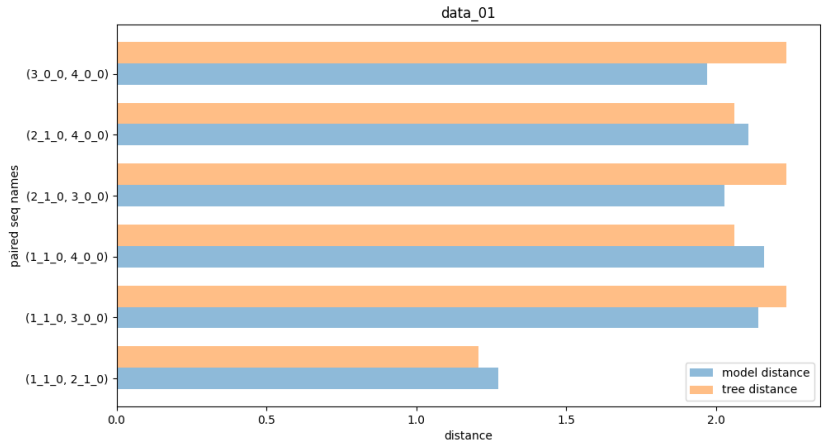


Figure 3.3: Example of a two dimensional histogram used to compare the patristic tree distances and model-base evolutionary distances

display can be done, both using the commandline as output (in form of an ASCII-art) or we can simply generate a picture of the tree. Both methods are available in the Biopython library, examples can be seen in 3.2.

After visualization we need to save our results. The Distance matrices are saved under `src/data/Simphy/dataset_namefolder/results/` as `.TXT` files. Tree pictures are saved under `src/data/Simphy/dataset_namefolder/pictures/` as `.PNG` files.

3.4.1 Visualization tools

The library used to display matrices is the tabulate module. Its main purpose is to display tables. It accepts lists and arrays as input and returns a printable object as output. The printable object can either be saved as file or printed on the screen. In order to compare

the matrices, histograms are used. Matplotlib is well suited for drawing different types of diagrams. Matplotlib is a library that is mainly used for data visualization in Python language, it can plot 2D and 3D graphs, it provides interactive tools and animation possibilities. Finally, we use the Python standard library for saving our results into local files.

4 Naive Convolution Window Approach

From the direct comparison between patristic tree distance and model-based evolutionary distance there was no evidence for HGT events. An important issue is that we only calculate the patristic distance for a single gene tree and then compare it to the model-based evolutionary distance of the respective gene sequence. It is difficult to detect HGT event when only one gene is taken into consideration. Theoretically, an incoherence between the two distances could be observed between distinct genes. A method that allows us to do such analyse is the use of convolution window. When using the convolution window we combine sequences from two gene into a single sequence, which is then used for convolution algorithm. The window will slide from a sequence to another giving different distances.

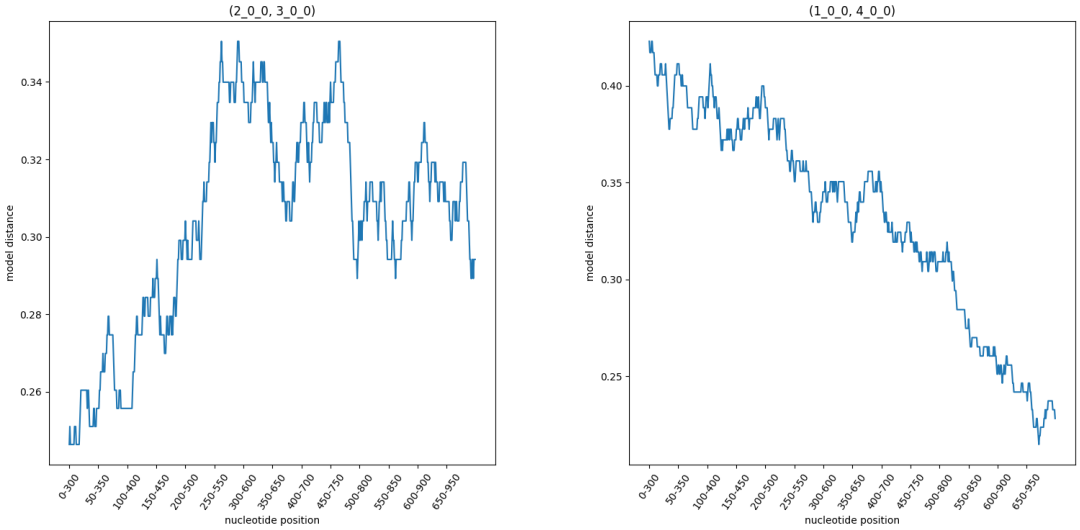
In this chapter, we will present how to deploy the convolution window approach to better analyse the relation between gene tree patristic and the model-based evolutionary distances. We will introduce convolution windows in Section 4.1 and outline how we use it in HUGS 4.2. There are some challenges which we present in Section 4.4.

4.1 Introduction to Convolution

A convolution algorithm is a straight-forward algorithm where a window with fixed size moves along the data to capture different portions of it. It is generally used for large datasets such as genomic sequences or image processing. In fact, the convolution window algorithm is a commonly used method for studying the properties of genome sequences. Instead of a single value, the convolution window provides a moving value for a window of a specific length that moves along the sequence. When the length of the window is too wide then the accuracy of the sliding window decreases, when the length is equal to the sequence length then the result will be the same as normal distance calculation. If the window length is too small there may be numerical problems, for further details please refer to Section 4.4. In our case, instead of a single model-based evolutionary distance, a convolution window approach provides a series of consecutive model-based evolutionary distances that change along with the window position.

4.2 Application and Discoveries

In HUGS, for an input sequence with a length 1000 nucleotides (500 per gene), we use a window of 300 nucleotides length and a stride length of 1. We choose a window size of 300 because this is the smallest window size that we can use without causing numerical



(a) Example of the convolution window approach visualization where HGT is not present

(b) Example of the convolution window approach visualization where HGT is not present

Figure 4.1: Example of the convolution window approach visualization using the JC69 model: on the top are the names of the compared sequences, on the x-axis is the distance computed for every window position, and on the y-axis is the start and end position of the window

errors during the calculation, for details please refer to Section 4.4. The implementation of the convolution window relies on the original distance calculation methods, presented previously in Section 3.3. To achieve this, we pre-process the the data. Instead of a single sequence with a length of 1000 nucleotides, we break it into a list of sequences of length 300. This list of shorter sequences is then used for model-based evolutionary distance calculation. The result of the distance calculation is no longer a single distance matrix, but a list of distance matrices, each corresponding to a different convolution window position.

As before, the convolution window results are saved in `.TXT` files in the folder `src/data/Simphy/dataset_namefolder/results/`. However, the visualization method used for the convolution window results is different. Before, we plotted all the pairwise distances in a single graph. For the convolution window, we use a separate graph for each pair of sequences. In each graph, we plot the model-based evolutionary distance obtained on the y-axis, while on the x-axis we display the positions of the convolution window. An example such a graph is shown in Figure 4.1.

In the graphs that we generated for the convolution windows, we can observe some patterns. If no HGT events occur, the values of the convolution window distance graphs are likely to be uncorrelated. In particular the distances (on the y axis) are uncorrelated with the convolution window position (on the x axis). If HGT events do occur and we observe the distance between the transferred taxon and other taxa, we can observe a correlation between the distance value and the window position. The distance will either

gradually decrease with the window movement, or it will gradually increase. Figure 4.1a provides an example for an uncorrelated plot while Figure 4.1b provides an example for a correlated plot.

This observation indicates that the the convolution window distances can be used for detecting HGT events. Since the convolution windows distance approach appears to be more promising than the comparison between patristic tree distances and model-based evolutionary distances, we decided to use the convolution window approach for further experiments. It is important to note that the convolution window distance is the model-based evolutionary distance calculated using convolution window algorithm. It does not use the patristic tree distance. Therefore, using exclusively the convolution window distance also means that we disregard patristic tree distances for further experiments.

4.3 Linear Regression

In order to assess the utility of our discovery and to further explore the observed property, we use the Linear Regression on the convolution window distance points to obtain a line that best fit the distance points. Linear regression is a mathematical approach for modeling the relationship between two numerical values. In our case it finds a line that best fits the points of the convolution window distance graph. In particular it uses least square method to minimize the squared error between the line and the points. The squared error is defined here as the sum of the square of distances from each point to the line. In order to better compare different results, we normalized the data and we calculated the deviation of the points from the line and the slope of the line itself. The normalization is done by mapping sliding window distances, which are positive real numbers, into a range between 0 and 1. An example of the linear regression used on the convolution window data can be seen in Figure 4.2. We compared slope and deviation on different graphs and decided that these two calculated values can be used as parameters that well describe the convolution window distance graph. We calculated the deviation and the slope for all convolution window distances and saved them as two dimensional vectors. In order to calculate linear regression on our data, we use the Sklearn library [30], which is a widely used Python machine learning library.

4.4 Challenges

There are few challenges when computing the convolution window distance for pair of sequences. One challenge is related to the computation time. For a sequence length of 1000 and a window size of 300, we need to compute 700 model-based distances. Therefore, we us a multi-threaded calculation where multiple distances at distinct convolution window positions are calculated in parallel. We calculate distances only once and we save them in local files. When data are needed, they will be simply loaded from local files. Since we will reuse the same dataset multiple times this approach will reduce the computing time in the future.

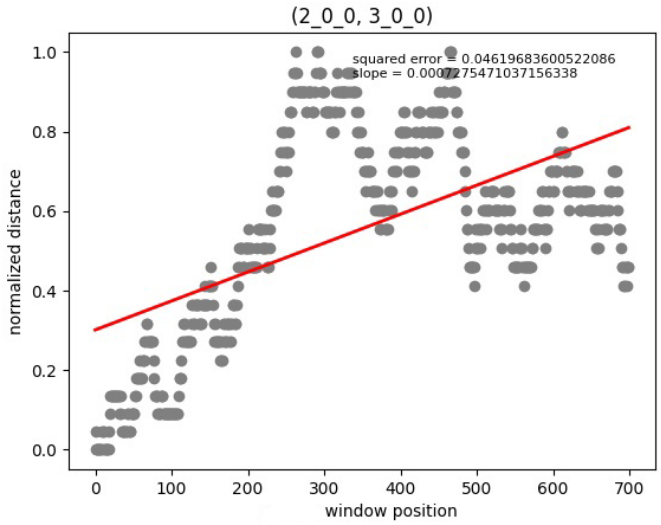
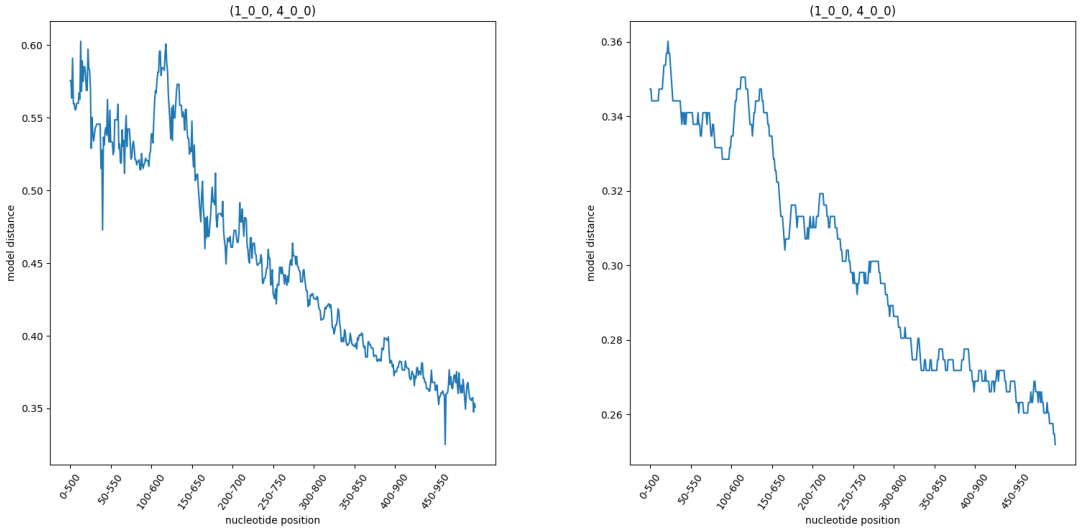


Figure 4.2: Example of linear regression on convolution window data: on the top are the names of the compared sequences, on the x-axis is the normalized distance and on the y-axis is the starting position of the nucleotide

Another major issue encountered were the numerical problems during the distance calculation. Since we use a convolution window, the input length for the distance calculation decreases from 1000 to the size of the convolution window. This leads to numerical problems when the size of the convolution window is too small. During the distance calculation if the input sequence is too short and there are too many substitutions, the p-distance (defined in Section 2.1.5) may be larger than the limit value leading to logarithm of negative number. According to our tests a convolution window smaller than 300 will lead to errors. This is also why we decided to adopt a window of size 300. Interestingly, the distances calculated using the JC69 model seem to be more stable than the distances calculated using the GTR model. In particular, in order to be correctly calculated the distance calculation using the GTR model requires a larger window size than the calculation using JC69 model. A possible explanation for this lays in the calculation process for the GTR model. During the calculation of GTR distance, numerical methods are used. Often the average of observed nucleotide frequencies is used to estimate the base frequencies. However the value may fluctuate too much when the length is too small. Furthermore, when a base frequency is zero it will cause the logarithm of a negative number. When both are correctly calculated, the two models do not show substantial differences in the results. However, the GTR model requires consistently more time to be calculated, on average 3 time slower than the JC69 model. We therefore decided to use the JC69 model for our convolution window distance calculations. A Comparison between the distance calculated using the two models can be seen in Figure 4.3.



(a) convolution window distance calculated using GTR model

(b) convolution window distance calculated using JC69 model

Figure 4.3: Comparison between the JC69 model and GTR model distances: The two graphs show the convolution window distance between the same pair of sequences. On the left is the distance calculated using GTR, on the right the distance calculated using the JC69 model. In both case the window size is 500. Because of numerical stability we choose a window of 500, a window of size 300 would cause possible numerical instability during the calculation involving GTR model. On avarage 1 out of 3 from the GTR distance calculation gives results with such fluctuation.

5 Classification of convolution window Distances

In this chapter we will describe how we classify convolution window distances. The main features that we use for the classification are the two resulting values from the linear regression (see Section 4.3 for details). We choose to model this problem as a binary classification where we decide if gene transfer occurs based on the convolution window distances. In particular, given a set of convolution window distance between pairs of sequences we decide if one or more pairs of sequences contains evidence of HGT events.

In the following, we will first describe the workflow of the classification process in Section 5.1. The datasets that are used in the classification are described in Section 5.2. In particular, we will describe how they were generated. The data structures that we used are described in Section 5.3. Section 5.4 covers the different machine learning methods that are used for the classification as well as their strengths and weaknesses. The different types of data visualization are described in Section 5.5.

5.1 Workflow

The workflow is divided into six modules, each with specific functions. The specific inputs and outputs of the modules are outlined in Figure 5.1. Here is an overview of the six modules:

1. **Generate Sequence Data**: takes the manually built trees and simulates sequences on them with the INDELible tool.

2. **Calculate convolution window Distance**: takes the simulated sequences and calculates the convolution windows distances using functions from PyCongent library.

3. **Linear Regression**: takes the calculated convolution window distances and calculates the linear regression line together with its slope and the deviation of the distances from the line, the deviation can also be referred as squared error.

4. **Data Preparation**: takes the computed slope and the deviation and builds training and test datasets using the Pandas library [31]. Pandas is a powerful library for data analysis in Python language. We use the library to manage data that are needed for the classification.

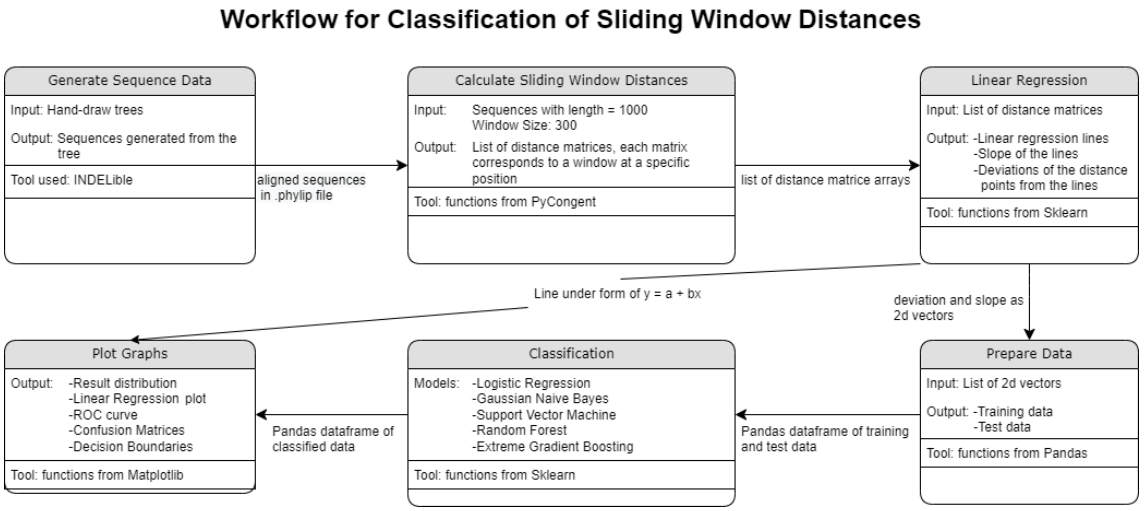


Figure 5.1: Workflow for the Classification of convolution window Distances

5. **Classification**: is a wrapper for the classification algorithms. It can use different machine learning models to classify the convolution window data. It uses functions from the Sklearn library [30]. Sklearn is a powerful library that provides various implementation of machine learning methods in python language.

6. **Plot Graphs**: visualizes different kinds of data using the Matplotlib library.

The first part of the workflow is very similar to the one described in Section 3.1. First, the gene trees are built manually, by modifying previously simulated trees. Since some of the classification methods that we used are sensitive to noise and in order to reduce the noise, we decided to build trees by hand. These manually modified trees are then used to simulate sequences along them. The produced simulated datasets are then used for distance calculations. The distances are calculated using the convolution window approach under the JC69 model. The results are then used for linear regression, where a regression line is computed. Furthermore, the deviation (in the form of squared error) of the convolution window distances to the line are calculated. The slope of the line and deviation need to be available before they can be used as features for the classification step. Thus, the slopes and deviations are saved as a list of two dimensional arrays. This list is then split into training data and test data. The classification is done using different machine learning models such as: Logistic Regression, Gaussian Naive Bayes, Support Vector Machine, Random Forest, and Extreme Gradient Boosting. Finally, the results of the classification are displayed using distinct graphical representations: result distribution, ROC curve, confusion matrices, decision boundaries. Also, the linear regression line can be plotted, as the regression line was calculated previously during the linear regression step.

5.2 Datasets

In contrast to the approach in Section 3.2, the gene trees that we use are not completely synthetic. We first simulate simple trees, then, we modify these tree by changing the branch length(s) or by moving branches/taxa. The gene trees generated in this way are simpler than the simulated gene trees, that is, the only HGT event that is the one we inserted by manually modifying the tree. In this way there will be less noise when we train the models. However, there are also problems related to the manually built trees, which we will discuss in detail in Chapter 7.

We use sequences with a total length of 1000 nucleotides (500 for each gene). The convolution window size is set to 300 nucleotides. Sequences are generated from 70 different pairs of gene trees with 4 or 5 taxa per tree. In total we have 140 sequences generated from manually built trees. Not all sets contain HGT events, some sets do not contain any events. The sequence simulation is conducted with INDELible (see Section 3.2.1 for details). Apart from the input trees, the remaining simulation steps are identical to Section 3.2.

After computing the convolution window distances, the dataset that will be used for the classification is built from the linear regression of the convolution window results. For each pair of distances, we compute the convolution window followed by the linear regression to determine the deviation of points from the regression line. The dataset is a list of two dimensional vectors, with the first entry of the vector representing the deviation and the second entry the slope. For the training datasets, each entry is also be marked with a classification marking, 1 for the correlated data (where HGT occurs) and 0 for the uncorrelated data.

5.3 Datastructures

In order to orchestrate the input and output of different types of data, we have built two classes: the Dataset and the Dataframe class. A class-diagram of the two classes is shown in Figure 5.2. Each Dataset entity corresponds to sequence data simulated on a single, specific pair of gene trees. The Dataset class contains simulation related information such as: `id, sequences, seq_length, g_trees, s_tree, taxa_names` and `taxa_numbers`. In detail, `id` is the identifier of the dataset, `sequences` saves the sequences, `seq_length` is the length of the sequences, `g_trees` saves the two gene trees, `s_tree` is the corresponding species tree, `taxa_names` is the list with name of taxa and `taxa_numbers` is the number of taxa. It also saves information related to distance calculations such as: `model, model_distance, sliding_window_distance_list` and `tree_distance`. In detail, `model` is the model used to calculate evolutionary distance, `model_distance` is an matrix that saves the calculated model-based evolutionary distances (it is calculated as the average of sliding window distances) and `sliding_window_distance_list` is a list of distance matrix. It offers methods for distance calculation, methods to read from and write to local files, and methods to visualize the convolution window distances. On the other hand, the Dataframe class is used to manage information that is needed during the classification. The Dataframe

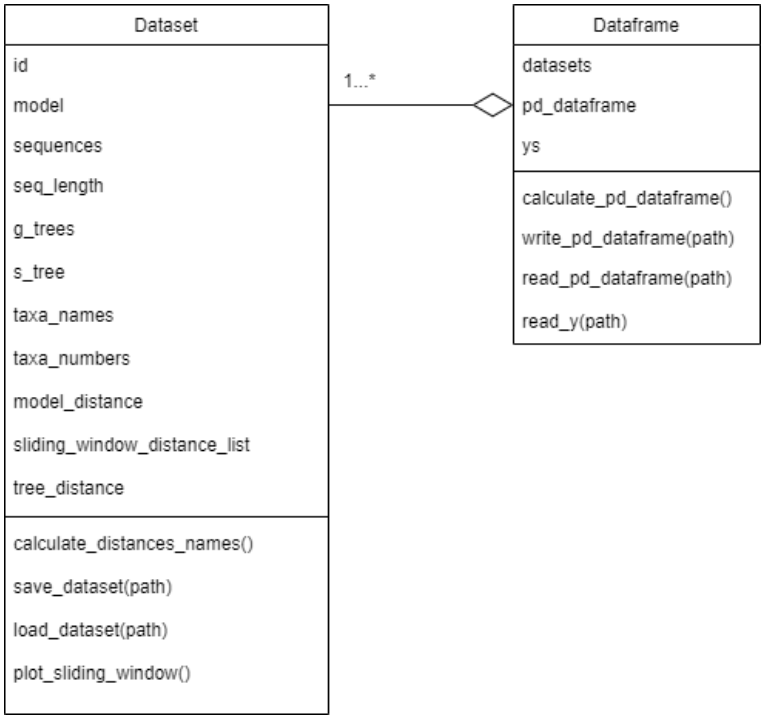


Figure 5.2: Class Diagram for data organization

class contains a list of all Dataset entities. It also contains a `pd_dataframe` which is a Pandas Dataframe that can be directly used as input for classification and that can be exported as `.CSV` file. The classification marking is saved under `ys`. The classification marking are needed during the training phase. This class offers methods for writing and reading `.CSV` files.

5.4 Machine Learning Methods Used

For the classification process, we compared results obtained from different machine learning methods: Logistic Regression, Gaussian Naive Bayes, Support Vector Machine, Random Forest, Extreme Gradient Boosting. Almost all models are implemented in the Sklearn library [30]. The recently introduced Extreme Gradient Boosting model is implemented in XGBoost [32], which is an extension of Sklearn. In the following sections, we will shortly describe these machine learning models. A summary of the strengths and weaknesses of each model is provided in Table 5.1.

5.4.1 Logistic Regression

Logistic Regression is a regression algorithm that can be used for binary classification. It is one of the simplest binary classification algorithms. It is very similar to linear regression, but instead of finding the best fitting line, it fits the logistic function. The logistic function is a sigmoid function that maps real values to a range between 1 and 0, in the classification problem of the output is more or equal than 0.5 then it is classified to 0, otherwise it is

Machine Learning Models	Advantages	Disadvantages
Logistic Regression	simple to implement, fast at classifying	only supports linear decision boundaries, assumes linear relationship between variables
Gaussian Naive Bayes	easy training process, works well on large amounts of data	its assumptions about the data not always true, relative simple decision boundary
Support Vector Machine	can handle non-linear data with relative simple implementation with respect to methods such as Random Forest and Gradient Boosting	sensitive to noise and data overlaps
Random Forest	can handle a large number of features (immune to curse of dimensions)	requires more time and computing power during training phase because of large number of decision trees, sensitive to over-fitting
Extreme Gradient Boosting	often better accuracy than other algorithms	requires even more time and computing power during training phase because of the decision tree building process, sensitive to over-fitting

Table 5.1: Advantages and Disadvantages of Machine Learning Methods Used in HUGS

classified to 0. In our case 1 is for the data that contains an HGT event while 0 is for data without HGT events. During the training process, the sigmoid function that best approximate the training data is found. In the respective classification process, the data are classified using the trained sigmoid function. The key advantage of Logistic Regression is that it is fast and easy to implement. This is also the main reason that we chose Logistic Regression as one of the first models to try.

5.4.2 Gaussian Naive Bayes

Gaussian Naive Bayes is a variation of the Naive Bayes classification algorithm which is based on Bayes' Theorem and a strong assumption that the features are independent, in our case the features are the slope of the linear regression line and the deviation of the datapoints from the line. Gaussian Naive Bayes makes an additional assumption that the data involved follows a Gaussian distribution. In particular, each class follows Gaussian distribution, in our case the two classes are: class with data that contain HGT event and class with data that do not contain HGT events. Different from other machine learning methods, during the training phase there are not any parameter to be fit, we calculate only the standard deviation σ and the mean of the points within μ the two classes from the training data. During the classification the Probability of for the new input sample x can be defined using the previous calculated mean and standard deviation:

$$P(x|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp(-\frac{(x-\mu_y)^2}{2\sigma_y^2}). \tag{5.1}$$

Where y is one of the two classes, in our case the input x is the pair of deviation and slope value. In order to classify the data the z-score of each input points from each class are calculated. The z-score from one class is defined to be the distance from the mean of that class divided by the standard deviation of the class, a picture explaining the z-score can be seen in Figure 5.3. Each point is then classified according to their z-scores, the point will be assigned to the class to which it has the smallest z-score.

One of the main weaknesses of Gaussian Naive Bayes are the assumptions that it makes. Real data often do not satisfy its assumptions. In our case, the independence assumption between features is satisfied, because the slope and the deviation of points are independent. But we cannot prove that our data follow the Gaussian distribution, on the contrary, from the distribution plot in Section 6.3 we can see that the data pattern is not typical of Gaussian distribution.

5.4.3 Support Vector Machines

Support Vector Machines, or shortly SVMs, are also a comparatively straight-forward machine learning algorithm. One of the key feature of SVMs is that they work in feature space, instead of directly on the data. SVMs analyze the points in feature space and find a line (in two dimensions) or a hyper plane (in higher dimensions) that best splits the two (or more) classes. The lines or hyper planes are better known as decision boundaries. Using

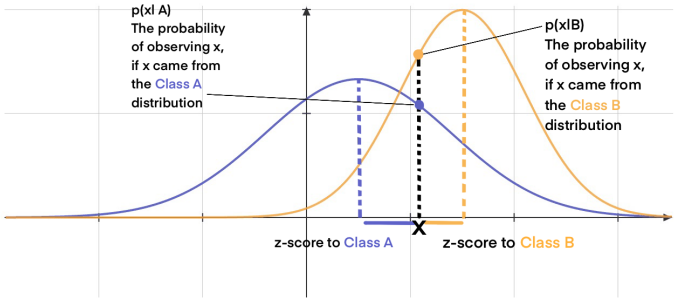


Figure 5.3: Illustration of how Gaussian Naive Bayes work

SVMs we try to maximize the margin between decision boundaries and data points in feature space. It is important to note that the decision boundary can assume different forms, it does not have to be a simple line. We can handle non linear data by applying a kernel technique . In short in the kernel technique we can use different mappings to map their inputs into high-dimensional feature spaces in order to make the data from different class easier to be separated. These mappings are generally called kernel. Popular kernels include: Polynomial Kernel, Gaussian Kernel, Radial Basis Function (RBF), Laplace RBF Kernel, Sigmoid Kernel and Anove RBF Kernel. When kernel technique is applied, the shape of decision boundary changes to match. However, here we only used a linear kernel because other simple kernels did not improve the results substantially and more complicated kernels are very difficult to define for our dataset. In order to define complicated kernels we have to study carefully our data and find or build the best mapping for it.

During the training phase we try to find the best decision boundary for our data. In order to classify a data point, we simply compute which side of the decision boundary the point is on. In this case, the decision boundary is just a line, so the computation is simple.

5.4.4 Random Forest

Before introducing the random forest approach, we need to explain what a decision tree is. Decision trees are a classification algorithm which works in a similar way to a flow chart. After a series of questions, the input samples can be classified based on the answers to these questions. In particular, a tree can be built by splitting the training data into subsets based on an attribute value test. This process is repeated on each derived subset in a recursive manner, each split add a new layer of nodes. The recursion is completed when target value is reached. The key idea behind the random forest algorithm is the wisdom of crowds. In our context it means: when a large number of relatively uncorrelated trees operates as a committee they will outperform any of the individual member tree. A random forest uses a large number of decision trees that are built independently. Each individual tree initially reaches an independent classification. The result of a random forest algorithm is then obtained by a majority vote from the trees. During the training phase, the decision trees are built from the training samples. These trees are then used for classification of the actual data.

5.4.5 Extreme Gradient Boosting

Extreme Gradient Boosting (XGBoost [32]) is a specific modification of the gradient boosting algorithm. The word extreme refers to the fact that it pushes the limits of computational power. The basic intuition behind boosting is to combine weak learners to attain improved performance. In boosting algorithm weak learners are defined as classifier that perform slightly better than random classifier. In our case, weak learners are the decision trees. During the training phase, decision trees are built one after another. During this iterative process, we attempt to reduce the miss-classification rates of the trees. In general, extreme gradient boosting performs better than the Random Forest algorithm, since theoretically its trees are better than the ones in a Random Forest.

5.5 Visualization

For data visualization we use different types of graphs. They are constructed using the Matplotlib library [33]. For comparing the results from the distinct machine learning methods, we generated ROC curves, confusion matrices, and decision boundary plots. Furthermore, other graphs such as linear regression lines and result distributions can also be visualized. Details about these graphs are presented in the following.

5.5.1 ROC Curve

ROC curve stands for receiver operating characteristic curve and it describes the trade-off between true positive rate (on the y axis) as well as the false positive rate (on the x axis). An ROC curve is perhaps the most common way to analyze the performance of a specific machine learning classifier. The true positive rate (TPR) is related to the sensitivity of the machine learning method. It is defined as $TPR = \frac{TP}{TP+FN}$. Where TP stands for True Positive and is the number of samples that are classified as positive and that are truly positive. FN stands for False Negative and is the number of samples that are classified as negative but are positive. TP and FN together forms the set of positive sample. The false positive rate (FPR) is related to the specificity, in particular it is equals to $1 - specificity$. FPR is defined as $FPR = \frac{FP}{FP+TN}$, where FP stands for False Positive and indicates the samples that are negative but are classified as positive, TN is True Negative and indicates samples that are classified as negative and are negative.

For a totally random classifier, the ROC curve will be a straight line with the Area Under the Curve (AUC) being exactly 0.5. For machine learning classifiers, the AUC should be at least 0.7 in order to be usable [34]. For analyzing the performance of an algorithm, the bigger the AUC the better is the algorithm. In particular AUC is directly connected to the ability of a classifier to do the correct classification, for example an AUC of 0 means that the classifier did all the predictions wrong and an AUC of 1 means that the classifier did all the predictions correctly. An example for the ROC curve can be seen in Figure 5.4.

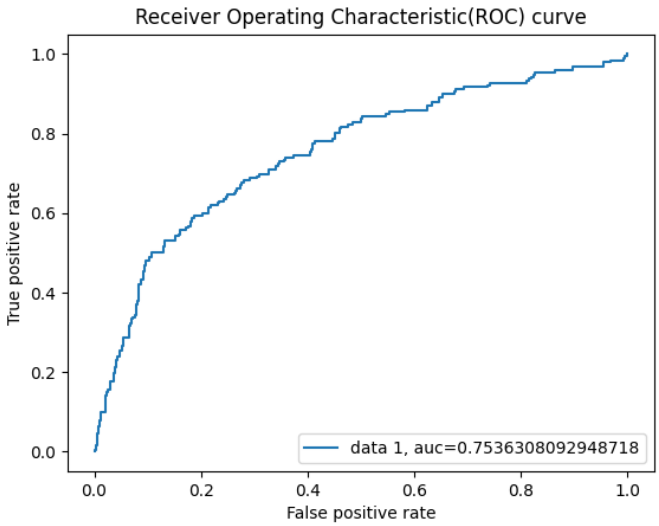


Figure 5.4: Example of ROC curve

5.5.2 Confusion Matrix

Confusion matrix, or error matrix, is a relatively simple and straight forward way of visualizing the results of classification methods. It is a 2x2 matrix with four fields: True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN). An example for the confusion matrix can be seen in Figure 5.5. The numbers in these fields represent the number of samples in the respective categories. Hence, a confusion matrix displays where the main errors of the classifier are.

An ideal classifier only has TP and TN, the other fields are 0. However, this is in general not the case. Depending on the the usage of the classifier, some may tolerate a relatively high FP rate and some may tolerate a high FN rate. In our case, we are more interested in FP values.

5.5.3 Decision Boundary Plots

Decision boundaries are one of the best ways to represent distinct characteristics of different machine learning classifier. Since each machine learning method has its own characteristic decision boundary. The decision boundary plot is created by drawing a line (or a plane in three dimensional space) in feature space. This line divides the data points into two classes. Depending on the type of classifier, the line can be straight or a curve. In extreme cases, the boundary is not represented as a single line. From the decision boundary plot, we can observe how the classifier would classify a particular dataset. Additionally, we may also observe possible over-fitting when the boundary is too fragmented as shown in Figure 5.6.

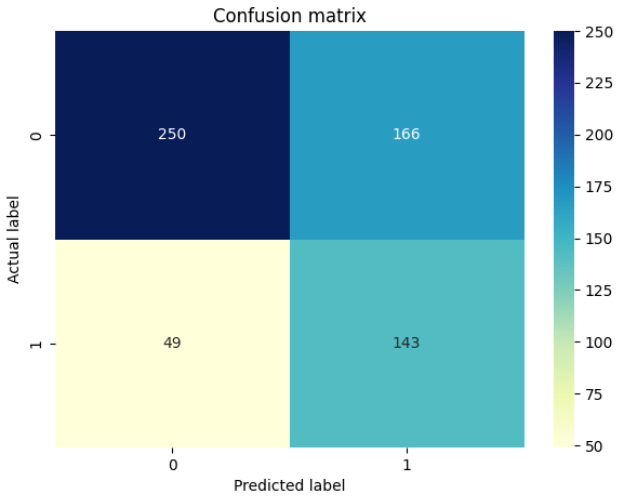


Figure 5.5: Example of Confusion Matrix

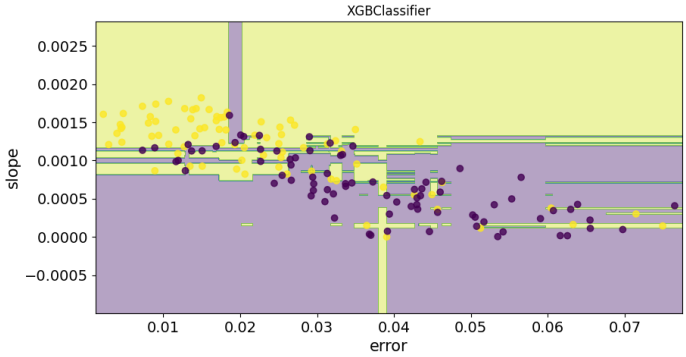


Figure 5.6: Illustration of how Gaussian Naive Bayes work

5.5.4 Result Distribution

A result distribution is a single graph that visualizes all the data points from the dataset along with their true classification. By comparing the result distribution to the decision boundary plots, we can see which points are mis-classified by each classifier and how far they are from the decision boundaries. Hopefully this information will help us adjusting the classifier by trying to bring the mis-classified sample to the other part of the boundary.

5.5.5 Linear Regression Plot

A linear regression plot is a visualization method that we used to display previously calculated linear regressions line that best fit the sliding window distance. Please refer to Section 4.2 for linear regression computations. In the plot, the data points are displayed together with the line and the deviation (or squared error). The slope of the line is displayed as a real number for better comparison. By checking the linear regression line for the

mis-classified samples, we may be able to find an explanation for the mis-classification and try to improve the classifier.

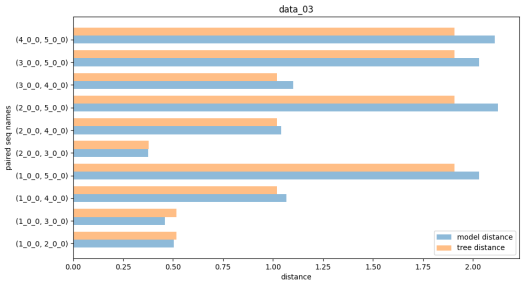
6 Results

In this chapter we will describe the result from our experiments and tests. We will start from the distances that we obtained from comparing patristic tree distance and model-based evolutionary distances in Section 6.1. Then we will show examples of convolution window distances and their linear regression plots in Section 6.2. Finally we will describe various classification in Section 6.3. Because of the large number of pairwise distances and the limited space, we will choose the most representative data for discussions and analysis. All sequences and tree related data, calculated values and pictures can be found under the folder `src/data/Simphy/`.

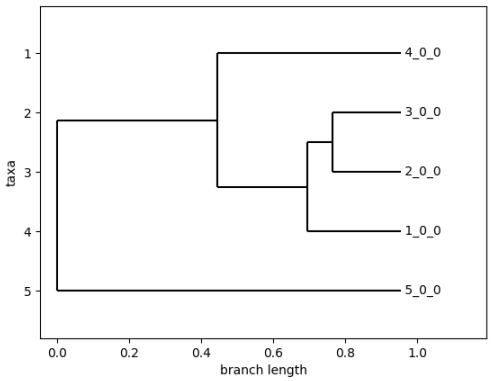
6.1 Tree and Model-Based Evolutionary Distances

In this section we will present a series of sequences with their respective gene trees, in particular we will visualize the trees and the comparison between patristic tree distance and model-based evolutionary distance.

Figure 6.1 is an example of comparison result between patristic tree difference and model-based evolutionary distance. On the left of the figure we have the histogram that compare the two distances and on the right we have the corresponding gene tree. We are comparing single gene trees with their respective sequences, therefore, it should not have any discordance. This result fits our theory. On the other hand, Figure 6.2 does not fit our theory, however we could not find any explanation for the discordance. Figure 6.3 shows a numerical problem during the distance calculation. The distance values of 20 is caused by numerical errors. When there is a logarithm of a very small number the output is set to 20. This type of results are discarded because it will cause seriously biasing results.

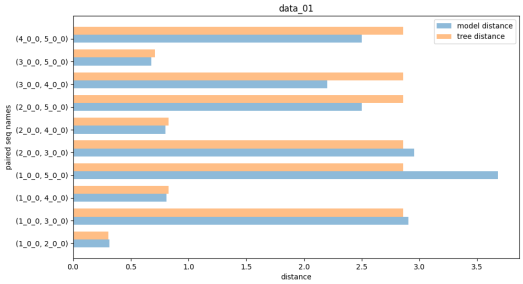


(a) comparison between patristic tree distance and model based evolutionary distance

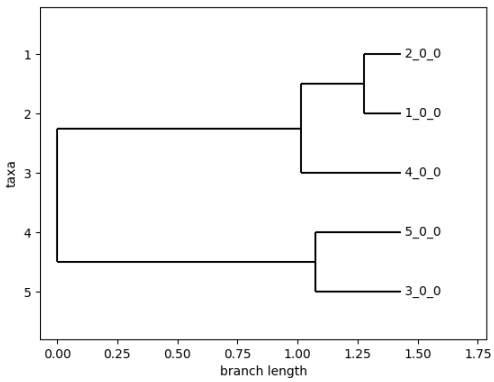


(b) respective gene tree with five taxa

Figure 6.1: From the comparison plot in 6.1a we can see that there is no large difference between patristic tree distance and model-based evolutionary distance. This result fits to our theory, since we are comparing single gene trees with their respective sequences and it should not have any discordance.

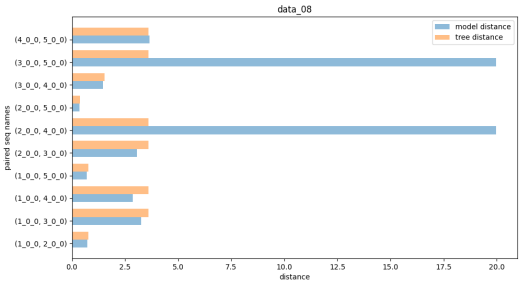


(a) comparison between patristic tree distance and model based evolutionary distance

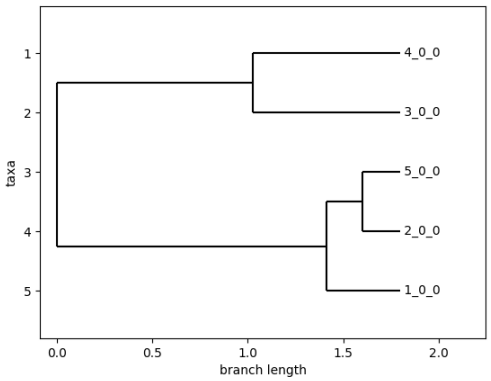


(b) respective gene tree with five taxa

Figure 6.2: In the comparison plot 6.2a we observe some discordance between the patristic tree distance and model based evolutionary distance. We could not find any explanation for the discordance.



(a) comparison between patristic tree distance and model based evolutionary distance



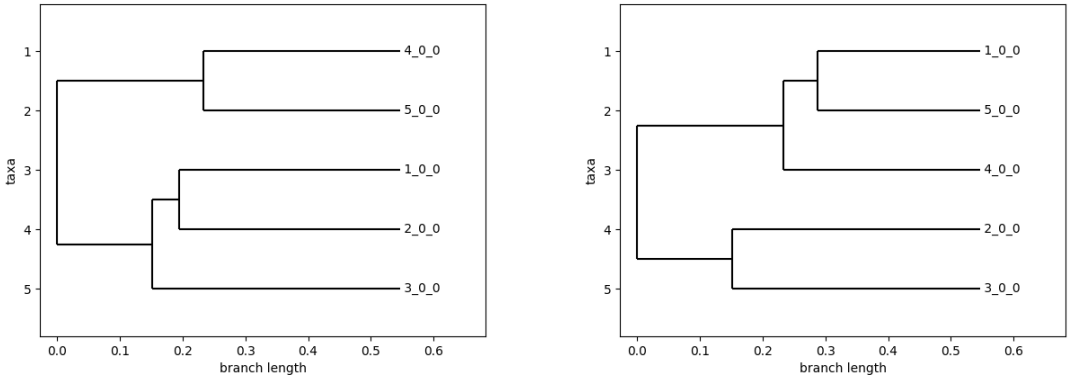
(b) gene tree

Figure 6.3: There are two values of 20 in the comparison plot 6.3a, from analysis of the method used in calculating model-based evolutionary distance this extreme value is caused by numerical problems. For example, when a logarithm of extremely small number is encountered the output is set manually by the distance calculation library we used. In this case it is set to 20. While setting a concrete number for the distance when the computation is undefined is valid in some contexts, four our purposes it was seriously biasing results, so we had to discard results such as these.

6.2 convolution window Distances and Linear Regression Plots

In this section we will present a series of convolution window distances calculated from one pair of gene trees together with the linear regression plots obtained from them. The two gene trees are shown in Figure 6.4. Note that in this sample the HGT event happens to node 1_0_0. For plots that do not contain 1_0_0 the non linearity between distance and window position is quite evident, please refer to Figures 6.9, 6.10, 6.11, 6.12, 6.13 and 6.14. However, for plots that do contain 1_0_0, the linearity is not as evident as we hoped. Figures 6.5 and 6.7 have small peaks, while Figure 6.8 has a deviation from the line in the beginning of the plot. Figure 6.6 follows the trend of the line as we expected.

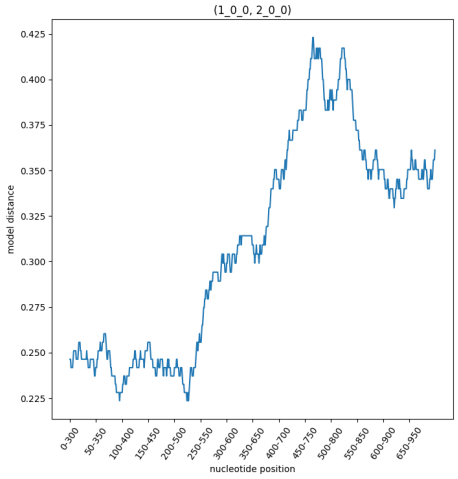
Figure 6.5 shows the distance between sequence 1_0_0 and 2_0_0. In the convolution window distance when can see a peak at position between 450 and 500. From our theory we are not expecting any peaks in the sliding window distances. However, the squared error (0.025) is relatively low and the distances show trend to increase with the increase of window position. From our theory a low squared error and a high slope value are characteristics of a HGT event. Figure 6.6shows the distance between sequence 1_0_0 and 3_0_0. This is a result that better fits our theory, we do not observe and strong variation. Figure 6.7 shows the distance between sequence 1_0_0 and 4_0_0. The plot contains 1_0_0 and therefore we expect some relationship. The plot has two small peaks but in general distances show trends of decreasing with increase of window position and has a relative low squared error of 0.023. Figure 6.8 shows the distance between sequence 1_0_0 and 5_0_0. We expect some sort of relationship in this plot. The plot shows a general trend of decreasing and it has a squared error of 0.018 which is small especially when there is a small deviation from the line in the beginning of the plot. Figure 6.9 shows the distance between sequence 2_0_0 and 3_0_0. Here we do not have any HGT event, we can see here a relatively high squared error of 0.046, which doubles the average of previous plots. Figure 6.10 shows the distance between sequence 2_0_0 and 4_0_0. Here we do not expect any linear relation. We can see a clear non linearity with a very low slope (0.0004) and a general disagreement between data and linear regression line. Figure 6.11 shows the distance between sequence 2_0_0 and 5_0_0. It shows also a non linearity with a high squared error of 0.06 and a low slope of 0.00001. Figure 6.12 shows the distance between sequence 3_0_0 and 4_0_0. It show a non linear relationship with a high squared error of 0.05. Figure 6.13 shows the distance between sequence 3_0_0 and 5_0_0. The plot has a high squared error of 0.042 and shows clear non linearity. Figure 6.14 shows the distance between sequence 1_0_0 and 2_0_0. The plot shows non linearity and a high squared error of 0.05.



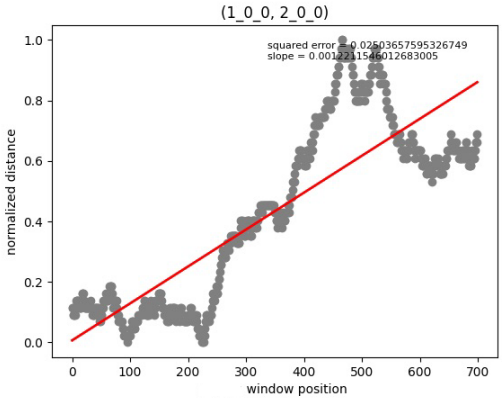
(a) gene tree 01

(b) gene tree 02

Figure 6.4: Two gene trees that are used to generate sequences used in the following convolution window calculation and respective linear regression. The HGT event happens to node 1_0_0.

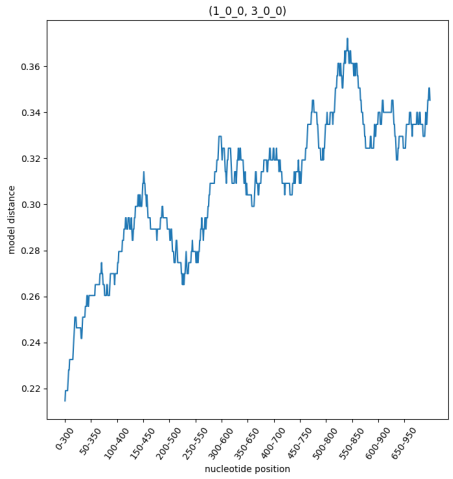


(a) convolution window distance between sequence 1_0_0 and 2_0_0 using JC69 model

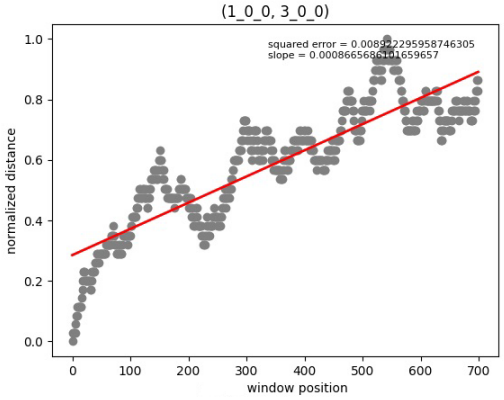


(b) linear regression of convolution window distance between sequence 1_0_0 and 2_0_0

Figure 6.5: Distance between sequence 1_0_0 and 2_0_0: From Figure 6.4 we can see that the node 1_0_0 is the object of a HGT event. In the convolution window distance we can see a peak at position between 450 and 500. However, the squared error is relatively low and the distances show trend to increase with the increase of convolution window start position.

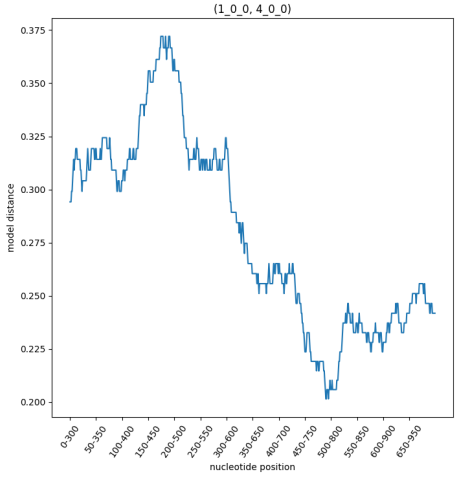


(a) convolution window distance between sequence 1_0_0 and 3_0_0 using JC69 model

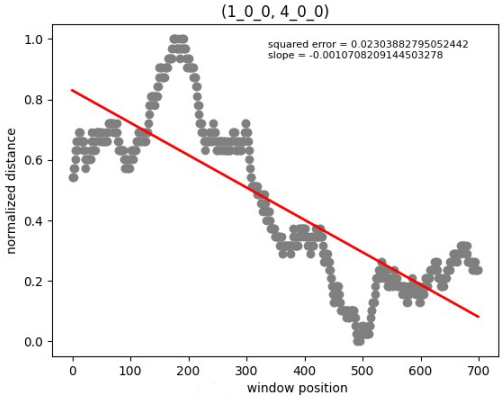


(b) linear regression of convolution window distance between sequence 1_0_0 and 3_0_0

Figure 6.6: Distance between sequence 1_0_0 and 3_0_0: From Figure 6.4 we can see that the node 1_0_0 is object of a HGT event. The distance between two sequences has the trend to increase with the movement of convolution window.

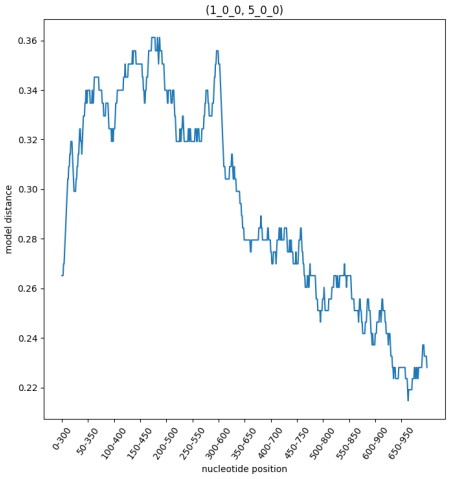


(a) convolution window distance between sequence 1_0_0 and 4_0_0 using JC69 model

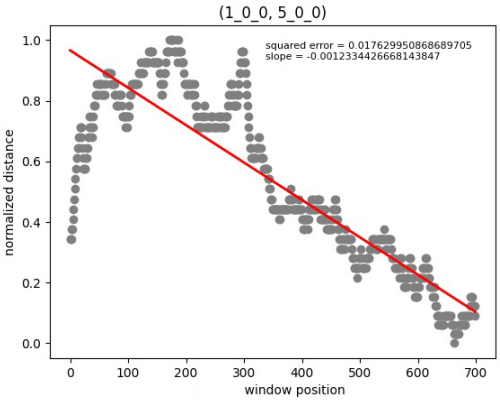


(b) linear regression of convolution window distance between sequence 1_0_0 and 4_0_0

Figure 6.7: Distance between sequence 1_0_0 and 4_0_0: From Figure 6.4 we can see that the node 1_0_0 is the object of a HGT event. The plot has two small peaks, but in general the distance shows the trend to decrease with the convolution window movement and the squared error value is relatively low.

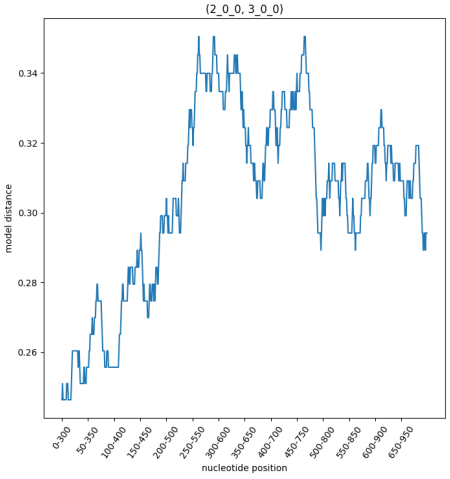


(a) convolution window distance between sequence 1_0_0 and 5_0_0 using JC69 model

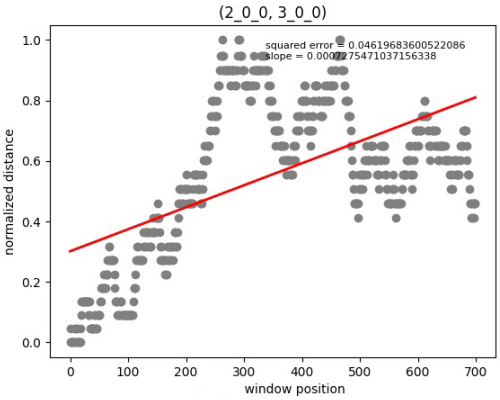


(b) linear regression of convolution window distance between sequence 1_0_0 and 5_0_0

Figure 6.8: Distance between sequence 1_0_0 and 5_0_0: From Figure 6.4 we can see that the node 1_0_0 is the object of a HGT event. The plot shows a general trend of decreasing, the squared error which indicates the deviation of the points is relatively low.

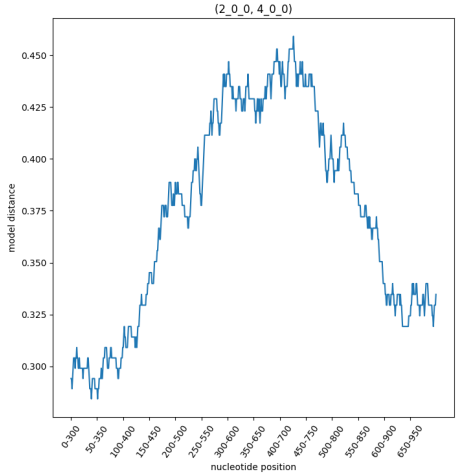


(a) convolution window distance between sequence 2_0_0 and 3_0_0 using JC69 model

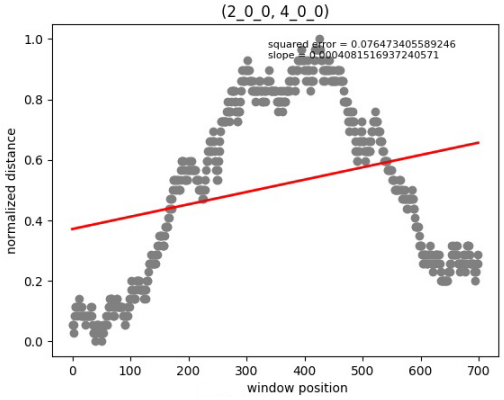


(b) linear regression of convolution window distance between sequence 2_0_0 and 3_0_0

Figure 6.9: Distance between sequence 2_0_0 and 3_0_0: From Figure 6.4 only 1_0_0 is object of HGT event. In this plot there is no HGT event happening. We can see a relative high squared error, before it was lower than 0.025, in this plot it is about 0.046.

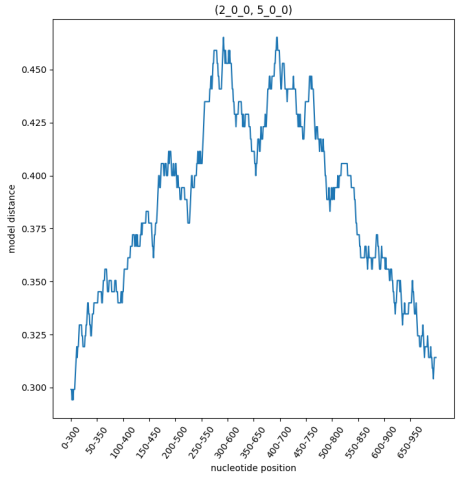


(a) convolution window distance between sequence 2_0_0 and 4_0_0 using JC69 model

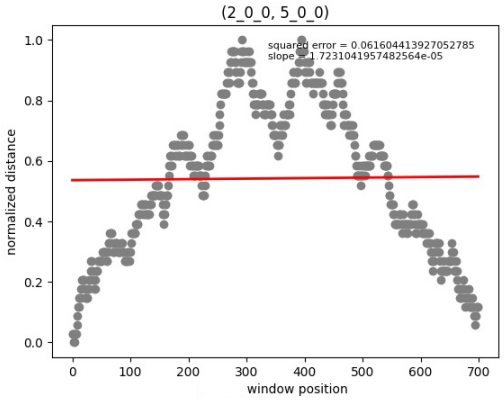


(b) linear regression of convolution window distance between sequence 2_0_0 and 4_0_0

Figure 6.10: Distance between sequence 2_0_0 and 4_0_0: From Figure 6.4 only 1_0_0 is object of HGT event. This plot does not have a clear trend of increasing or decreasing. Therefore the slope is relatively low with 0.0004 while other plots have a slope of 0.001. The squared error is also high due to the disagreement between data points and the linear regression line.

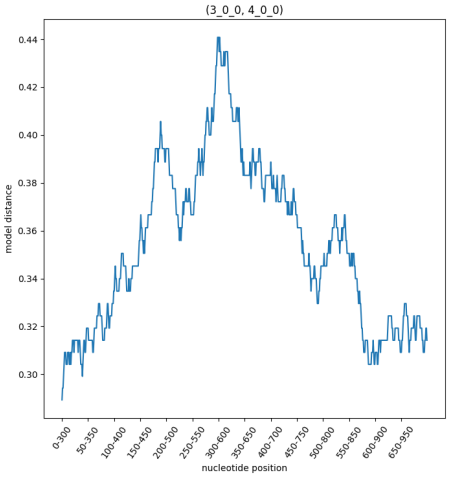


(a) convolution window distance between sequence 2_0_0 and 5_0_0 using JC69 model

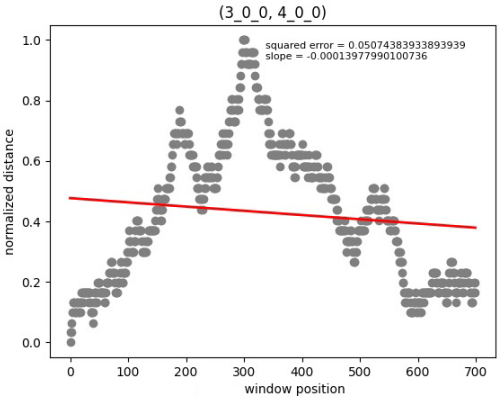


(b) linear regression of convolution window distance between sequence 2_0_0 and 5_0_0

Figure 6.11: Distance between sequence 2_0_0 and 5_0_0: From Figure 6.4 only 1_0_0 is the object of HGT event, and so we would expect to see no relationship. In Figure 6.10, this plot has a non linear relation between distance and convolution window position. It also has a high squared error (0.06) and a low slope value (0.00001).

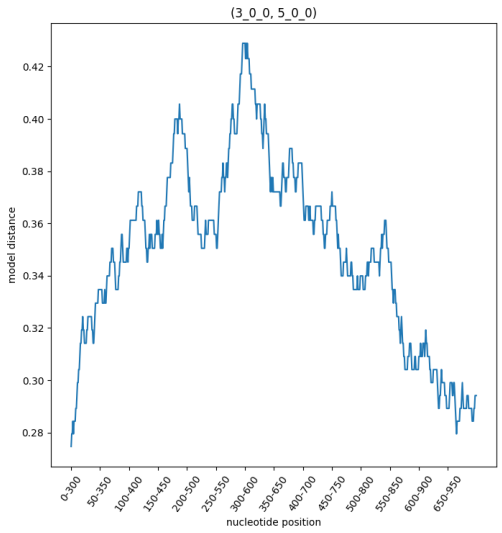


(a) convolution window distance between sequence 3_0_0 and 4_0_0 using JC69 model

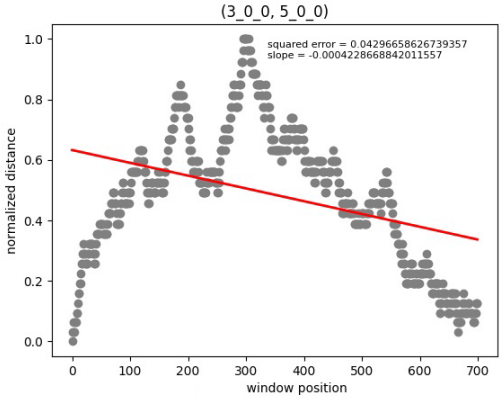


(b) linear regression of convolution window distance between sequence 3_0_0 and 4_0_0

Figure 6.12: Distance between sequence 3_0_0 and 4_0_0: From Figure 6.4 only 1_0_0 is the object of HGT event. Also this plot has a non linear relation between the distance and convolution window position. It also has a high squared error of 0.05.

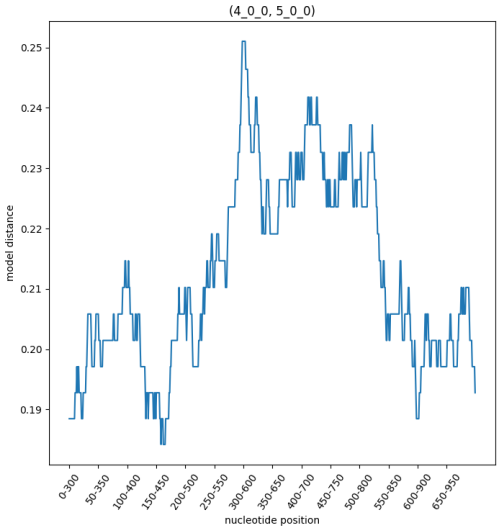


(a) convolution window distance between sequence 3_0_0 and 5_0_0 using JC69 model

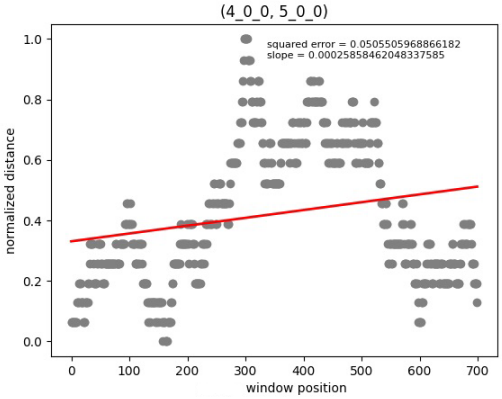


(b) linear regression of convolution window distance between sequence 3_0_0 and 5_0_0

Figure 6.13: Distance between sequence 3_0_0 and 5_0_0: From Figure 6.4 only 1_0_0 is the object of HGT event. Also this plot has a non linear relation between distance and convolution window position. It also has a high squared error of 0.042.



(a) convolution window distance between sequence 4_0_0 and 5_0_0 using JC69 model



(b) linear regression of convolution window distance between sequence 4_0_0 and 4_0_0

Figure 6.14: Distance between sequence 4_0_0 and 5_0_0: From Figure 6.4 only 1_0_0 is the object of HGT event. Also this plot has a non linear relation between distance and convolution window position. It also has a high squared error of 0.05.

6.3 Classification of convolution window Distances

In this last section we will describe various plots used to describe the results of machine learning classifiers, in particular ROC curves, confusion matrices and decision boundary plots. We have also a result distribution plot that shows the all the data points and their classed on Figure 6.15. The data are obtained from 70 pair of gene trees with 4 o 5 taxa. We can see that in the plot there are points that appear in the region of opposite class, we call this data overlapping. Causes of data overlapping may be errors during data computation or noises. In the following we will discuss the performance of these classifiers based on the three plots.

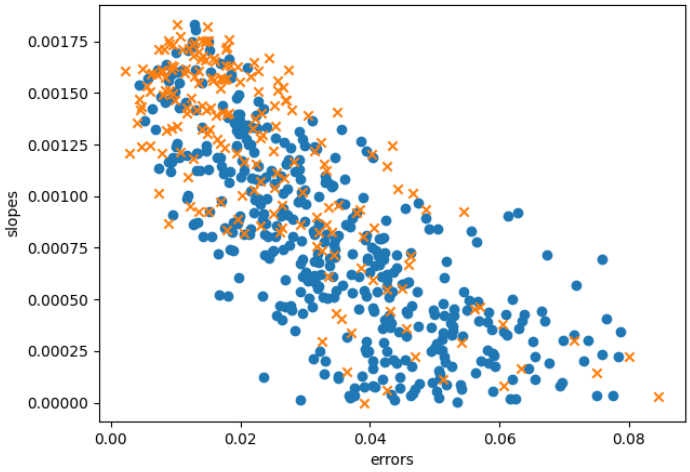
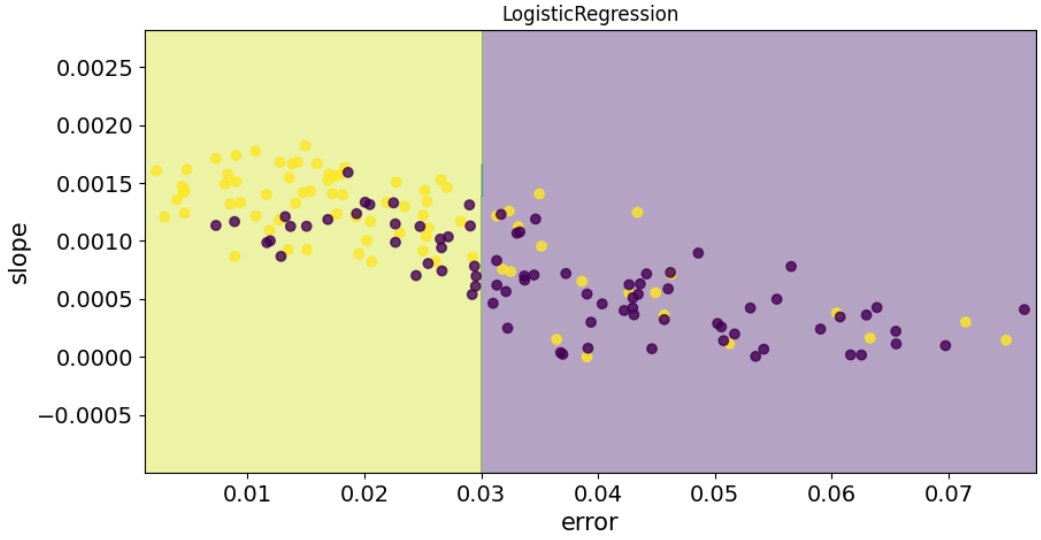


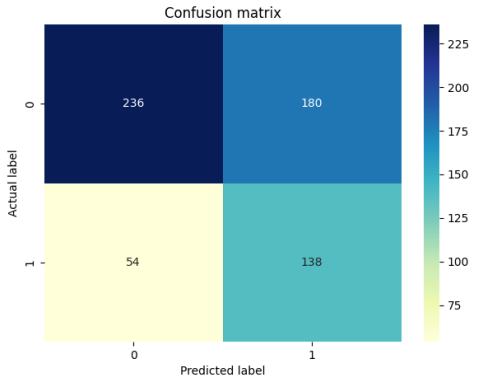
Figure 6.15: Result Distribution that contains all the data and their class

6.3.1 Logistic Regression

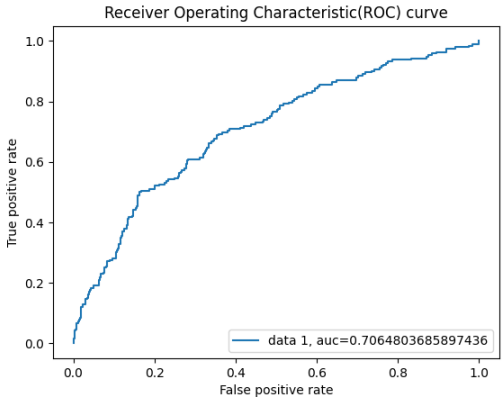
From Figure 6.16a we can see that Logistic regression has a linear decision boundary. A linear decision boundary does not perform well when there is data overlapping and because of data overlapping we have mis-classified data already during the training phase. The confusion matrix in Figure 6.16b shows a high counter for False Positive values, it is much high than the True Positive counter. The reason for this is also the overlapping of data that is very difficult to handle for logistic regression. We can see from the ROC curve in Figure 6.16b that the performance of linear regression on our dataset is not very high. It has an Area Under Curve (AUC) of only about 0.7.



(a) Decision boundary plot of Logistic Regression



(b) Confusion matrix of Logistic Regression

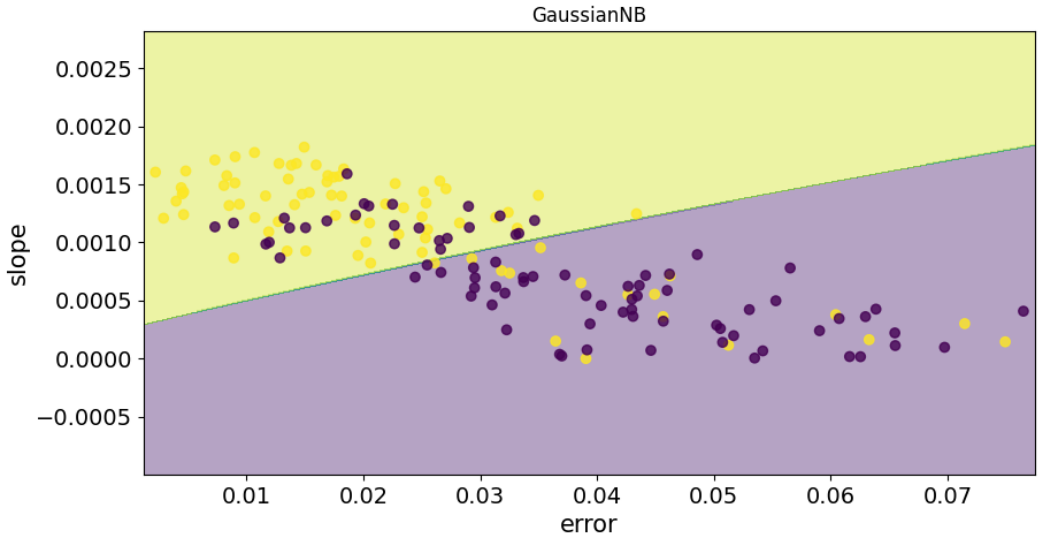


(c) ROC curve with AUC value of Logistic Regression

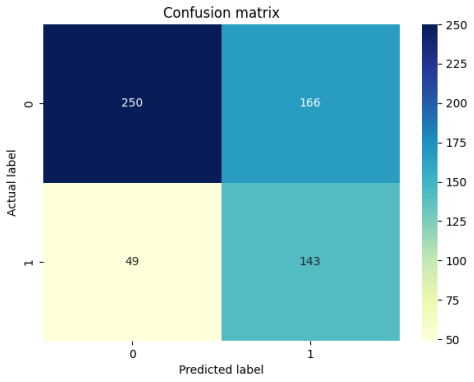
Figure 6.16: Logistic Regression Plots: The three plots show the result from the Logistic Regression.

6.3.2 Gaussian Naive Bayes

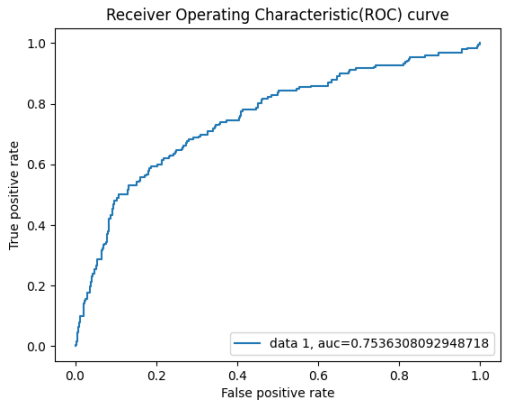
In the Figure 6.17a we can see the decision boundary of Gaussian Naive Bayes. Note that the decision boundary for Gaussian Naive Bayes in not a straight line but a parabola. However this type of decision boundary does not perform well with overlapped data. Also the confusion matrix for Gaussian Naive Bayes in Figure 6.17b shows a high counter for True Negative. The reason probably belongs to the same data overlapping problem. The ROC curve in Figure 6.17c is slightly better, it has an AUC of about 0.75 which indicates that it is an usable classifier for our data.



(a) Decision boundary plot of Gaussian Naive Bayes



(b) Confusion matrix of Gaussian Naive Bayes

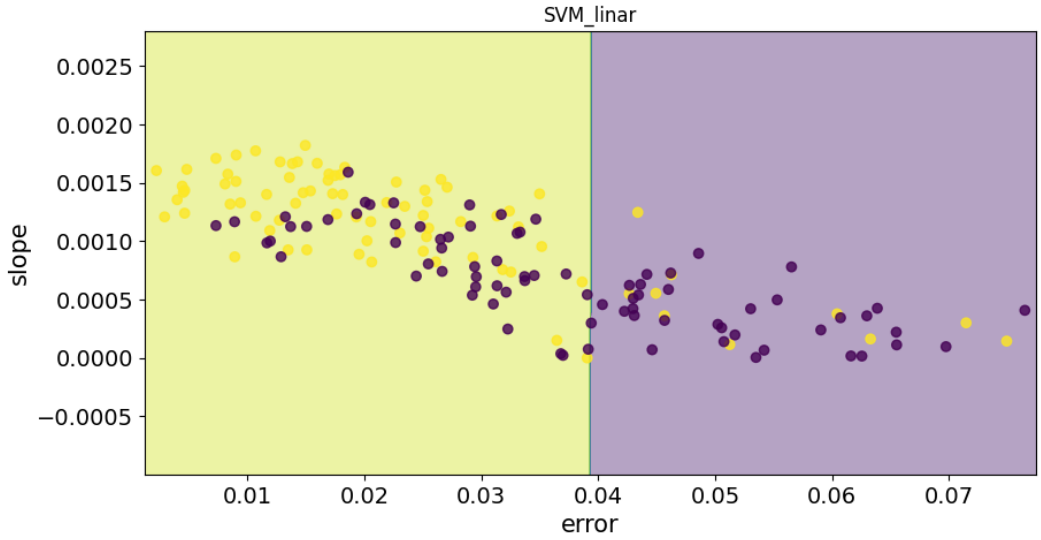


(c) ROC curve with AUC value of Gaussian Naive Bayes

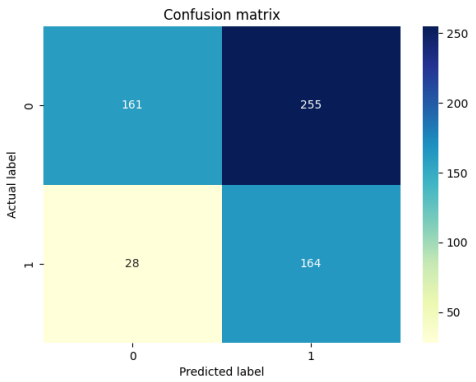
Figure 6.17: Gaussian Naive Bayes Plots: The three plots show results from the Gaussian Naive Bayes.

6.3.3 Support Vector Machine Classifier

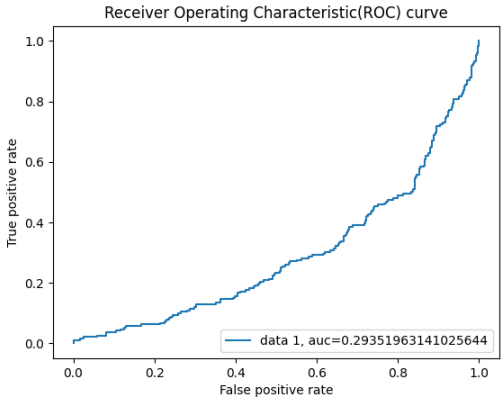
In Figure 6.18a we can see the decision boundary of SVM Classifier. Because it tries to maximize the margin between decision boundaries and data points, it may be strongly influenced by overlapping data. We think this is the reason for such poor performance and we can see that its ROC in Figure 6.18cis even worst than a random classifier. The True Negative counter in the confusion matrix (Figure 6.18b) is also very high.



(a) Decision boundary plot of Support Vector Machine Classifier



(b) Confusion matrix of Support Vector Machine Classifier

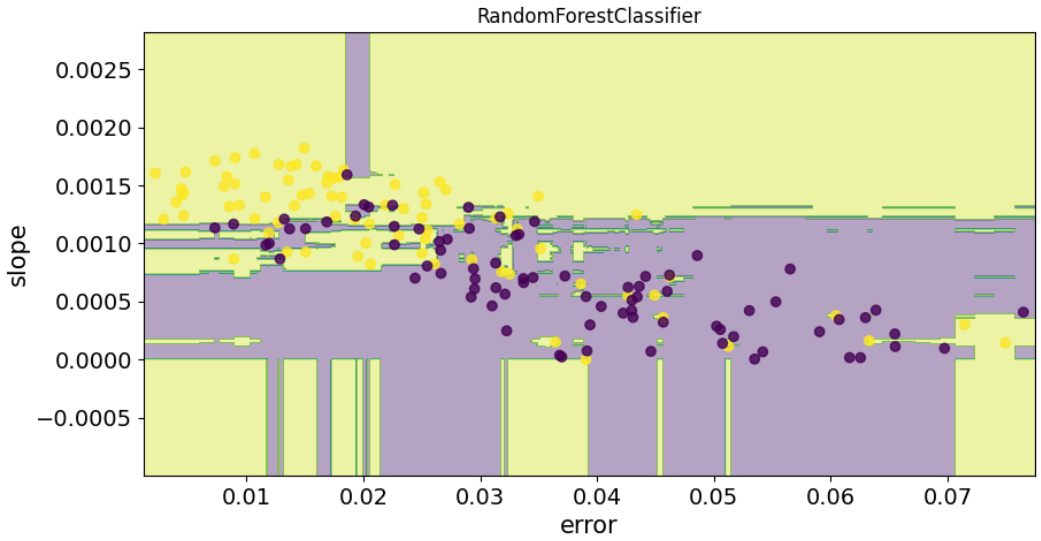


(c) ROC curve with AUC value of Support Vector Machine Classifier

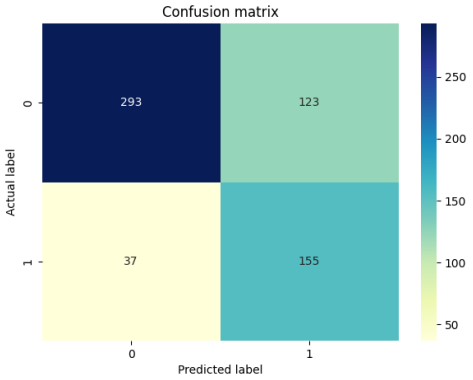
Figure 6.18: Support Vector Machine Classifier Plots: The three plots show results from the Support Vector Machine (SVM) Classifier.

6.3.4 Random Forest Classifier

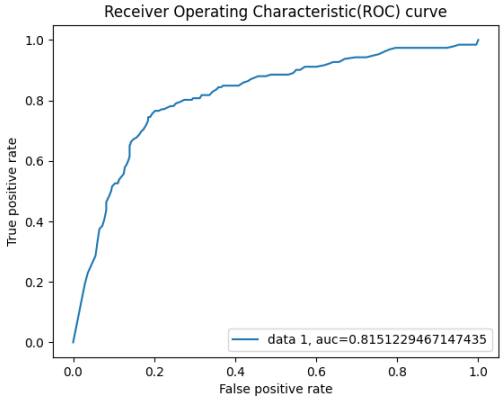
In Figure 6.19a we can see the decision boundary of the Random Forest Classifier and it is quite fragmented, which may be an indication of overfitting. However, because the form of its decision boundary, Random Forest is less subjective to sample overlapping. From the confusion matrix in Figure 6.19b we can also see a reduced False Positive counter. The shape of the ROC curve (Figure 6.19c) is quite good comparing to previous methods and it has an AUC of about 0.81.



(a) Decision boundary plot of Random Forest Classifier



(b) Confusion matrix of Random Forest Classifier

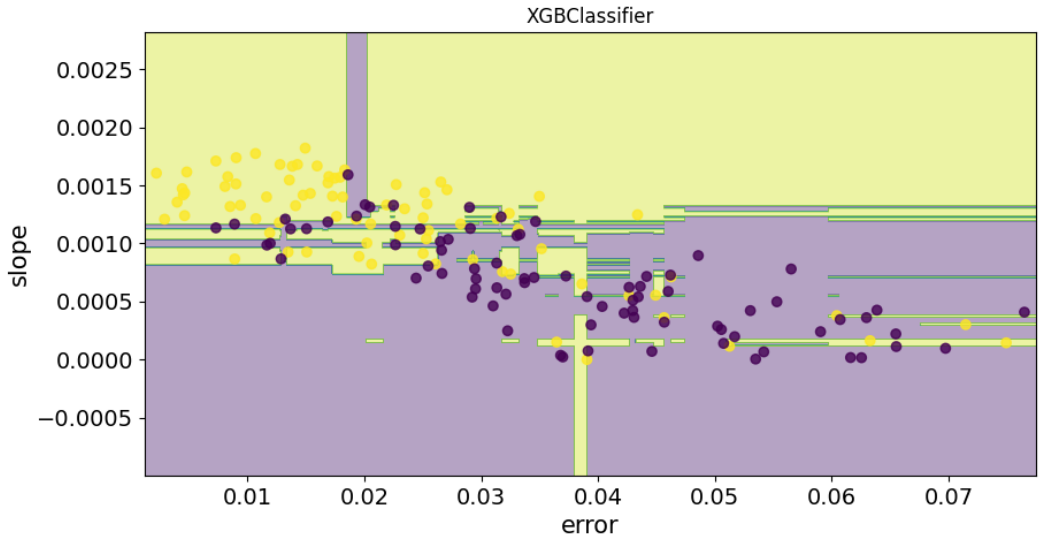


(c) ROC curve with AUC value of Random Forest Classifier

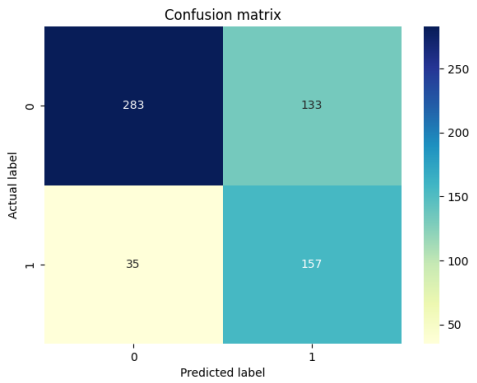
Figure 6.19: Random Forest Classifier Plots: The three plots show results from the Random Forest Classifier.

6.3.5 Extreme Gradient Boosting Classifier

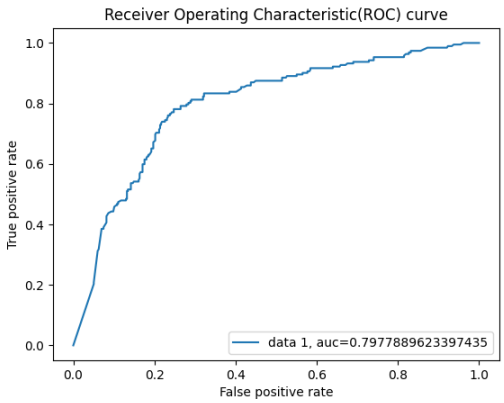
In figure 6.20a we can see the decision boundary of Extreme Gradient Boosting Classifier. Like the Random Forest Classifier, this decision boundary is quite fragmented. However, it is more reasonable than before when there is a large purple area on the bottom side of the plot. Contrary to our expectation, the confusion matrix (Figure 6.20b) and ROC curve (6.20c) are not better than those of Random Forest. It has a AUC of 0.80 which is lower than the one with Random Forest.



(a) Decision boundary plot of Extreme Gradient Boosting Classifier



(b) Confusion matrix of Extreme Gradient Boosting Classifier



(c) ROC curve with AUC value of Extreme Gradient Boosting Classifier

Figure 6.20: Extreme Gradient Boosting Classifier Plots: The three plots show results from the Extreme Gradient Boosting Classifier.

7 Conclusion and Future Work

At beginning of this thesis we have discussed about the possibility to detect HGT events given gene trees and respective sequences. We have reached to the conclusion that this is possible. We have developed HUGS to automatically distinguish which pair of distances may contain the trace of HGT. However, the problem is more complicated than what we expected and more work and studies must be done.

We have started with the comparison between patristic tree distances and model-based evolutionary distances. Contrary to our expectation, these first comparisons did not give valid results. We could not find any reliable connections between the variation of distances and HGT events. Then we decided to further analyze the model-based evolutionary distance with a convolution window approach. The results of the convolution window approaches show some possibility of detecting HGT events. Therefore we tried to classify the results of convolution window approach into two classes: one with HGT events and the other one without. Logistic Regression, Gaussian Naive Bayes and Support Vector Machine tend to behave poorly with our data because they tend to behave poorly in presence of data overlapping. However they are relatively faster, they do require less time during the training phase. In particular the performance of Gaussian Naive Bayes is better among the three method, it is fast and its accuracy of 75% is also acceptable. Random Forest Classifier has the best accuracy of 82%, however in order to reach this accuracy we have to give up computational efficiency. The training time required for Random Forest is the longest among the methods (15s). Extreme Gradient Boosting has also a high accuracy but the time required during training (3s) is lower than Random Forest Classifier. During the classification, further problems emerges and the data seems to contain noise and overlapping classes. The reason for overlapping and noise may lay in the convolution window distances. From analyzing the sliding window graphs we observed some similar pattern (also the slope and squared error are very similar) both in cases with HGT event and in cases without HGT. Thus further studies on the results of convolution window approach is needed.

During the creation of our dataset we have built tree by hand. However, building trees by hand substantially reduces the number of trees that can be generated, as it constitutes a time consuming process. This hence limits the size of our datasets. Another problem of our data generation process is that it the data are not generated from empirical data, that is, it may differ from the gene trees and sequences that are found in nature. Because of this, HUGS may not be able to correctly classify real gene trees. Hence, it may require further work and training such that it can be deployed on empirical data. Even when we could apply to the empirical data we do not know if they are truly correct, we do not have

a ground truth for them.

To improve HUGS, more studies on the convolution window applied to model-based evolutionary distances should be done. We need to know what apart from HGT events can influence the convolution window distances, in order to detect and eliminate sources of noise from our data. Currently we have extracted only two features from the convolution window, an increase of number of features will reduce the difficulty in classification problem. Furthermore, we may need other data than only the convolution window distance for the classification. In some of our dataset it is even difficult for human to distinguish if there is a HGT event in the plot. Further machine learning methods such as Support Vector Machines with more complicated kernels and K-Nearest Neighbors should be tried. Unfortunately, due to time constraints, we were unable to explore the results of more complicated machine learning approaches, such as neural net based methods. Neural Network often tries to find relationship between the features in a data set, in our case we have to further study our data and possible features that can be used. Hopefully Neural Network will help us to better understand our data.

Bibliography

[1] Paschalia Kapli, Ziheng Yang, and Maximilian J Telford. "Phylogenetic tree building in the genomic age". In: *Nature Reviews Genetics* 21.7 (2020), pp. 428–444. DOI: doi.org/10.1038/s41576-020-0233-0. URL: https://doi.org/10.1038/s41576-020-0233-0.

[2] Wayne P Maddison and L Lacey Knowles. "Inferring Phylogeny Despite Incomplete Lineage Sorting". In: *Systematic Biology* 55.1 (Feb. 2006), pp. 21–30. ISSN: 1063-5157. DOI: 10.1080/10635150500354928. eprint: https://academic.oup.com/sysbio/article-pdf/55/1/21/26554321/10635150500354928.pdf. URL: https://doi.org/10.1080/10635150500354928.

[3] Eliran Avni and Sagi Snir. "A new phylogenomic approach for quantifying horizontal gene transfer trends in prokaryotes". In: *Scientific reports* 10.1 (2020), pp. 1–14. DOI: 10.1038/s41598-020-62446-5. URL: https://doi.org/10.1038/s41598-020-62446-5.

[4] Gergely J. Szöllősi et al. "The Inference of Gene Trees with Species Trees". In: *Systematic Biology* 64.1 (July 2014), e42–e62. ISSN: 1063-5157. DOI: 10.1093/sysbio/syu048. eprint: https://academic.oup.com/sysbio/article-pdf/64/1/e42/24585192/syu048.pdf. URL: https://doi.org/10.1093/sysbio/syu048.

[5] Luay Nakhleh, Derek Ruths, and Hideki Innan. "Gene Trees, Species Trees, and Species Networks". In: *Meta-analysis and Combining Information in Genetics and Genomics* (July 2009). DOI: 10.1201/9781420010626.ch17.

[6] Luay Nakhleh. "Computational approaches to species phylogeny inference and gene tree reconciliation". In: *Trends in ecology & evolution* 28.12 (2013), pp. 719–728.

[7] Krister M Swenson and Nadia El-Mabrouk. "Gene trees and species trees: irreconcilable differences". In: *BMC bioinformatics*. Vol. 13. 19. BioMed Central. 2012, pp. 1–9. DOI: 10.1186/1471-2105-13-S19-S15. URL: https://doi.org/10.1186/1471-2105-13-S19-S15.

[8] T Ryan Gregory. "Understanding evolutionary trees". In: *Evolution: Education and Outreach* 1.2 (2008), pp. 121–137. DOI: 10.1007/s12052-008-0035-x. URL: https://doi.org/10.1007/s12052-008-0035-x.

[9] Melissa Emamalipour et al. "Horizontal gene transfer: from evolutionary flexibility to disease progression". In: *Frontiers in Cell and Developmental Biology* 8 (2020), p. 229. DOI: 10.3389/fcell.2020.00229. URL: https://doi.org/10.3389/fcell.2020.00229.

[10] Wayne P Maddison. "Gene trees in species trees". In: *Systematic biology* 46.3 (1997), pp. 523–536. DOI: 10.1006/mpev.1996.0390. URL: https://doi.org/10.1006/mpev.1996.0390.

[11] Ziheng Yang. *Molecular evolution: a statistical approach*. Oxford University Press, 2014. DOI: 10.1093/acprof:oso/9780199602605.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780199602605.001.0001.

[12] THOMAS H Jukes, Charles R Cantor, HN Munro, et al. "Mammalian protein metabolism". In: (1969).

[13] Simon Tavaré et al. "Some probabilistic and statistical problems in the analysis of DNA sequences". In: *Lectures on mathematics in the life sciences* 17.2 (1986), pp. 57–86.

[14] Matt Ravenhall et al. "Inferring horizontal gene transfer". In: *PLoS computational biology* 11.5 (2015), e1004095. DOI: 10.1371/journal.pcbi.1004095. URL: https://doi.org/10.1371/journal.pcbi.1004095.

[15] Vincent Daubin, Emmanuelle Lerat, and Guy Perrière. "The source of laterally transferred genes in bacterial genomes". In: *Genome biology* 4.9 (2003), pp. 1–12. DOI: 10.1186/gb-2003-4-9-r57. URL: https://doi.org/10.1186/gb-2003-4-9-r57.

[16] Gur Sevillya, Orit Adato, and Sagi Snir. "Detecting horizontal gene transfer: a probabilistic approach". In: *BMC genomics* 21.1 (2020), pp. 1–11. DOI: 10.1186/s12864-019-6395-5. URL: https://doi.org/10.1186/s12864-019-6395-5.

[17] Jotun Hein. "Reconstructing evolution of sequences subject to recombination using parsimony". In: *Mathematical Biosciences* 98.2 (1990), pp. 185–200. ISSN: 0025-5564. DOI: https://doi.org/10.1016/0025-5564(90)90123-G. URL: https://www.sciencedirect.com/science/article/pii/002555649090123G.

[18] GD Paul Clarke et al. "Inferring genome trees by using a filter to eliminate phylogenetically discordant sequences and a distance matrix based on mean normalized BLASTP scores". In: *Journal of bacteriology* 184.8 (2002), pp. 2072–2080. DOI: 10.1128/JB.184.8.2072-2080.2002. URL: https://doi.org/10.1128/JB.184.8.2072-2080.2002.

[19] Paul Bastide et al. "Phylogenetic Comparative Methods on Phylogenetic Networks with Reticulations". In: *Systematic Biology* 67.5 (June 2018), pp. 800–820. ISSN: 1063-5157. DOI: 10.1093/sysbio/syy033. eprint: https://academic.oup.com/sysbio/article-pdf/67/5/800/30001686/syy033.pdf. URL: https://doi.org/10.1093/sysbio/syy033.

[20] Dapeng Xiong et al. "Towards a better detection of horizontally transferred genes by combining unusual properties effectively". In: (2012). DOI: 10.1371/journal.pone.0043126. URL: https://doi.org/10.1371/journal.pone.0043126.

[21] Christophe Dessimoz, Daniel Margadant, and Gaston H. Gonnet. "DLIGHT – Lateral Gene Transfer Detection Using Pairwise Evolutionary Distances in a Statistical Framework". In: *Research in Computational Molecular Biology*. Ed. by Martin Vingron and Limsoon Wong. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 315–330. ISBN: 978-3-540-78839-3. DOI: 10.1007/978-3-540-78839-3_27. URL: https://doi.org/10.1007/978-3-540-78839-3_27.

[22] Chen Li, Jiaxing Chen, and Shuai Cheng Li. "Deep learning for HGT insertion sites recognition". In: *BMC genomics* 21.11 (2020), pp. 1–18. DOI: 10.1186/s12864-020-07296-1. URL: https://doi.org/10.1186/s12864-020-07296-1.

[23] Diego Mallo, Leonardo de Oliveira Martins, and David Posada. "SimPhy: phylogenomic simulation of gene, locus, and species trees". In: *Systematic biology* 65.2 (2016), pp. 334–344. DOI: doi.org/10.1093/sysbio/syv082. URL: https://doi.org/10.1093/sysbio/syv082.

[24] William Fletcher and Ziheng Yang. "INDELible: A Flexible Simulator of Biological Sequence Evolution". In: *Molecular Biology and Evolution* 26.8 (May 2009), pp. 1879–1888. ISSN: 0737-4038. DOI: 10.1093/molbev/msp098. eprint: https://academic.oup.com/mbe/article-pdf/26/8/1879/3033727/msp098.pdf. URL: https://doi.org/10.1093/molbev/msp098.

[25] Andrew Rambaut and Nicholas C Grass. "Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees". In: *Bioinformatics* 13.3 (1997), pp. 235–238. DOI: 10.1093/bioinformatics/13.3.235. URL: https://doi.org/10.1093/bioinformatics/13.3.235.

[26] Reed A Cartwright. "DNA assembly with gaps (Dawg): simulating sequence evolution". In: *Bioinformatics* 21.Suppl_3 (2005), pp. iii31–iii38. DOI: 10.1093/bioinformatics/bti1200. URL: https://doi.org/10.1093/bioinformatics/bti1200.

[27] Peter JA Cock et al. "Biopython: freely available Python tools for computational molecular biology and bioinformatics". In: *Bioinformatics* 25.11 (2009), pp. 1422–1423. DOI: 10.1093/bioinformatics/btp163. URL: https://doi.org/10.1093/bioinformatics/btp163.

[28] Rob Knight et al. "PyCogent: a toolkit for making sense from sequence". In: *Genome biology* 8.8 (2007), pp. 1–16. DOI: 10.1186/gb-2007-8-8-r171. URL: https://doi.org/10.1186/gb-2007-8-8-r171.

[29] *Python-Tabulate Project Description Page*. URL: https://pypi.org/project/tabulate/.

[30] Fabian Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12.85 (2011), pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

[31] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[32] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: ACM, 2016, pp. 785–794. ISBN: 978-1-4503-4232-2. DOI: 10.1145/2939672.2939785. URL: http://doi.acm.org/10.1145/2939672.2939785.

[33] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[34] Jayawant N. Mandrekar. "Receiver Operating Characteristic Curve in Diagnostic Test Assessment". In: *Journal of Thoracic Oncology* 5.9 (2010), pp. 1315–1316. ISSN: 1556-0864. DOI: https://doi.org/10.1097/JTO.0b013e3181ec173d. URL: https://www.sciencedirect.com/science/article/pii/S1556086415306043.