Karlsruhe Institute of Technology

Use Cases of Predictive Modeling for Phylogenetic Inference and Placements

Master's Thesis of

Julius Wiegert

at the Department of Informatics
Institute of Theoretical Informatics (ITI)

Reviewer:	Prof. Dr. Alexandros Stamatakis
Second reviewer:	Jun.-Prof. Dr. Christian Wressnegger
Advisor:	M.Sc. Julia Haag
Second advisor:	M.Sc. Dimitri Höhler

09. October 2023 – 07. March 2024

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.

Freudenstadt, 07.03.2024

[illegible]
. . . .
(Julius Wiegert)

Abstract

In this work, we present two distinct applications of predictive modeling within the domain of phylogenetic inference and placement.

Phylogenetic placements aim to place new entities into a given phylogenetic tree. While there exist efficient implementations for producing phylogenetic placements, the underlying reasons why particular placements are more difficult to perform than others are unknown.

In the first use case, we focus on the prediction of the difficulty of those phylogenetic placements. We developed *Bold Assertor of Difficulty (BAD)*. BAD can predict the placement difficulty between 0 (easy) and 1 (hard) with high accuracy. On a set of 3000 metagenomic placements, we obtain a mean absolute error of 0.13. BAD can help biologists understand the challenges associated with placing specific sequences into a reference phylogeny during metagenomic studies based on SHapley Additive exPlanations (SHAP) explanations.

Estimating the statistical robustness of the inferred phylogenetic tree constitutes an integral part of most phylogenetic analyses. Commonly, one computes and assigns a branch support value to each inner branch of the inferred phylogeny. The most widely used method for calculating branch support on trees inferred under maximum likelihood is the Standard, non-parametric Felsenstein Bootstrap Support (SBS). The SBS method is computationally costly, leading to the development of alternative approaches such as *Rapid Bootstrap* and *UltraFast Bootstrap 2* (UFBoot2).

The second use case of this work is concerned with the fast machine learning-based approximation of those SBS values. Our SBS predictor, *Educated Bootstrap Guesser (EBG)*, is on average 9.4 ($\sigma = 5.5$) times faster than the major competitor UFBoot2 and provides an SBS estimate with a median absolute error of 5 when predicting SBS values between 0 and 100.

Zusammenfassung

Diese Arbeit beschreibt zwei Anwendungen von prädiktiver Modellierung im Bereich der phylogenetischen Inferenz und Platzierung.

Phylogenetische Platzierung beschreibt den Prozess neue Entitäten in einen gegebenen phylogenetischen Baum zu platzieren. Während es effiziente Implementierungen für die Erzeugung phylogenetischer Platzierungen gibt, bleiben die zugrunde liegenden Gründe, warum bestimmte Platzierungen schwieriger auszuführen sind als andere, unbekannt.

In unserem ersten Anwendungsfall von prädikativer Modellierung fokusieren wir uns auf die Vorhersage der Schwierigkeit von phylogenetischen Platzierungen und stellen den *Bold Assertor of Difficulty (BAD)* vor. BAD kann die Schwierigkeit phylogenetischer Platzierungen auf einer Skala von 0 (einfach) bis 1 (schwierig) mit hoher Genauigkeit vorhersagen. Auf einem Datensatz von 3000 metagenomischen Platzierungen ist der mittlere absolute Fehler der Vorhersage von BAD nur 0.13. BAD kann Biologen bei metagenomischer Studien unterstützen, indem er Erklärungen für Vorhersagen mittels SHapley Additive exPlanations (SHAP)-Erklärungen bietet.

Die Abschätzung der statistischen Robustheit eines berechneten phylogenetischen Baumes ist ein wichtiger Bestandteil der meisten phylogenetischen Analysen. Als Robustheitsmaß werden üblicherweise sogenannte Support-Werte für alle inneren Zweige des phylogenetischen Baums berechnet. Eine weit verbreitete Standardmethode für die Berechnung von Support-Werten ist der nicht-parametrische Felsenstein Bootstrap Support (SBS). Aufgrund der Rechenintensität der SBS-Prozedur wurden alternative Ansätze wie *Rapid Bootstrap* und *UltraFast Bootstrap 2* (UFBoot2) entwickelt.

Der zweite Anwendungsfall in dieser Arbeit beschreibt die Vorhersage von SBS-Werten mittels maschinellem Lernen. Unser SBS-Prädiktor *Educated Bootstrap Guesser (EBG)* ist im Schnitt 9.4-mal ($\sigma = 5.5$) schneller als der Hauptkonkurrent UFBoot2 und liefert eine SBS-Schätzung mit einem mittleren absoluten Fehler von 5 bei der Vorhersage von SBS-Werten zwischen 0 und 100.

Contents

List of Figures

List of Tables

1. Introduction

Geisser [37] defines predictive modeling as the development of a mathematical tool or model that generates an accurate prediction. As a powerful facet of data science and machine learning, predictive modeling demonstrates success in diverse domains, such as electricity consumption prediction [2] and weather forecasting [16]. In biology, its usage ranges from tasks like the prediction of the three-dimensional structure of proteins [49] to the machine learning-based discovery of mutations in human cells [105]. This work focuses on the specific biological subarea of phylogenetics.

Phylogenetic methods establish evolutionary relationships among individuals, groups, species, or populations [15]. We typically depict those relationships in a tree structure that we infer using methods of computational phylogenetics. One important technique for phylogenetic tree inference is the optimization of the Maximum Likelihood (ML) criterion. This method strives to identify the most plausible phylogenetic tree that elucidates the biological data we observe from biological entities, e.g. in the form of gene sequences. One successful application of predictive modeling in phylogenetics is the prediction of the difficulty associated with the inference process of a phylogenetic tree for a given set of sequences [42]. Building on this, Togkousidis et al. [102] use this difficulty to obtain a speedup in the process of the phylogenetic tree search.

Our work aims to explore novel applications of predictive modeling in the realm of phylogenetic tree inference. We consider the area of placing new data into existing phylogenetic trees as a potential use case for predictive modeling. We aim to introduce a quantifiable notion of difficulty for these phylogenetic placement processes. To this end, we identify a set of features for the prediction of placement difficulty. Additionally, we seek to point out the capabilities of explainable AI techniques to gain insights into the phylogenetic placement process.

Moreover, we apply predictive models to enhance the efficiency of phylogenetic analyses. More specifically, we aim to predict the statistical robustness of branches in phylogenetic trees. The quantification of uncertainty of branches in phylogenetic inference is an important part of modern phylogenetic pipelines [50]. Therefore, a fast machine learning-based statistical robustness test could provide an efficiency gain for phylogenetic analysis.

Through these two use cases, our research aims to contribute to applying predictive modeling techniques within the domain of phylogenetics.

We start in Chapter 2 with an introduction to phylogenetic inference and placements under the ML criterion and selected properties of phylogenetic trees. Chapter 3 describes the machine learning techniques we employ in this work. Chapter 4 deals with the development of our first use case, the phylogenetic placement difficulty predictor *Bold Assertor of Difficulty* (BAD), whereas Chapter 5 describes *Educated Bootstrap Guesser* (EBG), a tool for bootstrap support prediction and our second use case. Finally, Chapter 6 summarizes this work and lays out future work.

2. Phylogenetics

Phylogenetic trees have applications in assessing the process of speciation [64], epidemiology [104], classification of proteins and genes [18], and more [109, 68, 85]. We estimate phylogenetic trees based on evolutionary signals present in genetic material such as the DeoxyriboNucleic Acid (DNA) or in proteins, encoded as Amino Acid (AA) sequences [15].

Phylogenetic trees depict these evolutionary relationships [71]. Figure 2.1 shows an example of a phylogenetic tree of six bacterial species. A tree is an acyclic, connected graph (V, E) where V is the set of vertices (nodes) and E is the set of edges, called branches in phylogenetics [59]. We call nodes connected through only one single branch *leaves* or *taxa* (singular: *taxon*) [59]. Each branch defines a *bipartition* of the taxa in the tree. A bipartition is a split of the taxa into two mutually exclusive sets [108]. The branch length between two nodes in the tree represents the evolutionary distance between them [108]. Phylogenetic trees typically are bifurcating, meaning that each internal node has exactly two children [71]. Internal nodes represent hypothetical common ancestors of the taxa [71].

Branching in an evolutionary tree is also called *lineage splitting* [86]. After this splitting, the evolution of the two created child lineages occurs independently. The sequences of lineage splits comprise the topology of a phylogenetic tree [86]. Phylogenetic trees can be rooted or unrooted.

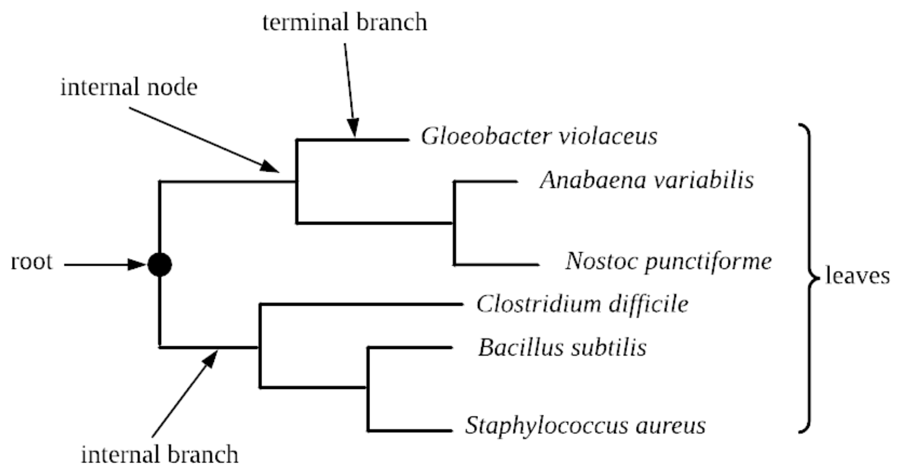


Figure 2.1.: Phylogenetic tree of bacterial species. Based on Fig. 1 of Moreira [71].

The root corresponds to the hypothetical most recent common ancestor [86] of all taxa. Rooting is important for the directionality of the evolutionary process and the interpretability of the tree [76]. We can root an unrooted phylogenetic tree by placing a distantly related species, called *outgroup*, into the tree. We place the root in the branch to the outgroup resulting in a rooted tree for the original taxa, called *ingroup* [108]. In the following, we explain how we obtain phylogenetic trees.

2.1. Phylogenetic Inference

Phylogenetic inference is the task of inferring the evolutionary history for a given set of taxa [43]. For this inference process, we use morphological (anatomical traits) or molecular data such as DNA or AA sequences. Molecular data provides an independent source of evidence from morphological studies and can reveal patterns of evolution obscured at the morphological level [43]. Furthermore, the ease of availability and representation of molecular data allows for using efficient computational methods for inferring phylogenetic trees [43]. We can divide phylogenetic tree inference methods into distance-based and character-based methods [59].

The distance-based methods compute a pairwise distance matrix between all taxa [59] and subsequently use this matrix to create the phylogenetic tree. Examples of this approach are the Unweighted Pair Group Method with Arithmetic mean (UPGMA) [91] or Neighbor Joining (NJ) [82]. Distance-based methods tend to perform poorly when estimating large evolutionary distances [108].

Character-based methods split the phylogenetic inference problem into two subproblems: evaluating an optimality criterion for a given phylogenetic tree topology and optimizing the tree topology itself [59]. Two approaches for this class of phylogenetic inference are Maximum Parsimony (MP) and ML.

MP follows the principle of preferring simple hypotheses for evolution in the taxa over complex ones. For phylogenetic trees, this results in minimizing the total amount of evolutionary change in the data [26]. The tree with the least character changes is the MP tree. There exist efficient dynamic programming algorithms for the evaluation of the parsimony criterion, like Sankhoff's [83] or Fitch's [34] algorithm. MP suffers from problems such as the clustering of long branches (called *long branch attraction*) and not capturing different substitution rates between the nucleotides [108]. Furthermore, MP does not include branch length information. Branch lengths represent the extent of accumulated evolutionary change and are thus a valuable source of information [108].

This work focuses on phylogenetic inference under the ML criterion. The ML criterion evolves from the MP criterion by the incorporation of differences in branch lengths and substitution rates between nucleotides [108]. Figure 2.2 summarizes the ML tree inference procedure.

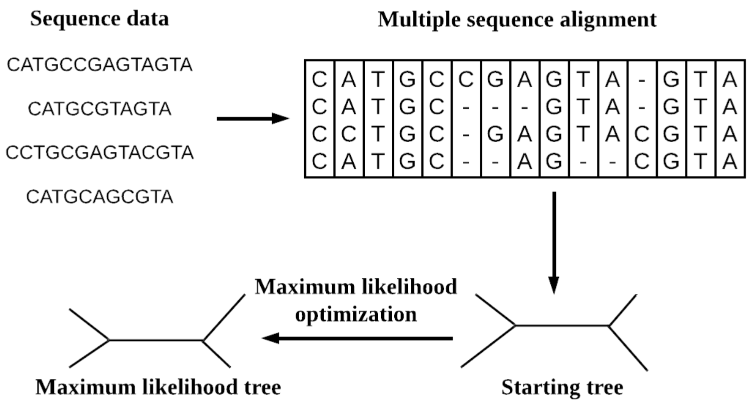

Sequence data
CATGCCGAGTAGTA
CATGCGTAGTA
CCTGCGAGTACGTA
CATGCAGCGTA
Multiple sequence alignment
C A T G C C G A G T A - G T A
C A T G C - - - G T A - G T A
C C T G C - G A G T A C G T A
C A T G C - - A G - - C G T A
Maximum likelihood optimization
Maximum likelihood tree
Starting tree

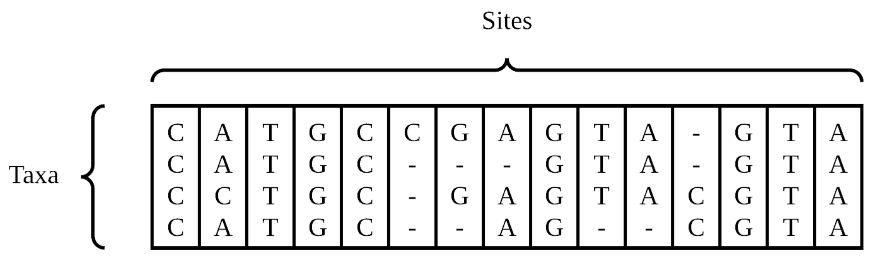
Figure 2.2.: Overview of ML tree inference procedure

In Section 2.1.1 we describe the foundations of Multiple Sequence Alignments (MSAs). Following this, in Section 2.1.2 to Section 2.1.5 we explain the calculation of the likelihood before delving into the fundamentals of likelihood maximization and optimization of tree topologies. Finally, in Section 2.1.6 we briefly describe RAxML-NG [58] as an example of a ML-based phylogenetic inference tool.

2.1.1. Multiple Sequence Alignment

The input for phylogenetic inference under the ML criterion are MSAs [59]. We represent each taxon in the phylogenetic inference process by a sequence of discrete characters (e.g., DNA or AA). We assume those sequences to be *homologous*, meaning that they are similar due to descent with modification from a common ancestor [28]. We are not able to prove homology as this requires proving common ancestry [28]. *Convergence* describes the phenomenon of sequence similarity without common ancestry. If the sequences of two taxa are similar enough, we assume homology instead of convergence, however, there is no consensus on the threshold of sequence similarity for this homology assumption [28].

An MSA algorithm aligns a set of taxa in a data matrix. Each row of the matrix corresponds to a taxon. The columns are the homologous sites of the set of taxa. The algorithm assigns positional homology to each site s of the data matrix by inserting gaps into the sequences [28]. We call the resulting aligned matrix, as well as the procedure, an MSA [108]. Figure 2.3 shows an example MSA with four taxa and 17 sites. It makes the sequences comparable for further



C	A	T	G	C	C	G	A	G	T	A	-	G	T	A
C	A	T	G	C	-	-	-	G	T	A	-	G	T	A
C	C	T	G	C	-	G	A	G	T	A	C	G	T	A
C	A	T	G	C	-	-	A	G	-	-	C	G	T	A

Figure 2.3.: Example for a DNA MSA with four taxa.

evolutionary analysis [94]. We denote the k-th site of the i-th sequence (taxon) as $s_{i,k}$.

It is possible to use classic pairwise sequence alignment techniques like the Needleman-Wunsch Algorithm [87] for creating MSAs, but it is computationally expensive due to the exponentially growing number of sequence comparisons $N = (S - 1)!$ with S being the number of sequences [94].

More efficient *progressive methods* use lightweight sequence comparisons such as k-mer similarities to construct a guide tree. The guide tree construction uses a phylogenetic inference method that does not need an MSA as input, e.g. UPGMA or NJ. This guide tree determines the sequence alignment order, progressively adding more sequences to the resultant alignment. To manage complexity, progressive methods compute a consensus sequence of the current alignment after each alignment step, representing it in subsequent steps. We define the consensus sequence as the sequence comprising the most frequently occurring characters (nucleotides or AAs), at each position within an alignment [3].

A drawback of the progressive approach lies in the potential propagation of alignment errors from initial alignment steps throughout the entire procedure [94]. Iterative progressive methods address this issue by iteratively disaligning previously aligned groups and realigning them, thereby mitigating potential errors. Two examples of iterative progressive alignment tools are MUSCLE [25] and MAFFT [52]. MAFFT calculates the Fast Fourier Transform (FFT) of numerical representations of the sequences to compare them in the spectral domain using the similarities obtained for the guide tree creation [52].

2.1.2. Substitution Models

The evaluation of the likelihood of a given tree topology requires a substitution model that includes the probabilities for evolutionary changes in DNA or AA data. From a formal point of view, these models are continuous-time Markov chains. Under the assumption that sites evolve independently, we characterize each site by a Markov chain with, e.g., four states (A, C, T, and G) for DNA data [108].

To define a Markov chain, we require a *transition-probability matrix*. To calculate this matrix, we define a substitution-rate matrix Q, which represents the instantaneous rate of change between the nucleotides [108]. The most general substitution-rate matrix is the General Time Reversible (GTR) model

$$Q = \begin{pmatrix} - & a\pi_c & b\pi_g & c\pi_t \\ a\pi_a & - & d\pi_g & e\pi_t \\ b\pi_a & d\pi_c & - & f\pi_t \\ c\pi_a & e\pi_c & f\pi_g & - \end{pmatrix}, \tag{2.1}$$

where a to f are the rates of the corresponding type of substitution and π_x is the equilibrium frequency of $x \in \{A, C, T, G\}$. The diagonal elements of Q are the negative sum of the off-diagonal elements in the row. Furthermore, a relative definition of the equilibrium frequencies π_x and the rates reduces the parameters of the GTR model to eight [108].

Since the GTR model is the most general substitution model, all other substitution models are submodels of it [79]. Other models assume fewer free parameters compared to the GTR model. For example, the Jukes-Cantor [48] model assumes that all nucleotides are equally frequent at equilibrium and the rate of one nucleotide changing to another is equal among all nucleotides. It can be obtained by setting $\pi_a = \pi_c = \pi_g = \pi_t$ and $a = b = c = d = e = f$ in the Q matrix of Equation (2.1). The Kimura [55] model only distinguishes between changes from pyrimidine (C, T) to purine (A, G) nucleotides. We call changes from pyrimidine to purine (and vice versa) transversions, and changes that stay within the nucleotide group transitions. This results in a different rate of change for both cases [108].

Using Q we define the transition-probability matrix as

$$P(t) = e^{Qt}, \tag{2.2}$$

which is a matrix with entries $P_{xy}(t)$. Those entries yield the change probability from x to y over a branch of length t where, for DNA data, $x, y \in \{A, C, T, G\}$ [59]. We can solve Equation (2.2) using eigendecomposition or diagonalization of Q [108]. $P_{xy}(t)$ describes the mutation of one nucleotide into another as a function of time [59]. The time-reversibility of the substitution model captures the property that $\pi_x P_{xy}(t) = \pi_y P_{yx}(t)$ where P_{xy} is the probability of nucleotide x mutating to nucleotide y in time t and π_x is the frequency of nucleotide x at the equilibrium [59].

For AA sequences, the Q matrix of the GTR model (Equation (2.1)) comprises 20×20 entries, a consequence of the expanded parameter space resulting from the AA residue code. The GTR model for AAs has $(19 \times 20)/2 - 1 = 189$ free parameters for the matrix Q as well as 19 free parameters for the equilibrium frequencies of the AAs [108]. Simpler AA substitution models like the Poisson [73] model are analogous to the Jukes-Cantor model and specify equal frequencies and substitution probabilities for all AAs.

Another facet of substitution models is the necessity for selective pressure to vary across different sites or regions of genes or proteins. These pressure differences are due to the diverse roles of sites in the structure and function of the encoded protein [108]. A prevalent approach for modeling this variation is to employ a Γ distribution to model the distribution of mutation rates r and sample from this distribution [108]. The α parameter determines the shape of the Γ distribution [108].

Given the substitution model, a tree topology, and an MSA, we can compute the likelihood of the tree.

2.1.3. Likelihood Evaluation

We define the likelihood of a tree

$$L(Tree, \theta | Data) = P(Data | Tree, \theta) \tag{2.3}$$

as the probability of observing the data (the MSA) given a tree topology and θ as the combination of the substitution model parameters and the branch lengths [43]. To assess the likelihood of a tree, it is necessary to calculate the probabilities for each nucleotide (or AA) at the internal nodes. Under the assumption of independence, we compute the likelihood independently for all m sites as

$$\prod_{i=1}^{m} P(s_i | Tree, \theta). \tag{2.4}$$

To avoid numerical challenges during the computation, we employ the log-likelihood

$$\sum_{i=1}^{m} log(P(s_i | Tree, \theta)). \tag{2.5}$$

A computationally efficient method for computing the log-likelihood for a specified phylogenetic tree topology is the Felsenstein pruning algorithm [30]. This algorithm works by traversing the tree in a post-order manner, extending up to the root of the phylogenetic tree. At the leaves, the algorithm establishes the likelihood vectors based on the observed data. For example, if the site s contains an A at a specific leaf, the corresponding likelihood vector would be $L(s) = [1, 0, 0, 0]$. We can handle gaps in the sequences by treating them as an ambiguous nucleotide, i.e. $L(s) = [1, 1, 1, 1]$ [108].

The algorithm computes the log-likelihood of an internal node k at site s for nucleotide x recursively based on the likelihoods of the left (l) and right (r) child node as

$$L_x^k(s) = \left(\sum_{n \in T} P_{xn}(b_l) L_n^{(l)}(s) \right) \left(\sum_{n \in T} P_{xn}(b_r) L_n^{(r)}(s) \right). \tag{2.6}$$

T is the set of nucleotides and b_l and b_r are the branch lengths connecting the left and right child nodes.

$$L(s) = \sum_{n \in T} \left(\pi_n L_n^{(l)}(c) \sum_{m \in T} P_{nm}(b_{vr}) L_m^{(r)}(c) \right). \tag{2.7}$$

is the log-likelihood of a complete site at the root. Finally, the log-likelihood of a tree

$$llh = \sum_{s=1}^{m} log(L(s)) \tag{2.8}$$

is the sum of the likelihoods of all alignment sites. We optimize Equation (2.8) for a given tree topology with respect to the branch lengths, all parameters of Q and α for the Γ distribution (if we model the rate heterogeneity) [108].

2.1.4. Parameter Optimization

One way to maximize the likelihood is setting the derivative of the log-likelihood function with respect to θ to 0 and solving the system of equations [108]. This, however, only works under simple substitution models like the Jukes-Cantor model and only for pairwise distances [108]. In practice, iterative numerical algorithms maximize the log-likelihood.

Branch length optimization capitalizes on the fact that only the likelihoods of ancestral nodes change when a branch length changes. Therefore, we can optimize independently, keeping the other branch lengths and parameters fixed [108]. One method for branch length optimization is Newton's method [36].

Since substitution model parameter changes typically affect the likelihood of all nodes in the tree, we can not optimize them one at a time. Yang [107] proposes one way of substitution parameter optimization in two phases. In the first phase, it optimizes the branch lengths using Newton's method with fixed substitution parameters. In the second phase, it applies a multivariate optimization framework such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) [38]. BFGS can optimize the substitution parameters with fixed branch lengths. This two-stage optimization works well, if the substitution rates of the substitution model and the branch lengths are not strongly correlated [108]. If we use a Γ distribution for modeling the rate heterogeneity, this might lead to such a correlation. In that case, the α parameter is often negatively correlated with the branch lengths. Then an alternative is to combine both optimization phases [108].

2.1.5. Tree Topology Optimization

In theory, we need to compare the likelihoods for all possible tree topologies to find the ML topology [108]. This problem is NP-hard due to the super-exponential growth of the number of different tree topologies. For n taxa there exist $\prod_{i=3}^{n} 2i - 5$ tree topologies, making the exhaustive topology search impractical for large sets of sequences [14]. An alternative approach are heuristic tree searches, which aim to find potentially better tree topologies, given a starting topology [108]. There are two categories of heuristic tree search.

The first category comprises agglomerative and divisive clustering algorithms. Starting from single sequences, agglomerative approaches merge closer sequences based on an optimality criterion (such as ML or MP). One example of an agglomerative heuristic tree search is the stepwise addition algorithm. It adds each sequence one by one into a tree. For each new sequence, the algorithm evaluates the optimality criterion at each position and inserts it into the branch yielding the best score. Divisive approaches work the other way around: They split the taxa into finer groups while optimizing the optimality function [108].

The algorithms of the second category use perturbations on a given tree topology to create new topology candidates. Following this perturbation, the algorithm decides which candidate to choose based on the optimality criterion. Three possible perturbations are Nearest Neighbor Interchange (NNI), Subtree Pruning and Regrafting (SPR), and Tree Bisection and Reconnection (TBR) moves. Figure 2.4 shows an example of NNI and SPR moves. Swapping subtrees at the internal branches from one side to the other (*B* and *D* or *B* and *C*) leads to different tree topologies. SPR prunes a subtree from the tree and reinserts it at another position. Finally, Figure 2.5 shows how TBR bisects a given tree and rejoins the trees to obtain a new tree topology.

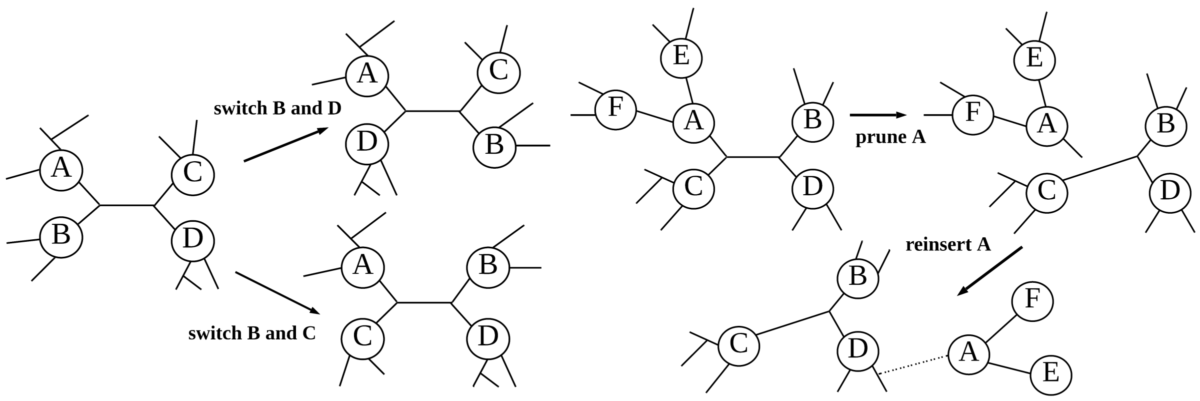


Figure 2.4.: Example of NNI (left) and SPR (right) based on Allen and Steel [4], p. 3f

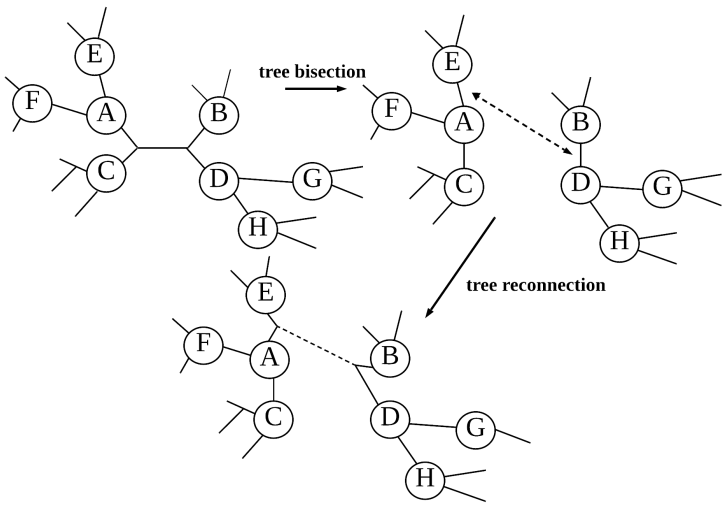


Figure 2.5.: Example of TBR based on Allen and Steel [4], p. 4

2.1.6. RAxML-NG

RAxML-NG is a phylogenetic inference tool using the ML optimization approach [58]. The tree search starts from a set of starting trees. By default, RAxML-NG infers the starting trees using random topologies and by inferring MP trees in a stepwise addition procedure. During the heuristic search for the ML tree, RAxML-NG optimizes the branch lengths, substitution model parameters, and topology. For branch lengths and the substitution model it uses Newton's method, Brents method [13], or an approximation of the BFGS for memory efficient computing called L-BFGS (limited memory BFGS) [63]. For topological optimization, it employs an adaptation of the topological perturbation technique SPR. RAxML-NG implements a fast variant of SPR which considers possible insertion points up to a certain maximum distance away from the point of pruning [58].

2.2. Phylogenetic Placements

In metagenomics, the accurate placement of new sequences within a given phylogenetic tree is a common use case. Metagenomics itself involves the investigation of genetic material directly from environmental or clinical samples. Examples of metagenomic projects are water quality monitoring from water samples [10], or the analysis of the soil microbiom [21].

Metagenomic samples are typically obtained via Next Generation Sequencing (NGS) techniques [74]. NGS can generate many cross-species short reads from the genetic material in the metagenomic specimens. The categorization of the reads requires a similarity analysis with known sequences. A common, similarity-based analysis tool is the Basic Local Alignment Search Tool (BLAST) [57]. BLAST has some drawbacks when it comes to the metagenomic use case: it can generate false positives [57], biological factors like parasitic DNA distort the results [67] and it does not provide any evolutionary information.

The phylogenetic placement provides another approach to the metagenomic use case. It is a method to place new sequences into already established phylogenies under an optimality criterion like ML. One tool for phylogenetic placement is the Evolutionary Placement Algorithm (EPA) [9], and its successor EPA-NG [6]. Figure 2.6 provides an overview of a phylogenetic placement process.

We call the sequences to place *Query Sequences* (QS) and the tree *Reference Tree* (RT). A sequence in the reference MSA is a *Reference Sequence* (RS). The input for EPA-NG is the RT with its substitution model, the MSA that served as the foundation for the tree, and the QS already aligned to the MSA. In the following section, we focus on the possibilities of aligning a new sequence to a fixed MSA using MAFFT.

2.2.1. Extending Multiple Sequence Alignments

MAFFT implements two ways to add new sequences in existing phylogenies [51].

First, MAFFT can add new full-length sequences by an adopted guide tree procedure (−−add-option). MAFFT computes a pairwise distance matrix with additional rows and columns for the set of QSs. Based on this matrix, MAFFT establishes a guide tree. It does not compute the alignment of a node in the guide tree if the children of the node are in the existing alignment or already aligned to the existing alignment [51].

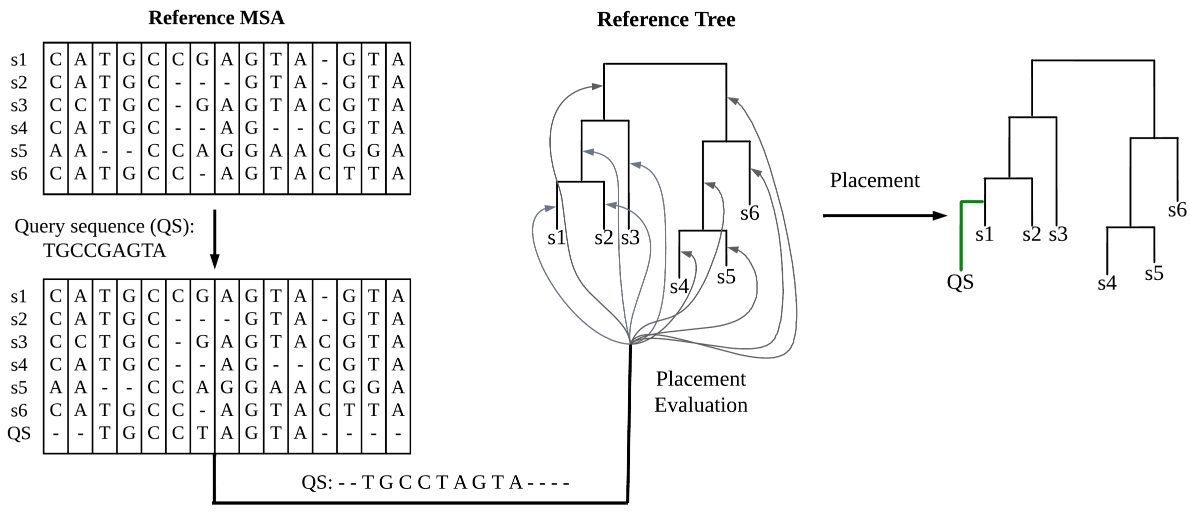

Reference MSA
s1 C A T G C C G A G T A - G T A
s2 C A T G C - - - G T A - G T A
s3 C C T G C - G A G T A C G T A
s4 C A T G C - - A G - - C G T A
s5 A A - - C C A G G A A C G G A
s6 C A T G C C - A G T A C T T A
Query sequence (QS): TGCCGAGTA
s1 C A T G C C G A G T A - G T A
s2 C A T G C - - - G T A - G T A
s3 C C T G C - G A G T A C G T A
s4 C A T G C - - A G - - C G T A
s5 A A - - C C A G G A A C G G A
s6 C A T G C C - A G T A C T T A
QS - - T G C C T A G T A - - - -
QS: - - T G C C T A G T A - - - -
Reference Tree
s1
s2
s3
s4
s5
s6
Placement Evaluation
Placement
QS
s1
s2
s3
s4
s5
s6

Figure 2.6.: Standard procedure of phylogenetic placement. The first step is the alignment of a new query sequence to the MSA. Then, the placement algorithm evaluates all branches as candidates for placement and places the QS into the branch which leads to the highest score.

The second approach is for short fragment QS. Because of the potential missing overlap between the QS and the MSA sequences, the computation of global pairwise distances is meaningless. Therefore, MAFFT uses the Smith-Waterman algorithm [90] to find the best local alignment between every single QS and all RSs. Afterward, MAFFT uses the same guide tree procedure as in the −−add-option for each QS independently. Finally, it combines all the different MSAs resulting from the QS alignments and inserts gaps as needed [51].

2.2.2. Evolutionary Placement Algorithm

The aligned QSs serve as input to EPA. Algorithm 1 shows the pseudocode for the insertion score computations of EPA based on Berger et al. [9]. Each QS will be placed in the branch yielding the highest insertion score. EPA computes a result matrix of the size *number of QS* $\times$ *number of branches*. It only optimizes the three branches adjacent to the insertion position, as depicted in Figure 2.7. We define the *Likelihood Weight Ratio* (LWR) as the ratio of the likelihoods of the different placements.

Algorithm 1 Evolutionary Placement Algorithm (EPA)

```
$R = \mathbb{R}^{|q| \times 2|r|-3}$                          ▷ Initialize result matrix, r/q as set of RS/QS
$stack \leftarrow \{b\}$                                 ▷ Select starting tip branch b at random
while $stack \neq \{\}$ do
    $branch \leftarrow stack.pop()$
    for $i \in \{1, ..., |q|\}$ do
        $proximal, distal, pendant \leftarrow RT.insert(q_i, branch)$
        $optimize(proximal, distal, pendant)$            ▷ Newton's method or fast heuristic
        $insertion\_score \leftarrow likelihood(RT)$
        $R(i, branch.index) \leftarrow insertion\_score$          ▷ Update result matrix
    end for
    if $branch.left\_neighbor.added = False$ then
        $stack.push(branch.left\_neighbor)$
        $branch.left\_neighbor.added \leftarrow True$
    end if
    if $branch.right\_neighbor.added = False$ then
        $stack.push(branch.right\_neighbor)$
        $branch.right\_neighbor.added \leftarrow True$
    end if
end while
```

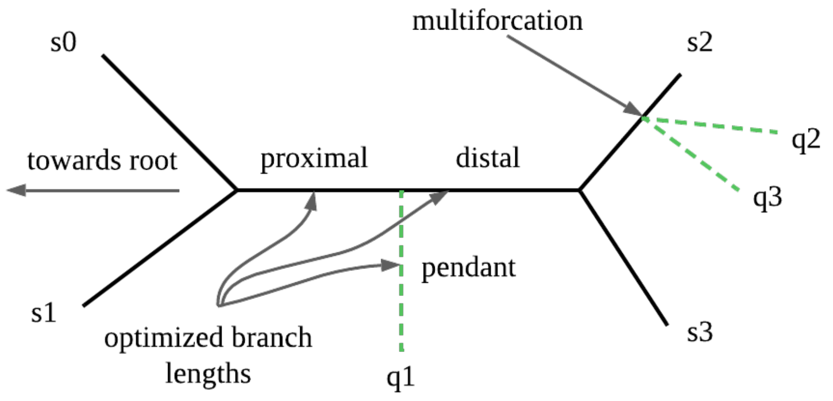


Figure 2.7.: EPA optimization procedure, based on Berger et al. [9]. EPA places QS $q1$ in the inner branch via the *pendant*, splitting the branch into *proximal*, which is oriented towards the (virtual) root, and *distal*, which is oriented away from it. EPA only optimizes proximal, distal, and pendant. When EPA inserts multiple sequences in the same branch, it produces multifurcations (like $q2$ and $q3$).

2.3. Properties of Phylogenies and Distances

In this final section on phylogenetics, we describe properties of phylogenies, namely the difficulty of their inference process and their statistical robustness. We furthermore address the measurement of topological distance through common tree distances, and we introduce a method for comparing the branch lengths of trees.

2.3.1. MSA Difficulty

The tree inference heuristics described in Section 2.1.5 only yield locally optimal trees. A consensus tree summarizes the information of multiple trees. It represents the splits that occur in the majority of the trees in a set. Some MSAs lead to a topologically similar set of trees, indicating a single likelihood peak and unambiguity of the analysis. In contrast, others result in topologically distinct, yet statistically equivalent locally optimal trees. This suggests a more complex likelihood surface. The MSA difficulty as defined by Haag et al. [42] quantifies this notion of difficulty of phylogenetic inference under the ML criterion on a given MSA. It ranges from a score between 0 (easy) and 1 (difficult). The authors introduce Pythia, a computationally efficient, machine learning-based tool for predicting the MSA difficulty. Pythia employs a LightGBM [53] boosting regressor as predictor. It uses various lightweight features from the MSA, including the site-entropy and site-over-taxa rate, along with features capturing topological variety derived from 100 MP trees inferred with RAxML-NG. Since ML tree inference is costly, prior knowledge of the inference difficulty before starting the process helps to refine the dataset or to have more informed decision-making about the tree inference [42].

2.3.2. Measures for Branch Support

After the inference of a phylogenetic tree, it is good practice to assess its statistical confidence [50]. Therefore this chapter describes measures of statistical confidence of a given tree topology.

2.3.2.1. Felsenstein Bootstrap Support

Felsenstein [31] proposes the concept of the SBS of a phylogenetic tree. The SBS is a means to calculate the statistical confidence in a given tree topology. Statistical bootstrap values in general estimate the variability of an unknown distribution based on a limited set of observed data [27]. The standard SBS procedure samples alignment sites of the MSA with replacement to create a set of replicate MSAs. For each replicate, the procedure infers the corresponding ML tree. Thus, the SBS procedure creates a set of bootstrap replicate ML trees. The frequency of a specific branch within the set of replicate ML trees determines the SBS value for that branch. The higher the SBS value for a branch, the more statistically robust it is. Common representations of the SBS values are percentage values between 0 and 100, or fractions between 0 and 1. In the following, we represent SBS values as percentages between 0 and 100.

Felsenstein and Kishino [32] provide a statistical interpretation for the SBS values. They propose to interpret the quantity of one minus the SBS value as a p-value for the null hypothesis of the branch *not* forming part of the true tree. However, Susko [101] shows that this approximation is too conservative as a p-value. Given an SBS value > 95, the probability of the branch not being in the true tree is substantially lower than 5%. Their experiments suggest that, depending on the nature of the true tree, an SBS between 70 and 85 corresponds to a 5% false positive bound.

The SBS procedure assumes, that each character is a random sample from a distribution of all possible character configurations. This distribution depends on the phylogeny at hand. Furthermore, the prior assumption of the phylogenetic bootstrap is the evolutionary independence of the MSA sites. Under this independence assumption, the MSA sites are independently and identically distributed (i.i.d.), which is a crucial condition for the validity of the bootstrap method [31]. The independence assumption is one of the major points of criticism of the bootstrap method. Furthermore, Galtier [35] proved that the bootstrap method can lead to an overestimation of the support of wrong internal branches. Despite the criticism, the SBS is a common tool for phylogenetic analysis [92].

2.3.2.2. Alternative Branch Support Measures

The SBS procedure is time- and resource-consuming, due to the high computational cost of conducting a phylogenetic tree inference on each replicate. Typically, we need to infer 100-500 replicate trees for the support values to stabilize [75]).

To alleviate this computational bottleneck, researchers have proposed a plethora of faster as well as alternative methods to measure branch support. For example, Stamatakis et al. [97] propose the Rapid Bootstrap (RB) as part of the phylogenetic inference tool RAxML [96] as a faster alternative to the SBS. RB uses a heuristic approach that implements a more superficial ML tree search for approximating the SBS values and reducing computational costs. On multiple large datasets Stamatakis et al. [97] show, that RB support values are highly correlated with the SBS values (Pearson correlation between 0.92 and 0.99). The ultrafast bootstrap UFBoot [70] and its current version UFBoot2 [47] entwine parametric and non-parametric aspects. The tree space sampling of UFBoot2 is substitution model-dependent (parametric). UFBoot2 combines it with a non-parametric bootstrap sampling of the MSA. UFBoot2 calculates easy-to-interpret, unbiased branch support values, the UltraFast Bootstrap Support (UFBS). According to the authors' experiments, UFBoot2 is extremely fast, as, on the median, it is 778 times faster than SBS and 8.4 times faster than RB.

Both RB and UFBoot2 employ an iterative approximation of their branch supports until they either meet a stopping criterion or reach a maximum number of iterations. Thus, the runtimes of these methods can vary, since they depend on the input MSA, which determines if and when the support value calculations will converge.

Anisimova and Gascuel [5] propose an alternative definition of branch support based on a parametric method for branch support estimation: The approximate Likelihood-Ratio Test (aLRT). The aLRT compares the two best NNI moves at each inner branch via aLRT test, to obtain a branch support. To accelerate the NNI likelihood evaluation, aLRT only optimizes the branches adjacent to the branch of interest. As aLRT is parametric, it can be sensitive to substitution model violations. Substitution model violations occur, for example, when we choose a substitution model that is too simple for the data at hand. To correct those violations, Guindon et al. [40] propose the Shimodaira–Hasegawa-like (SH-like) aLRT, which is a non-parametric version of the aLRT. The aLRT and SH-like aLRT only focus on local perturbations of the given

ML topology for which they calculate supports. This can induce overconfidence in branches if there are other highly likely, yet topologically substantially distinct tree topologies [40]. One recent example of such a tree space is a phylogeny of SARS-CoV2 genome sequences [72].

Despite the availability of these tools, SBS remains an important approach for measuring branch support [20, 1, 69]. Guindon et al. [40] propose to combine the SBS with the SH-like aLRT to obtain a holistic estimate of the branch robustness. UFBoot2 is less vulnerable to severe model violations than UFBoot [47]. Yet UFBoot2 still requires an additional step to check for such violations. SBS is inherently robust against model violations, as it is non-parametric.

2.3.3. Tree Distances

We compare phylogenetic trees using distance measures. The usage of both topological and branch length distance measures provides a holistic interpretation of the difference between two phylogenetic trees. Tree distances are a useful tool for capturing the variety of a set of trees. Furthermore, we can use them to analyze the effect of changes in the MSA on the resulting tree topology.

2.3.3.1. Robinson-Foulds Distance

The Robinson-Foulds (RF) distance is one common way to quantify the topological difference between phylogenetic trees [81]. The normalized Robinson-Foulds (nRF) distance provides an RF variant with a fixed value range. The nRF computes as

$$nRF(t_1, t_2) = \frac{|B_1 \cup B_2| - |B_1 \cap B_2|}{2(n-3)}, \tag{2.9}$$

where B_1 and B_2 are the sets of bipartitions of tree t_1 and t_2. The denominator normalizes the result such that nRF $\in [0, 1]$ by dividing the RF distance by the number of branches (or bipartitions) in an unrooted phylogenetic tree with n taxa. The (n)RF has the drawback of ignoring potential similarities between almost identical splits. This can lead to the edge case, in which a single different taxon position results in the maximum (n)RF distance [89].

2.3.3.2. Quartet Distance

Another topological distance measure, which is more robust to almost identical splits, is the quartet distance (QD) [19]. The QD quantifies symmetric differences between two phylogenetic trees by the number of *quartets* that are different [29]. A quartet is a four-taxon topology and the smallest number of taxa with more than a single distinct tree topology. Figure 2.8 depicts the three different tree topologies for unrooted binary trees that exist for a quartet. We compute the QD as

$$QD(t_1, t_2) = \frac{|Q(t_1) - Q(t_2)| + |Q(t_2) - Q(t_1)|}{2}, \tag{2.10}$$

where $Q(t_1)$ and $Q(t_2)$ are the quartets induced by the trees t_1 and t_2. One way of normalizing the QD is the division by the total number of quartets $\binom{4}{n}$ [98]. The normalized QD (nQD) between two random trees is on average $\frac{2}{3}$ [98]. This value is the reference value for the maximum nQD during interpretation and the basis for the scaling of the nQD between 0 and 1.

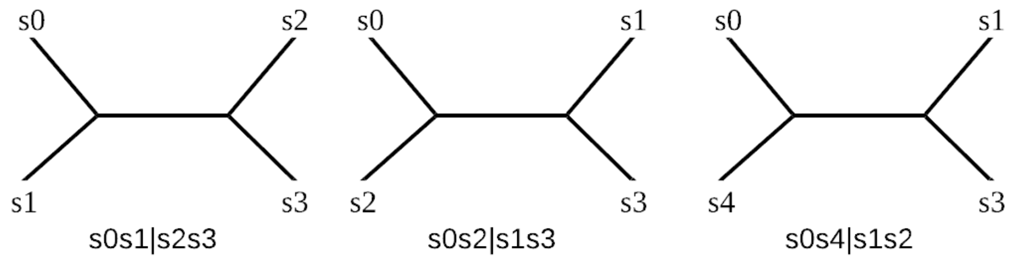


Figure 2.8.: All possible topologies for a four taxa, unrooted binary tree with its induced bipartitions.

2.3.3.3. Felsenstein Branch Score Distance

Kuhner and Felsenstein [60] propose the Branch Score Distance (BSD) as the square root of the sum of squared distances between the branches of two trees

$$BSD(t_1, t_2) = \sqrt{\sum_{i=1}^{N} (b_{1,i} - b_{2,i})^2}, \tag{2.11}$$

where $b_{1,i}$ and $b_{2,i}$ are the branch length of branch b_i in tree t_1 and t_2. We compare branches that are only present in one tree with a branch of length zero. The scaling of both trees such that their branch lengths sum up to one provides a way of normalizing the BSD [60].

3. Machine Learning

Two types of machine learning algorithms are relevant for this work: regression and classification. Modeling the relationship between a continuous dependent variable, called target, and independent variable(s) is a regression analysis [24]. If the target is discrete, the problem is a classification problem. A regression or classification uses the independent variables to predict the value of the target. There are different types of algorithms to perform regression or classification tasks. One type of algorithm that can handle both problems are Gradient Boosting Trees (GBTs). To date, GBT algorithms are in the top-scoring machine learning algorithms [33]. Consequently, we use a GBT algorithm for the regression as well as the classification tasks in this work.

In this chapter, we first introduce the concept of a decision tree as a foundational weak learner (Section 3.1) for GBTs. Subsequently, we delve into the utilization of sequentially trained decision trees, known as boosting algorithms (Section 3.2). Building upon this foundation, in Section 3.3 we describe how GBTs use gradient information to augment the prediction performance. This chapter is predominantly based on the book “The Elements of Statistical Learning” by Hastie et al. [45]. In this work, we use the Light Gradient Boosting Machine (LightGBM) implementation of GBTs. In Section 3.4 we present technical details of LightGBM as a specific GBT algorithm.

Finally, the chapter outlines the performance evaluation metrics we use for both the regression and classification problems (Section 3.5) and briefly describes a technique for providing explanations for predictions of machine learning models (Section 3.6).

3.1. Decision Trees

Decision trees are a type of machine learning algorithms that can solve regression and classification tasks. For the remainder of the chapter, we will denote the independent variables as a vector $x \in \mathbb{R}^N$, and the target variable as a scalar value $y \in \mathbb{R}$ for regression and $y \in \{-1, 1\}$ for (binary) classification.

Decision trees partition the data space of the independent variables into disjoint regions R_j and represent them by the j-th terminal node of the decision tree. For each region, the tree learns a constant γ_j that serves as the prediction for the target variable if a data point falls into region R_j. Thus, we formally define a decision tree T as

$$T(x, \theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j), \tag{3.1}$$

with θ being the parameter which comprises the J regions and their corresponding constant γ_j. The minimization of the empirical risk on the training data tuples (x_i, y_i) yields the parameter estimations

$$\hat{\theta} = \arg\min_{\theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, T(x_i, \theta)). \tag{3.2}$$

We can split this optimization problem into two parts: finding γ_j given R_j and finding R_j. One definition of γ_j for regression is the mean of all target variables y_i where $y_i \in R_j$. One approach to find R_j is a greedy top-down recursive partitioning that optimizes a splitting criterion until it fulfills a termination condition. Those splitting criteria enforce more and more homogeneous partitions (with respect to the target) of the data throughout the training process.

We start with the whole dataset as a partition and search for the best feature j of x and its split point s via the split criterion. For the original CART (Classification And Regression Tree) [12] regression tree algorithm, this criterion is the Sum of Squared Errors (SSE). This leads to the CART regression objective

$$\min_{j,s}(\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2) \tag{3.3}$$

with c_k as the mean of y_i for all $x_i \in R_k(j,s)$ and R_1 and R_2 as the two partitions that the split generates.

For a classification decision tree, one example of a split criterion is the misclassification error

$$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)), \tag{3.4}$$

where $k(m)$ is the most prevalent class in node m. R_m is the nodes region and N_m the total number of observations in R_m.

3.2. Boosting

Boosting is an ensemble method that combines multiple weak learners (such as decision trees) to a strong one. We consider a learner weak if the learner can produce a hypothesis that performs only slightly better than random guessing [84]. Boosting algorithms iteratively learn weak classifiers or regressors. After each iteration, the boosting algorithm weighs the training data. Wrong predictions get more weight than correct ones, which forces the subsequent weak learner to make up for the mistakes of his predecessors. Formally, we define a boosted tree model as the sum of a set of trees

$$BT(x) = \sum_{m=1}^{M} T(x, \theta_m) \tag{3.5}$$

which we obtain by the sequential optimization of

$$\hat{\theta}_m = \arg\min_{\theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i, \theta_m)) \tag{3.6}$$

where θ_m is the region set and the constants for the next tree, f_{m-1} the current model, and N the number of training data tuples. For a regression boosted tree model under the SSE loss, the solution to Equation (3.6) is the regression tree that predicts the residuals of the current model $y_i - f_{m-i}(x_i)$.

The solution of Equation (3.6) for a binary classification boosting algorithm under the exponential loss $L(y, f(x)) = e^{-yf(x)}$ is the tree that minimizes the exponentially weighted error rate:

$$\sum_{i=1}^{N} w_i^{(m)} I(y_i \neq T(x_i, \Theta_m)) \tag{3.7}$$

with $w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$.

3.3. Gradient Boosting

Gradient boosting algorithms are a specific type of boosting algorithms that use the gradient descent optimization technique to improve the performance of weak learners. They perform gradient descent on the objective Equation (3.6). Gradient descent solves the numerical optimization reformulation of the loss minimization in terms of a sum of M component vectors $f_M = \sum_{m=0}^{M} h_m$ with $h_m \in \mathbb{R}^N$. $h_m = -\rho_m g_m$ defines the step of steepest descent at each iteration with ρ_m as the scalar step size and $g_m \in \mathbb{R}^N$ as the gradient

$$g_m = \left[\frac{\delta L(y_i, f(x_i))}{\delta f(x_i)}\right]_{\forall i \in N} \tag{3.8}$$

of the loss $L(f_{m-1})$ of the current model on the training data. We define the step size ρ_m as $\rho_m = \arg\min_\rho L(f_{m-1} - \rho g_m)$. Given the gradient and the step size a gradient descent step for the current model is $f_m = f_{m-1} - \rho_m g_m$. The optimization uses the negative gradient since this is the direction of the steepest descent of the current loss $L(f)$. The gradient in Equation (3.8) is only defined at the training data points x_i. That is problematic since the goal is to generalize to unseen data. To overcome this problem, at each iteration, we fit a regression tree that approximates the negative gradient. In this context, the negative gradient serves as the *pseudo residuals* r_m.

The number of iterations M as well as the individual tree sizes J_m are hyperparameters. For large J_m and M the loss L on the training data can be arbitrarily small. This leads to the phenomenon called overfitting [46]. Overfitting occurs when a given model is too well optimized for the training data leading to a worse performance on unseen data. Another way to prevent the GBT model from overfitting is the introduction of a learning rate $v \in [0, 1]$. The hyperparameter v scales the individual contributions of each tree when updating the model. However, the hyperparameters M and v are not independent. Smaller values of v lead to a larger M for the same training loss. Therefore, careful hyperparameter tuning is crucial.

3.4. Light Gradient-Boosting Machine

LightGBM is an open-source GBT framework developed by Ke et al. [53]. It implements a histogram-based best split finding. LGBM calculates the feature value histograms and uses their bin values to find the best split, instead of trying all the possible feature values. Furthermore, it has two major techniques making it efficient and scalable to large datasets.

First is the Gradient-based One-Side-Sampling (GOSS). The authors noticed, that data instances with a larger gradient contribute more to the homogeneity of the splits during training. Thus, LGBM implements a downsampling procedure, which randomly drops those instances which have small gradients. This leads to fewer splits to evaluate during step 4 and therefore makes the whole algorithm more efficient.

The authors call the second technique Exclusive Feature Bundling (EFB). Due to the curse of dimensionality, high dimensional data spaces are sparse [54]. LGBM exploits the fact that in those high-dimensional spaces features often are mutually exclusive, which means that they are rarely non-zero simultaneously. Therefore, we do not lose information when bundling the corresponding features together in a new feature, the *exclusive feature bundle*. This reduces the complexity for histogram-based best split finding in step 5 from $O(N \times d)$ to $O(N \times B)$ where N is the number of training samples, d the dimensions, and B the number of bundles.

LGBM can speed up the training up to a factor of 20, depending on the dataset. Besides this, the accuracy on the benchmark classification tasks was comparable to major competitors such as XGBoost [17].

3.5. Prediction Quality

For the evaluation of the prediction quality of either regression or classification problems, there is a plethora of performance metrics. In this section, we introduce the regression and classification metrics we choose for model evaluation in our work.

3.5.1. Regression Metrics

Botchkarev [11] defines a typology of performance metrics for measuring the performance of regression algorithms. A generic formula for a regression performance metric for the target variable y and its prediction $\hat{y}$ is

$$\mathbb{M} = \mathbb{G}\{\mathbb{N}[\mathbb{D}(y_i, \hat{y}_i)]\}, \tag{3.9}$$

where $\mathbb{G}$ is an aggregation method, $\mathbb{N}$ a method of normalization and $\mathbb{D}$ a distance measure. For $\mathbb{G}$, we will use the arithmetic mean and – to overcome the outlier sensitivity – the median. $\mathbb{N}$ is the identity normalization since we are not comparing a series of predictions with different scales. For D, we choose a distance measure that is sensitive to outliers (squared distance) and two that are robust (difference and absolute difference). This leads to the following selection of regression performance evaluation metrics:

3.5.1.1. Mean Bias Error

The distance $\mathbb{D} = y_i - \hat{y}_i$ and the mean aggregation yields the mean bias error

$$MBE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i). \tag{3.10}$$

We use it to assess whether the prediction model is biased in the target prediction.

3.5.1.2. Mean/Median Absolute Error

The absolute distance $\mathbb{D} = |y_i - \hat{y}_i|$ and the mean aggregation yields the mean absolute error

$$MAE = \frac{1}{N}\sum_{i=1}^{N}(|y_i - \hat{y}_i|). \tag{3.11}$$

We use it for the assessment of the overall model quality on the original scale of the target variable. For the analysis of the influence of outliers, we select the Median Absolute Error (MdAE) as a supplement metric.

3.5.1.3. (Root) Mean Squared Error

The squared distance $\mathbb{D} = (y_i - \hat{y}_i)^2$ and the mean as aggregation function yields the mean squared error

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{3.12}$$

This metric penalizes large deviations more than small ones, which makes it sensitive to outlying large prediction errors. For interpretation purposes, we transform it back to the target scale by taking the square root, which yields the root mean squared error

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}((y_i - \hat{y}_i)^2)}. \tag{3.13}$$

3.5.2. Classification Metrics

We opt to employ the following four distinct classification metrics to assess the performance of a classifier.

3.5.2.1. Accuracy

We use the accuracy [80]

$$Acc = \frac{TP + FP}{TP + FP + TN + FN} \tag{3.14}$$

because it is easy to interpret. TP are true and FP false positives (threshold t exceeded). We define FP and FN analogously. Since the Acc is sensitive to class imbalances, we use other metrics as well.

3.5.2.2. Balanced Accuracy

We use the balanced accuracy

$$\mathrm{BAC} = \frac{\mathrm{recall} + \mathrm{specificity}}{2} \tag{3.15}$$

$$\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.16}$$

$$\mathrm{specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{3.17}$$

as a version of the Acc which is insensitive to class imbalances [22].

3.5.2.3. F1-score

The F1-score [80]

$$F1 = \frac{2 \times \mathrm{precision} \times \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} \tag{3.18}$$

$$\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{FP} + \mathrm{TP}} \tag{3.19}$$

serves as the harmonic mean between precision and recall, thus it penalizes FPs and FNs equally. In contrast to accuracy, the F1-score is robust against class imbalances. The values of the F1-score range from 0 (worst) to 1 (best).

3.5.2.4. ROC-AUC-score

The ROC-AUC-score [80] quantifies the Area Under the ROC (Receiver Operating Characteristic) Curve (AUC). In contrast to accuracy and the F1-score, it allows for evaluation without the need to set a class probability threshold, since it relies on the raw class probabilities. The best binary classifier yields an AUC-score of 1, while a random classifier yields an AUC-score of 0.5.

3.6. Explainable Machine Learning

Understanding a machine learning model and its predictions is crucial to gain insights into the problem it solves. A GBT algorithm, since it is a tree-based approach, provides information about the usage of split features. Those feature importances quantify, how often a GBT model uses a feature as a splitting feature, or how much the splits contribute to more homogeneous data partitions. While this helps explain the overall structure of the GBTs, it does not provide information on how the feature values influence the prediction outcome. One technique that does provide those explanations are Shapley values [62]. The main idea is to compare the absence of a feature *i* with its presence and quantify the effect on the prediction in the so-called Shapley values

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} (f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)), \tag{3.20}$$

with F as the set of all features, x_S representing the values of the input features in S and f as the machine learning model. These classic Shapley values require a retraining of the model for each feature subset S. Shapley sampling values approximate Equation (3.20) by sampling [99]. They approximate the removal of a feature by integration of the respective feature with sampling from the training dataset using Monte Carlo integration.

Building upon this, Lundberg and Lee [65] propose SHapley Additive exPlanations (SHAP), a model-agnostic, Shapley value-based framework for providing explanations for the predictions of machine learning algorithms. We can interpret SHAP values for an instance x as the amount ϕ_i a feature i contributed to the prediction of x, relative to the expected prediction of the model f. We use SHAP both to explain the global influence of feature values on the predictions and to obtain explanations for individual predictions. SHAP values inherit the property of efficiency from Shapley values [65]. Efficiency describes the fact, that the sum of all Shapley values is equal to the difference between the prediction for an instance x and the expected prediction of the model [62]:

$$\sum_{j=1}^{|F|} \phi_i = f(x) - \mathbb{E}_X f(X)) \tag{3.21}$$

We will use this property to aggregate individual SHAP values for features according to feature group membership for interpretation purposes.

4. Use Case I: Phylogenetic Placement Difficulty Prediction

4.1. Problem Definition

Our first use case of predictive modeling focuses on phylogenetic placements. A new sequence for a phylogenetic placement is typically placed in the branch of the RT with the highest LWR. If this LWR is evenly distributed across multiple branches, we consider this placement as difficult, as the optimal placement cannot be determined. We aim to predict this notion of difficulty associated with placing a new sequence into a provided phylogeny. In addition to this difficulty prediction, we analyze potential underlying reasons contributing to the uncertainty of placements. For all following analyses, we use EPA-NG as placement tool.

We define the difficulty of a placement based on the distribution of LWRs across the branches of the RT. An easily (or certainly) placeable sequence has a concentrated LWR on a single branch, whereas a highly challenging placement yields a more uniform spread of LWRs across multiple or (in the worst case) all branches. To quantify this notion of placement difficulty, we compute the normalized Shannon entropy [88] of the LWRs. To obtain the normalized entropy, we use the maximum placement entropy obtainable for each RT as a normalization constant, i.e., $log_2(\#branches)$ where $\#branches$ is the number of branches of the RT without the new sequence. This ensures a final target entropy $E \in [0, 1]$ as the placement difficulty metric. For easy placements, the entropy is low with values close to 0, whereas for difficult placements with a more uniform spread of LWRs, the entropy is closer to 1.

4.2. Experimental Setup

In this section, we present the MSAs and RTs we used for our analyses and explain how we generated sample placements (Section 4.2.1) as training data for the difficulty prediction. We furthermore describe the impact of our placement sampling strategy on the phylogenies and justify our employed procedure (Section 4.2.2).

4.2.1. Data Generation

Due to a lack of huge amounts of distinct metagenomic MSAs, we decided to train our predictor on non-metagenomic data. Additionally, we opted to rely on empirical rather than simulated MSAs. Trost et al. [103] demonstrate, that machine learning algorithms can distinguish between simulated and empirical MSAs with high accuracy. The authors conclude that sequence simulations do not fully capture all characteristics of empirical MSAs. Consequently, we used empirical MSAs available in TreeBASE [78] for our analyses. TreeBASE comprises representative MSAs that are commonly analyzed in phylogenetics, as it only contains MSAs of published studies. We decided to test our predictor on the limited amount of metagenomic MSAs to obtain a performance estimate in the intended domain.

Preparing the data, we noticed that approximately half of the MSAs of TreeBASE contain duplicate sequences, and decided to remove all duplicate sequences. We further selected the MSAs to include in our study based on the Pythia difficulty. To achieve a comprehensive representation of the difficulty spectrum in ML tree inference, we predicted the difficulty of each MSA in TreeBASE using Pythia. We selected MSAs of different levels of difficulty, thus encompassing easy and challenging MSAs. In Appendix A.1.1 we provide an overview of the difficulty distribution of the MSAs. Our final dataset comprises approximately 1800 distinct TreeBASE MSAs. For each MSA, we inferred 100 ML trees under the GTR+G substitution model using RAxML-NG to obtain the RT for placement. Since EPA-NG is only able to process unpartitioned substitution models, we filtered the TreeBASE for unpartitioned MSAs.

For the generation of QSs, we used a Leave-One-sequence-Out (LOO) approach. We sample up to 40 sequences from each MSA, which we deleted from the phylogeny (the MSA as well as the tree). 40 sequences is an arbitrary threshold which ensures that our data is not biased towards large phylogenies. Figure 4.1 summarizes the data sampling procedure. After the

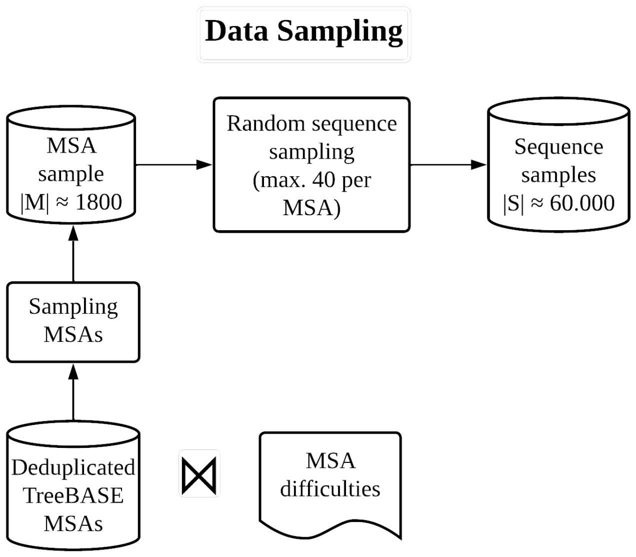

Data Sampling
MSA sample |M| ≈ 1800
Random sequence sampling (max. 40 per MSA)
Sequence samples |S| ≈ 60.000
Sampling MSAs
Deduplicated TreeBASE MSAs
⋈
MSA difficulties

Figure 4.1.: Overview of the data sampling strategy for the placement difficulty prediction.

deletion, we used EPA-NG to replace the QS in the now trimmed phylogeny, as Figure 4.2 depicts. We computed the ground truth placement difficulties as the normalized Shannon entropy of the placement LWRs. We observed that the entropy distribution is heavily right-skewed (see Figure 4.3).

In our set of placement difficulties, most samples are rather easy and only a few are very difficult. Only 2% of the 60 000 samples fall into the entropy range of 0.9 to 1.0. We decided to not balance out this distribution artificially since this would potentially lead to a loss of data and the balancing itself would be arbitrary. Thus, this skewed target distribution has to be considered during the prediction result interpretation.

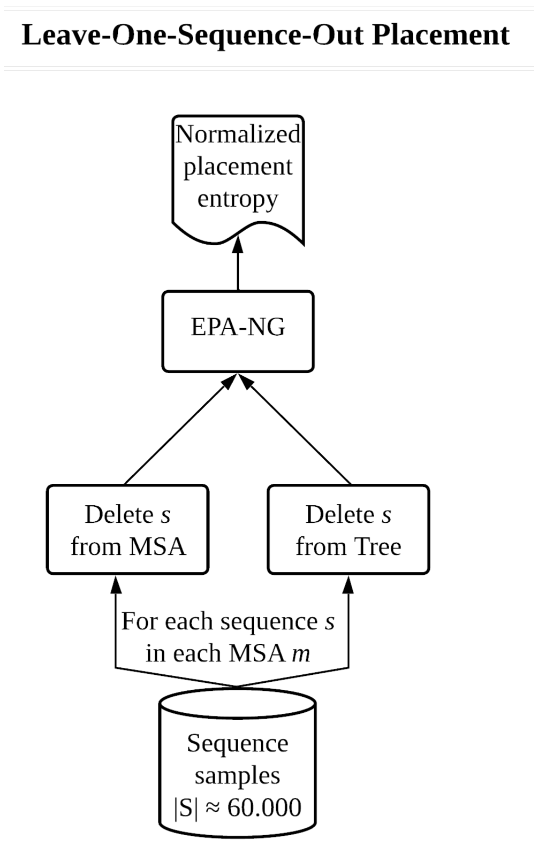

Leave-One-Sequence-Out Placement
Normalized placement entropy
EPA-NG
Delete *s* from MSA
Delete *s* from Tree
For each sequence *s* in each MSA *m*
Sequence samples |S| ≈ 60.000

Figure 4.2.: Overview of the target computation for the placement difficulty prediction.

4.2.2. Leave-One-Sequence-Out Impact

To justify the approach of the LOO placements for data generation, we analyzed its impact on the phylogenies. More specifically, we studied the influence of the LOO approach on the entropy of placements, as well as topological and branch length changes on the trees.

4.2.2.1. Re-estimation Target Leakage

The first aspect we analyzed as a potential consequence of the LOO is a change in the entropy of the placement. In particular, the hypothesis was, that we observe a lower entropy for the placement of a LOO sample if we do not perform a re-estimation of the MSA and the tree after the removal of it. The reason for this could be the presence of the removed sample during the inference process. According to our experiments, the opposite is the case. In a comparison of the LOO procedure once with the re-estimation of the MSA and the tree, and once without, the entropy tends to decrease by the re-estimation process. Figure 4.4 shows the histogram of the changes in entropy for approximately 2000 LOO placements.

We assume, that one potential reason for this phenomenon is that the changes depend on what particular sample we removed during the LOO process. Some sequences could have a greater influence on the tree structure than others. Figure 4.5 supports this hypothesis. There is a substantial standard deviation (up to 0.24) in the change of entropy depending on the LOO

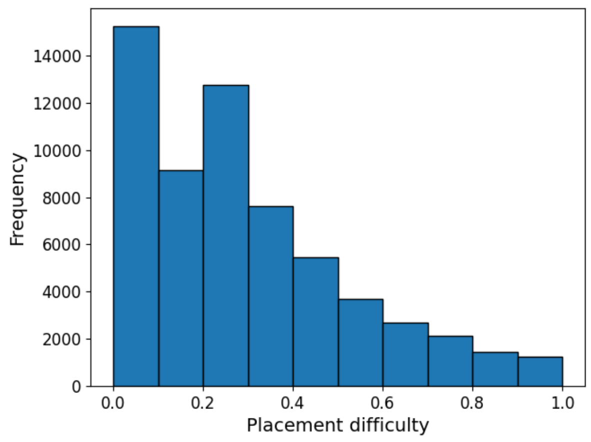


Figure 4.3.: Difficulty distribution of the datasets for the placement difficulty prediction. Sample size: 1800 MSAs, 60 000 LOO placements.

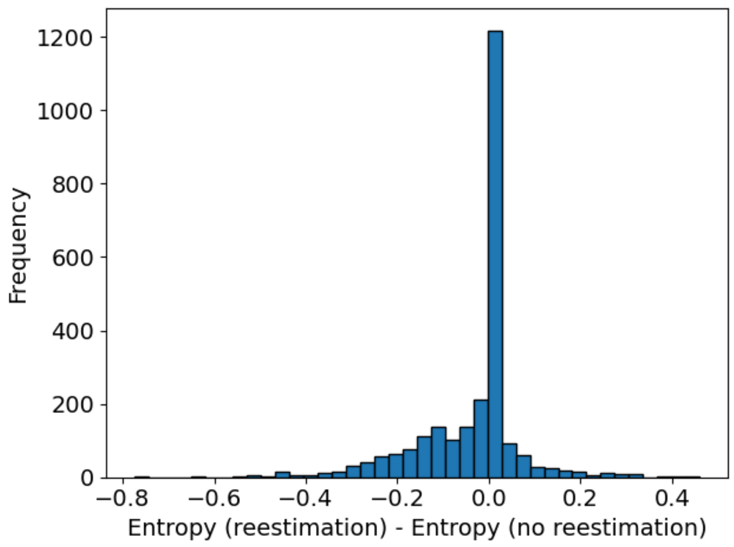


Figure 4.4.: Histogram of the differences between the entropy of LOO placements with and without re-estimation of the MSA and tree. The data includes 80 MSAs from the TreeBASE and their respective ML tree. We drew 30 LOO samples from each phylogeny. If there are less than 30 taxa in the phylogeny, each sequence once was the LOO sequence. The mean difference is -0.04 ($\sigma = 0.11$).

sample for some phylogenies. This means that, within the same TreeBASE MSA, the entropy change depends to some extent on the LOO sequence we choose. We were not able to find any significant correlations between the properties of the sequence we removed and the entropy change. Nonetheless, the LOO approach seems valid due to the small proportion of larger entropy deviations, as Figure 4.4 depicts.

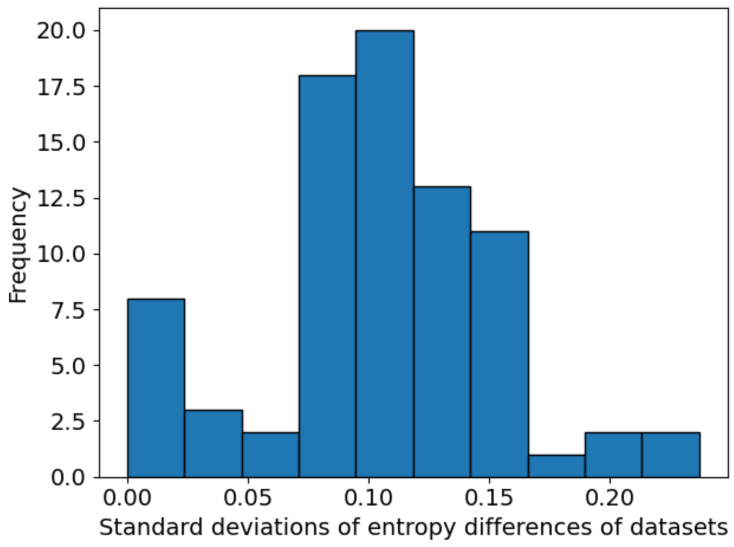
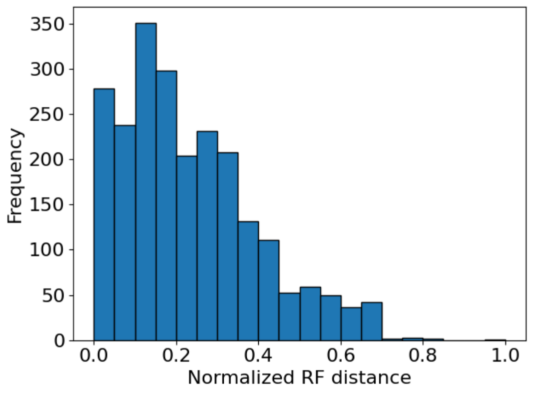


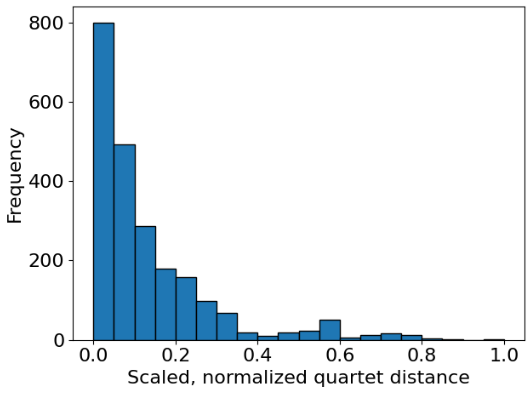
Figure 4.5.: Histogram of the standard deviations of the entropy difference for 80 datasets.

4.2.2.2. Tree Differences

Another aspect of the LOO procedure is its influence on the tree topology, as well as the tree branches. For topological changes, we analyzed the nRF (Figure 4.6a), as well as the nQD (Figure 4.6b) between the LOO procedure with and without re-estimation. While for the



(a) Histogram of the nRF distances between LOO procedure with and without re-estimation. The mean distance is 0.23.



(b) Histogram of the scaled nQDs between LOO procedure with and without re-estimation. The mean distance is 0.14.

Figure 4.6.: Effects of the LOO procedure on the corresponding RT topologies.

majority of MSAs, the tree topology appears to be only slightly affected with nRF and scaled nQD values between 0.0 and 0.3, we did observe substantial deviations up to 1.0 under both metrics. To determine whether this topological distance is caused by the re-estimation, we additionally inferred 1000 bootstrap replicate trees using the SBS procedure for each MSA. The hypothesis was that higher topological distances of the re-estimation can be attributed to higher uncertainty of the tree inference itself, which should lead to topologically more distinct bootstrap replicate trees. In fact, we found that the nRF distance in this context is highly correlated with the mean nRF distance of 1000 bootstrap trees (Spearman rank correlation: 0.72, p-value $\ll 1 \times 10^{-10}$). This suggests, that the SBS procedure and LOO both lead to a similar extent of change in tree topologies.

We used the maximum value of the nQD distance of 0.66 to rescale the nQD to values between 0 and 1. On average, the global change in topology as indicated by the mean nRF of 0.23 is larger than the local change as indicated by the mean nQD of 0.14. The changes in branch lengths are also negligible (Figure 4.7) with a mean BSD of 0.08.

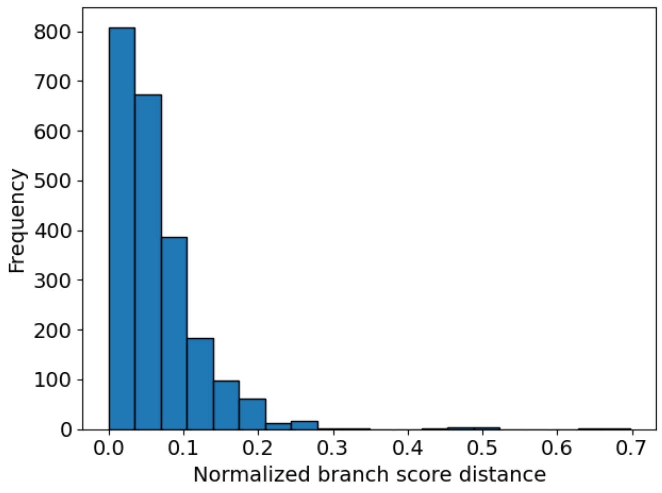


Figure 4.7.: Histogram of the BSD between LOO procedure with and without re-estimation. The mean distance is 0.08.

In conclusion, on average, there are only small topological differences (nRF: 0.23, nQD: 0.14) and small changes in the branch lengths (nBSD: 0.08) and we concluded that a re-estimation is not necessary. Thus, we used the LOO procedure without re-estimating the MSA and the tree as a technique for generating placement data.

4.3. Feature Engineering

For the prediction of the phylogenetic placement entropy, we computed over 300 features. We adopted an exploratory approach and categorized the features according to their data source, namely: QS, MSA, RT, or a combination thereof. We aimed to capture both the essential characteristics of the data sources and their inherent uncertainties. For instance, concerning the MSA, this uncertainty manifests as the gap fraction of the sequences or the site entropy. For the RT, we considered its SBS values in our analysis and the distances within the set of bootstrap trees.

Table 4.1 provides a comprehensive overview of the features. Most features are summary statistics of the data source characteristics. We calculated the following (central) moments as summary statistics: the minimum, maximum, mean, median, standard deviation or coefficient of variation, skewness, and kurtosis. We provide further information on selected experimental feature groups below.

Category	***Features***	***Description***
QS	Gap statistics	Statistics [1] of the set of all gaps in the QS.
	Longest gap	Longest continuous gap, relative to the length of the QS.
	Randomness measures	Uncertainty measures for binary data [7]. We encoded the sequence using UTF-8.
	Character fractions	Fractions of nucleotides/amino acids and consideration of ambiguity.
	Gap distribution	Statistics of the gap distribution over the sequence.
MSA	Gap statistics	Statistics of the gap distribution over the sites.
	Site entropy	Statistics of the normalized Shannon entropy of the sites.
	Pairwise sequence similarity statistics	Statistics of 15-mer similarity, hash Hamming distance, and Hamming distance.
	Sequence length/count	
	Character fractions	Statistics of the character composition of the sites.
	Randomness measures	Uncertainty measures for binary data [7]. We encoded the consensus sequence using UTF-8.
	Site-over-taxa ratio	Number of sites divided by the number of taxa.
	Invariant site fraction	Fractions of sites considered invariant according to different thresholds (0.5, …, 0.9).
Tree	Branch lengths statistics	Statistics of normalized branch lengths, inner branch lengths, and tip branch lengths.
	Depth	
	Imbalance ratio subtrees	Statistics of the ratios of the leaf numbers of the smaller and larger pending subtrees over all inner nodes.

	Closeness centrality statistics	Statistics of the closeness centrality [8] of nodes.
	Eigenvector centrality statistics	Statistics of the eigenvector centrality [44] of nodes.
QS and MSA	Hamming distance statistics	Statistics of Hamming distances between QS and MSA sequences.
	15-mer similarity statistics	Statistics of 15-mer similarity between QS and MSA sequences.
	Perceptual hash Hamming distance	Statistics of perceptual hash Hamming distances between QS and MSA sequences. Hash size 256-bit, linear greyscale encoding of the sequences. 256-bit hashes yielded the best results in the prediction.
	Perceptual hash k-mer similarity	Statistics of pairwise k-mer perceptual hash similarity with the MSA sequences.
	Perceptual hash LCS statistics	Statistics of the relative length of the LCS between the perceptual hashes of the QS and the MSA sequences.
	Image comparison statistics	Statistics of the pairwise PCA encoded Euclidean distances between the QS and the MSA sequences.
	Invariant site match statistics	Statistics of the matches between the QS and invariant sites of the MSA. Different thresholds t for the definition of invariant sites (0.5, …, 0.9).
	Invariant site mutation statistics	Statistics of the count of transitions and transversions induced by the QS at the invariant sites of the MSA.
	Invariant site fraction statistics	Statistics of the fractions of the QS characters in the invariant MSA sites.
MSA and tree	Parsimony mutation count statistics	Summary statistics of counts of parsimony mutations for each site.
	SBS statistics tree branches	Summary statistics of the SBS values of all branches for 1000 bootstrap trees created using RAxML-NG.

	SBS RF distance statistics	Summary statistics of the RF distances between the reference tree and the SBS trees.
	SBS trees unique topologies	Number of unique tree topologies in 1000 SBS trees inferred using RAxML-NG.
	Parsimony Support (PS) statistics	Statistics over the branch supports of the RT branches in 1000 parsimony starting trees inferred using RAxML-NG.
	Parsimony starting tree unique topologies	Number of unique tree topologies in 1000 parsimony starting trees inferred using RAxML-NG.
	Parsimony starting tree RF statistics	Summary statistics of the RF distances between 1000 parsimony starting trees inferred using RAxML-NG.
	Parsimony Bootstrap Support (PBS) statistics	Statistics over the branch supports of the RT branches in 200 parsimony bootstrap starting trees inferred using RAxML-NG.
	Parsimony bootstrap starting tree unique topologies	Number of unique topologies of 200 parsimony bootstrap starting trees inferred using RAxML-NG.
	Parsimony bootstrap starting tree RF statistics	Summary statistics of the RF distances between 200 parsimony bootstrap starting trees inferred using RAxML-NG.
MSA, tree, and QS	Summary statistics of low/high support branch bipartition distance to QS	Minimal Hamming distance between QS and parts of the most (un)supported central inner bipartition of the tree.

Table 4.1.: Feature overview.

[1] Minimum, maximum, median, mean, standard deviation/coefficient of variance, skewness, kurtosis

The k-mer similarities measure the degree of shared k-length subsequences between two sequences. The Hamming distance measures the number of differing positions between two equal-length sequences. For the k-mer similarities, we tested the values of $k \in \{5, 10, 15, 25, 50\}$ and only kept the k yielding the best performance.

Besides the conventional characteristics of the individual data sources, we computed novel features using multiple data sources at once.

In addition to a standard k-mer similarity and a Hamming distance between the QS and the MSA, we computed a perceptual hash Hamming distance. de Goër de Herve et al. [23] apply this distance measure to store and retrieve large DNA sequences. We encoded each DNA sequence as a linearly grayscale quadratic picture. Then, we performed a discrete cosine transformation and extracted the sign-only matrix from it. Finally, since the upper left entries in the matrix contain the most characteristic frequency components, we extracted the upper left square matrix of size 256 from it. The flattened matrix serves as a hash value for the sequence. A longer hash value did not improve the performance in our experiments. We compared the perceptual hash values of two sequences using the Hamming distance for the distance computation. We furthermore experimented with other measures of similarity or distances, such as the perceptual hash Longest Common Substring (LCS) or the perceptual hash k-mer similarity.

Another set of experimental features we used are summary statistics of the matches between the QS and the invariant sites of the MSA. We defined different thresholds t for the pureness of an MSA site to be invariant $(0.5, \ldots, 0.9)$. The idea is to quantify how well the QS matches conserved regions of the MSA. Furthermore, we computed the non-major character composition of the invariant MSA sites and to what extent the corresponding sequence character is present in those invariant sites.

A third group of features aims to incorporate all three sources of data: the QS, the MSA, and the tree. We intended to map a central certain or uncertain tree split back to the MSA and compare the sequence with both MSA splits. We iteratively searched for such a branch that is central to the tree. We defined a branch as central if the corresponding bipartitions are of roughly the same size. Besides being central, we require the branch to have either a high (≥ 80) or a low support (≤ 50). If we do not find a branch fulfilling this criterion, we relax the conditions gradually until we find one. After that search, we split the MSA according to the bipartition we found. Finally, we compare the QS with both parts of the MSA regarding similarity and distance.

The last novel feature group uses parsimony starting trees we computed using RAxML-NG to capture the tree space nature and uncertainty. RAxML-NG infers parsimony starting trees via a randomized stepwise addition order algorithm. This feature group is motivated by the success of Pythia which employs parsimony starting trees to obtain a lightweight approximation of the tree search space [42]. We computed the number of unique topologies in 1000 parsimony starting trees and summary statistics over the nRF distance among them as features. Furthermore, we adapted the SBS procedure to infer a parsimony starting tree for each replicate MSA instead of an ML tree. We call this procedure parsimony bootstrap. Again, we used the number of unique topologies and nRF statistics as features. Finally, we computed the Parsimony Support (PS) using the 1000 parsimony starting trees and the Parsimony Bootstrap Support (PBS) using 200 parsimony bootstraps and extracted summary statistics of the PS/PBS values.

For feature selection, we used Recursive Feature Elimination (RFE) [41] prior to the prediction pipeline, using a scikit-learn random forest [77] to select 15 features. More features did not lead to a substantially better prediction performance according to our experiments. The idea of RFE is to repeatedly delete a specific fraction (0.1 in our case) of the features with the smallest impact on the prediction. The aim was to find a good trade-off between the number of features and the performance of the predictor. This has several reasons. First, we want to avoid unnecessary computations. Furthermore, fewer features make the predictor more interpretable. Finally, we want to prevent the overfitting of our predictor to the training data. This can be the consequence of training a predictor with too little training data and too many, unnecessary features [46]. In that case, the predictor would generalize worse to unseen inputs.

4.4. Prediction Pipeline

Figure 4.8 depicts the training and testing procedure for the prediction of the phylogenetic placement difficulty. We used a holdout of 20% of the TreeBASE datasets for the estimation of

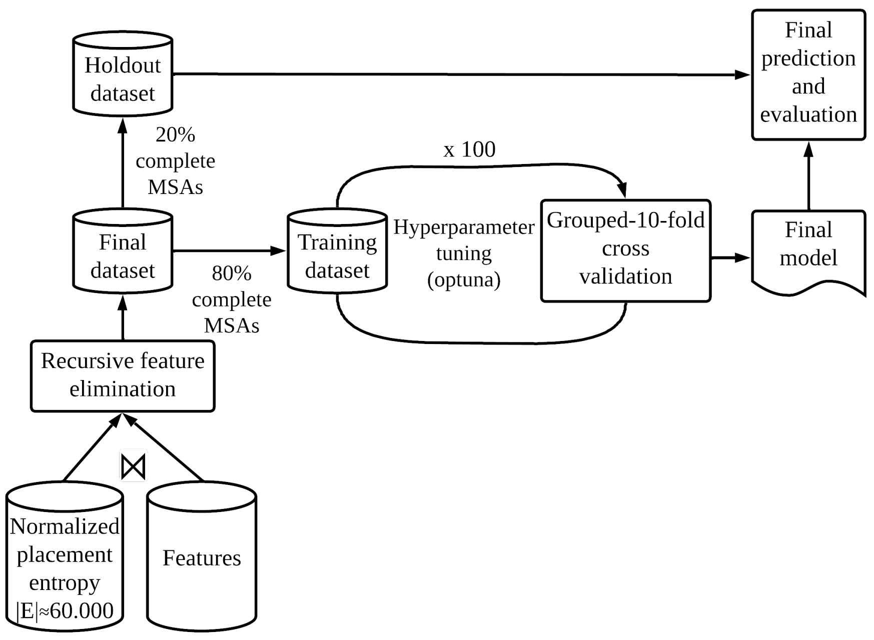

Holdout dataset
Final prediction and evaluation
20% complete MSAs
x 100
Final dataset
Training dataset
Hyperparameter tuning (optuna)
Grouped-10-fold cross validation
Final model
80% complete MSAs
Recursive feature elimination
⋈
Normalized placement entropy |E|≈60.000
Features

Figure 4.8.: Schematic overview of the training procedure for the placement difficulty prediction.

the predictor's performance on unseen data. To prevent overfitting during hyperparameter tuning, we used grouped 10-fold cross-validation [61] with the MSAs as a grouping feature. We repeated the whole process 10 times to average out effects due to random holdout sampling and obtain a more robust prediction performance estimate.

4.5. Evaluation

In this section, we evaluate our approach to placement difficulty prediction. We first summarize the results of our correlation analysis on the placement data in Section 4.5.1. In Section 4.5.2 we summarize the prediction performance, and finally in Section 4.5.3 we focus on the explainability of the predictions.

4.5.1. Correlation Analysis

We analyzed the final dataset for correlations between the features and the placement entropy. We used the Spearman rank correlation coefficient ρ [93] because of its robustness to outliers, and its ability to detect non-linear relationships. Table 4.2 shows correlations for which $|\rho| \geq 0.5$. For summary statistics that describe the same feature (e.g. bootstrap values, branch lengths, and RF distances) we only selected those statistics with the highest absolute ρ.

Feature	***Type***	***Spearmans ρ***	***p-Value***
MSA difficulty	MSA	0.69	$\ll 1 \times 10^{-10}$
Mean branch support	Tree	−0.69	$\ll 1 \times 10^{-10}$
Maximum nRF distance bootstrap	Tree	0.69	$\ll 1 \times 10^{-10}$
Std. branch length	Tree	−0.57	$\ll 1 \times 10^{-10}$
Skewness closeness centrality ($t := 0.7$)	Tree	−0.56	$\ll 1 \times 10^{-10}$
Kurtosis subtree imbalance ratios	Tree	−0.50	$\ll 1 \times 10^{-10}$
Std. minor characters on invariant sites ($t := 0.5$)	MSA	−0.50	$\ll 1 \times 10^{-10}$

Table 4.2.: Largest significant Spearman rank correlations between the placement entropy and the features.

The positive correlation of the MSA difficulty with the placement entropy implies that an increase in MSA difficulty is associated with an increase in placement entropy. Figure 4.9 visualizes that trend. Additionally, low mean branch support in the tree correlates with a higher placement entropy (see Figure 4.10). This suggests, that the statistical robustness of the tree is important for phylogenetic placements. The correlation with the maximum nRF distance among the bootstrap trees further supports this hypothesis.

In summary, based on our findings, we conclude that the primary correlations of the placement difficulty values are with the quantification of the difficulty inherent in the inference process and the uncertainty of the tree. These correlations are stronger than the correlations between the features of the QS and the placement difficulty. We refer to Appendix A.1.2 for further visualizations of feature and target correlations. We furthermore observe high correlations between the bootstrap support summary statistics and the MSA difficulty prediction by Pythia. We provide further details of these correlations in Appendix A.1.3.

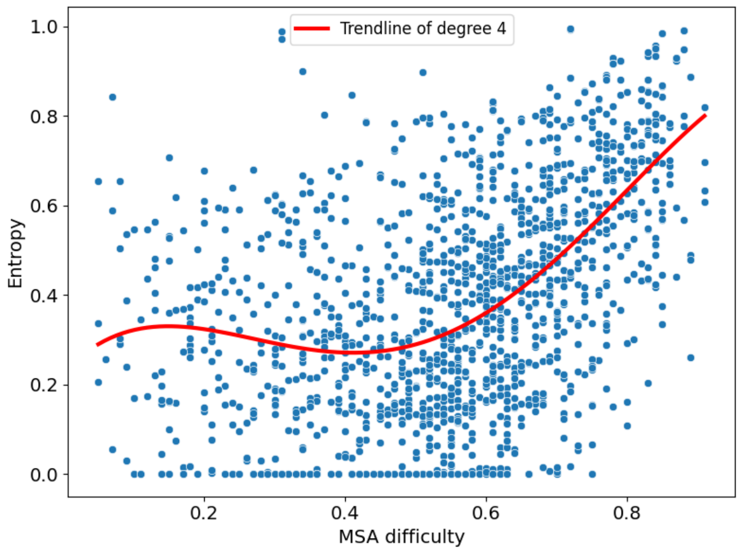


Figure 4.9.: Placement entropy as a function of MSA difficulty for 1500 randomly sampled LOO placements.

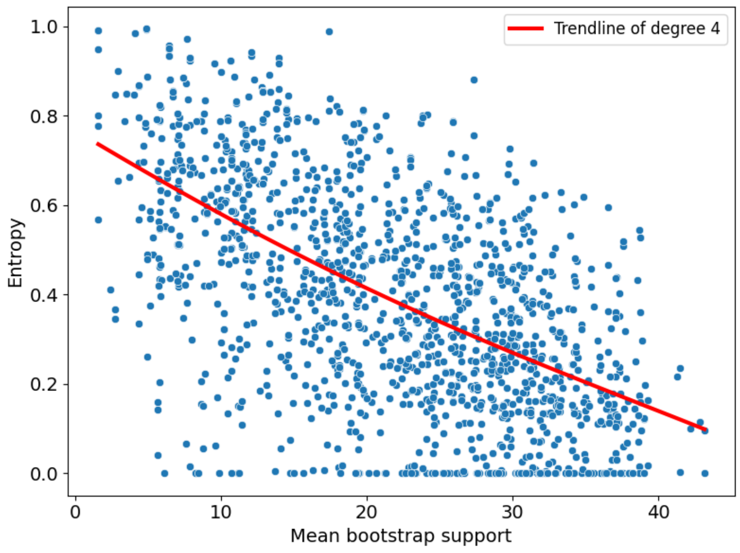


Figure 4.10.: Placement entropy as a function of mean SBS support for 1500 randomly sampled LOO placements.

4.5.2. Prediction Performance

Table 4.3 summarizes the phylogenetic placement entropy prediction performance metrics obtained by running our training pipeline as depicted in Figure 4.8 10 times. Since the SBS with 1000 replicates is computationally costly, thus inflating the runtime for feature computation substantially, we implemented two different predictors. One predictor incorporates features

Tool	*MBE*	*MAE*	*MdAE*	*RMSE*
Predictor$_{SBS}$	0.00 ± 0.01	0.13 ± 0.00	0.11 ± 0.01	0.16 ± 0.00
Predictor	0.00 ± 0.00	0.12 ± 0.00	0.10 ± 0.00	0.16 ± 0.00
Baseline	0.00 ± 0.04	0.19 ± 0.00	0.16 ± 0.01	0.24 ± 0.00

Table 4.3.: Placement entropy prediction evaluation for 10 random repeated holdouts of size 20%.

based on the SBS values (*predictor$_{SBS}$*), and one does not (*predictor*). Therefore, we removed the SBS features before the RFE for the latter. Since the performance of *predictor* is comparable to the performance of *predictor$_{SBS}$*, we conclude, that the remaining features can compensate for the missing SBS features. We compared both predictors against the baseline of the mean entropy prediction over all MSAs. Our predictors outperformed the baseline across all metrics. The MAE of 0.12 as well as the MdAE of 0.10 suggests a good overall performance when predicting the placement entropy. The low MBE value suggests, that the predictors are overall unbiased in their predictions. Since the MAE is smaller than the MdAE and the RMSE, there are some outliers in the prediction error. For a detailed distribution of the prediction errors, we refer to Appendix A.1.5.

4.5.3. Feature Importances and SHAP Values

Table 4.4 provides an overview of the feature importances of the final *predictor* without the SBS features. The feature importance indicates, how much an individual feature contributed to an improvement in the MAE during the training process. For a better overview, the features are grouped into four categories.

The most important group comprises features that quantify how the QS matches invariant sites of the MSA. This includes how well the QS character is represented in the corresponding MSA site. Furthermore, the transversion fraction of the mutations between the invariant sites and the QS is part of this group.

The second group captures summary statistics of the 15-mer similarity between the QS and all MSA sequences. One feature is relying on the 25-mer similarity of the perceptual hashes instead.

The third most important category uses parsimony (bootstrap) trees to capture features about the tree space's nature and uncertainty.

The least important final group contains different tree and MSA properties. Note that only this feature group includes RT features.

We conclude that, for the prediction of the placement entropy, the QS comparison with the MSA regarding the invariant sites of the MSA as well as k-mer similarities is far more

Feature	*Importance (%)*
Invariant MSA site composition comparison with QS	
Std. fraction of QS character in invariant MSA sites ($t := 0.7$)	24
Transversion mutation fraction QS on invariant MSA sites ($t := 0.7$)	8
Transversion mutation fraction QS on invariant MSA sites ($t := 0.5$)	7
Fraction of invariant MSA site matches ($t := 0.9$)	4
Min. fraction of QS character in invariant MSA sites ($t := 0.5$)	3
	46
Similarity QS and MSA sequences	
Mean 15-mer similarity	14
Kurtosis 15-mer similarity	6
Skewness 15-mer similarity	5
Std. 15-mer similarity	5
Kurtosis 25-mer similarity perceptual hash	2
	32
Tree space uncertainty	
Mean RF-distance 1000 parsimony starting trees	9
Number of unique tree topologies in 200 parsimony bootstraps	3
	12
Tree/MSA properties	
Skewness closeness similarity tree	4
Max. parsimony substitution frequency of MSA sites	3
Std. branch lengths	3
	10

Table 4.4.: Grouped feature importance of all 15 features.

important than the characteristics of the RT. Especially, the standard deviation of the query character fraction at invariant MSA sites makes up for almost a quarter of the normalized feature importance. Figure 4.11 depicts the SHAP values of the features on one of the 10 sets of MSAs for testing. It furthermore describes, how the feature values influence the predictions of the test placements. The SHAP values provide the benefit that we can use them to get hypotheses on *how* the features influence the placement difficulty.

A high mean nRF distance among the 1000 starting parsimony (first feature) trees contributes an additional difficulty of up to 0.2. Conversely, an elevation of the kurtosis in the 15-mer similarity (second feature) between the QS and the MSA reduces the entropy prediction. One plausible interpretation is, that a higher kurtosis characterizes a more sharply and pronounced distribution of similarity. That could lead to an easier placement. A high mean 15-mer similarity between the QS and the MSA sequences increases the predicted placement entropy up to 0.3. This could be due to a high similarity leading to a more ambiguous placement in the RT. Reducing the transversion fraction between the QS and invariant MSA sites (feature five) increases the predicted difficulty. Transversions are less likely than transitions in real data [108]. Having fewer mutations with a small probability might make the placement more ambiguous and thus increase the placement difficulty.

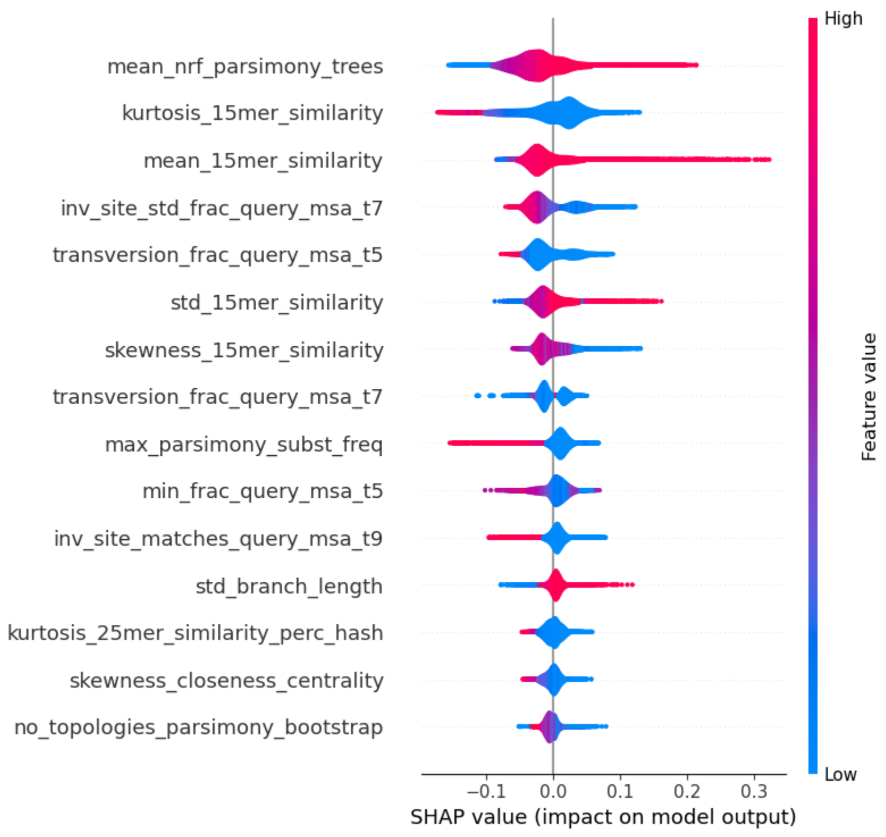


Figure 4.11.: SHAP value summary plot for the phylogenetic placement predictor.

Because of the moderate SHAP values of many of the features, we conclude, that all features are important for the predictor. For examples of individual Shapley value explanations for placements with high or low placement difficulty, we refer to Appendix A.1.4.

4.5.4. Performance on Metagenomic Placement Data

We trained and evaluated our predictor on non-metagenomic data. Therefore, we used 1000 placements of the Tara Oceans Project [100], the neotrop dataset [67] as well as the BV dataset [95] to test the prediction performance on real metagenomic data.

As explained in Section 4.2.1, we used whole sequences to replace them into the RT to generate the training data. However, the reads of metagenomic sequences are generally only a few hundred characters long [106]. To test whether using whole sequences results in worse performance on the real metagenomic test data, we simulated a metagenomic scenario on the LOO sequences. For this purpose, we employed the same target calculation as Figure 4.2 depicts, with the difference that we randomly sampled a read length between 200 and 450 for each LOO sequence. We furthermore sampled a random position. From this position, we extracted a continuous subsequence of the length previously sampled. Finally, we replaced each character that is not in the subsequence with gaps to obtain the simulated metagenomic read. We then trained a new predictor (*predictor$_m$*) on this new data to compare it to *predictor* which was trained on the whole LOO sequences.

Table 4.5 summarizes our results on the 3000 metagenomic placements. The MAE of *predictor* is comparable to the performance on the LOO sequences as Table 4.3 depicts. However, the MdAE is worse on metagenomic data and equal to the MAE suggesting that the distribution of the absolute errors is more symmetric on this metagenomic testing data.

Tool	***MBE***	***MAE***	***MdAE***	***RMSE***
Predictor	0.00	0.13	0.13	0.14
Predictor$_m$	0.01	0.14	0.15	0.16

Table 4.5.: Performance comparison of the placement difficulty predictor with (*predictor$_m$*) and without (*predictor*) training on simulated metagenomic sequences.

Interestingly, the performance of *predictor$_m$* is worse than the performance of *predictor* as indicated by all metrics having higher values. Either the metagenomic simulation procedure we employed is not suitable for generating metagenomic training data, or using whole sequences for the training is more beneficial than small simulated reads. Further investigating this observation remains subject to future work.

4.6. Conclusion

In this first use case, we demonstrated the application of ML techniques to predict the difficulty of phylogenetic placements.

We explored various methods for feature generation from QSs, MSAs, and phylogenetic trees (RT). We selected a subset of 15 from over 300 features using RFE. A predictor we trained using those features was able to achieve an MdAE of 0.1. We showed that on a small set of real metagenomic placements, our predictor still shows good performance. Our findings of the correlation analysis reveal that uncertainty in phylogenetic inferences correlates with uncertainty in the placement process as well.

The most important feature group of our predictor are matching and mutation statistics of the QS on invariant sites of the reference MSA. We furthermore showed how SHAP values are useful to provide explanations for the predicted placement difficulty. Additionally, the SHAP values can guide a hypothesis finding why specific features influence the prediction the way they do.

The non-metagenomically trained phylogenetic placement difficulty predictor (*predictor* in Section 4.5.4) is available as the command line tool BAD (`https://github.com/wiegertj/BAD`). BAD can provide SHAP value-based explanations for the difficulty of individual phylogenetic placements.

Future research could focus on a more thorough evaluation of the predictor performance on additional metagenomic data. Since training the predictor on simulated metagenomic sequences did not yield better performance on empirical metagenomic data, the incorporation of empirical metagenomic data in the training data could be a reasonable next step.

Another improvement would be the parallelization of the feature computation of BAD. The 1000 placements of the Tara Oceans Project used in Section 4.5.4 took approximately three hours to complete on a reference machine equipped with an Intel Xeon Platinum 8260 Processor (48 cores, 96 threads, 2.4 GHz) and 754 GB memory. The sequential sequence similarity computation alone accounts for 80% of the time-to-completion.

Pythia shows that parsimony starting trees are a powerful, lightweight approximation of the tree space under study. During the development of BAD, we used a similar approach to successfully replace SBS values in this use case. Subsequently, in the upcoming use case, we build upon this insight.

5. Use Case II: Bootstrap Support Prediction

This chapter is based on the following preprint:

> J. Wiegert, J. Haag, D. Hoehler, and A. Stamatakis. Predicting phylogenetic bootstrap values via machine learning. bioRxiv, 2024. doi: 10.1101/2024.03.04.583288.

5.1. Problem Formulation

This use case aims to enhance the efficiency of the analysis of the statistical robustness of branches in phylogenetic trees represented as SBS values. As the SBS has a high computational complexity, we provide a fast, machine learning-based SBS value predictor. The resulting predictors are available in a common command-line tool called EBG (`https://github.com/wiegertj/EBG`).

We address the challenge of predicting the SBS values via two distinct steps: regression and classification. In the regression step, EBG directly predicts the respective SBS values for *all* inner branches of a given ML tree. In the classification step, EBG then predicts the probability of the SBS value of each single inner branch to exceed a given SBS threshold using the regressor output as a feature. This classification step is based on the interpretation of one minus the SBS as p-value (see Section 2.3.2.1). As described in Section 2.3.2.1, the SBS is not unbiased but conservative. As values between 70 and 85 yield a 5% false positive bound, we use this SBS range for our classification step. More concretely, we predict the probability for each single branch i to exceed a specific SBS threshold t, i.e., $SBS_i > t$ with $t \in \{70, 75, 80, 85\}$.

In addition to the prediction itself, we estimate the uncertainty of the resulting predictions. The aim is to quantify whether and to what extent predictions are trustworthy.

5.2. Experimental Setup

5.2.1. Data

We used 1496 MSAs (DNA and AA) for training and evaluating EBG. We randomly sampled 220 additional MSAs for our final comparison with RB, UFBoot2, and SH-like aLRT. We selected MSAs representing distinct difficulty levels, thus encompassing easy and challenging MSAs.

For each MSA, we inferred 100 ML trees under the GTR+G substitution model using RAxML-NG. To obtain SBS “ground truth” values as a training target for EBG, we performed one SBS run with 1000 replicates for each MSA using RAxML-NG. Our final training dataset comprises approximately 80 000 inner branches and their corresponding SBS value.

5.2.2. Feature Engineering

To predict the SBS values, we computed a plethora of MSA features, as well as the respective best-known ML tree including the model parameter estimates. EBG uses 23 features for the prediction (Table 5.1).

Feature
Parsimony Bootstrap Support (PBS)
Parsimony Support (PS)
Normalized ML branch length
child inner branches
Skewness PBS
Mean Robinson-Foulds distance PB
Mean Parsimony Mutation per Side (PMS)
ML Branch length
Max. PSF
Coefficient of variation PMS
Max. PBS children*
Mean PBS parents
Max. PS children*
Branch number (ordered by level-order traverse)
Skewness PMS
Branch length ratio bipartition
Max. PS children*
Min. PS children
Std. PBS parent branches
Std. PBS child branches
Mean closeness centrality bipartition ratio
Min. PS children*
Min. PBS children*

*: weighted by branch length

Table 5.1.: Overview of the features subset used in EBG.

The majority of features are extracted from a set of parsimony starting trees we inferred using RAxML-NG (−−start-option). The first use case in this thesis demonstrated that by using computationally substantially less expensive parsimony starting trees, we can effectively compensate for missing SBS features. We therefore expect that parsimony-tree-based features are useful for predicting SBS values.

We calculated a set of 12 features based on parsimony trees. Those 12 features are subdivided into Parsimony Support (PS) and Parsimony Bootstrap Support (PBS) features. Figure 5.1 provides an overview of the feature computation.

The PS is the frequency of occurrence of an inner branch in 1000 parsimony starting trees inferred using the original MSA. According to preliminary experiments (see Appendix A.2.6), we set the number of inferred parsimony trees to 1000, as more than 1000 did not substantially

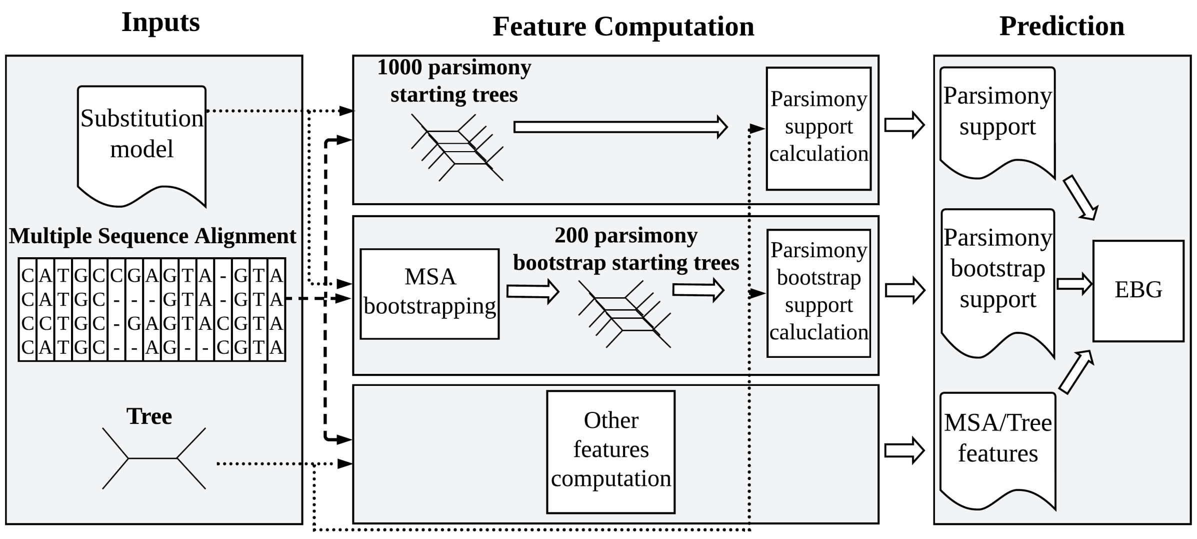

Inputs
Feature Computation
Prediction
Substitution model
Multiple Sequence Alignment
Tree
1000 parsimony starting trees
Parsimony support calculation
MSA bootstrapping
200 parsimony bootstrap starting trees
Parsimony bootstrap support caluclation
Other features computation
Parsimony support
Parsimony bootstrap support
MSA/Tree features
EBG

Figure 5.1.: Overview of the EBG feature generation and prediction.

improve prediction performance. The PBS employs a procedure that is highly similar to the SBS and uses replicate MSAs. The only difference is, that PBS computes a parsimony starting tree for each bootstrap replicate, instead of inferring an ML tree, yielding the computation substantially faster. To capture the variance of the respective bootstrap tree space, we also use the mean nRF distance between all PB trees as a feature. Again, according to preliminary experiments, more than 200 PBs did not improve predictor performance (see Appendix A.2.6).

In theory, we could lower the number of trees in dependence on the MSA difficulty, as the tree space for lower MSA difficulties is less complex. Thus, for lower MSA difficulties, a smaller number of parsimony trees might be sufficient to approximate the SBS values. However, because this might introduce additional uncertainty in the prediction process, we decided to keep the number of parsimony trees for both the PBS and the PS features fixed.

We expected that the P(B)S values of child and parent branches of the inner branch of interest are indicative of its SBS value. Therefore, we also included summary statistics for the P(B)S values of respective child and parent branches as features as well. Additionally, we computed summary statistics over the Parsimony Mutations per Side (PMS) as another group of features. Finally, we use the mean closeness centrality [8] of the two subtrees connected to the focal inner branch. The closeness centrality quantifies how densely connected the nodes of the tree are to each other. By taking the closeness centrality ratio of those subtrees, we aimed to capture if the branch connects two subtrees of different densities. Another feature, we refer to as the branch length ratio bipartition, represents the ratio between the sums of branch lengths of the two subtrees defined by the branch of concern.

The set of 23 features (Table 5.1) we used for EBG is a subset of a larger set of over 150 features we experimented with. This set consists of the PS/PBS features as well as all features described in Table 4.1 which are not dependent on a QS or SBS values. We reduced the initial set of features to the 23 features via RFE and a scikit-learn random forest. The aim was to find a good trade-off between the number of features and EBG performance for the same reasons as stated in the first use case (Section 4.3).

5.2.3. Prediction Pipeline

Similar to our first use case, we used a LightGBM tree-based boosting ensemble framework as a predictor, both for the regression and classification formulation. It is crucial to provide quantified trustworthiness for the predictions, as biologists need reliable SBS estimations in their studies. We provide a method to estimate the prediction uncertainty using quantile regression [56]. This approach estimates the conditional quantiles of the SBS value given the input features. By training the model to predict the 5th and 10th quantiles of the SBS value, we can provide lower bounds for the SBS value at 5% and 10%. We optimized the hyperparameters of all models in 100 trials using the hyperparameter optimization framework Optuna. The training pipeline is analogous to the pipeline presented in the first use case (Figure 4.8).

5.3. Evaluation

We evaluated the performance of the EBG regressor and classifier using different separations of our curated dataset of 1496 TreeBASE MSAs into training and testing MSAs. For both, the EBG regressor and classifier, we further tested the usage of EBG's uncertainty measures for quantifying the reliability of its predictions. We also compared EBG with RB, UFBoot2, and SH-like aLRT regarding prediction quality, and we compared EBG against its fastest competitors UFBoot2 and SH-like aLRT in terms of time-to-completion and accumulated CPU time. Furthermore, we analyzed the importance of the prediction features for EBG.

5.3.1. EBG Regressor Performance Evaluation

The EBG regressor predicts three values: In addition to the central SBS point estimate, EBG provides two SBS predictions that correspond to two lower bounds. One lower bound has a 5%, the other a 10% predicted probability that the SBS value is below the respective bound. On a random subset of 232 MSAs, EBG's central SBS estimate is highly correlated (mean Pearson correlation of $\mu = 0.91$, $\sigma = 0.05$) with the SBS values. For a more detailed view, we refer to Appendix A.2.3. For the evaluation of the EBG regressor's central SBS point estimate, we randomly sampled 20% of the 1496 MSAs as a holdout testing dataset and trained EBG on the remaining 80%. Table 5.2 summarizes the results of 10 such random holdouts. To assess the

Metric	***EBG*** ($\mu \pm \sigma$)	***Baseline*** ($\mu \pm \sigma$)
MBE	0.6 ± 0.3	0.1 ± 0.5
MAE	8.3 ± 0.2	13.8 ± 0.4
MdAE	5.0 ± 0.1	8.6 ± 0.5
RMSE	12.8 ± 0.2	20.5 ± 0.5

Table 5.2.: EBG regression performance for 10 repeated random holdouts against a parsimony bootstrap support of 200 replicate trees as the baseline. To infer those replicate trees, we first obtain 200 replicate MSAs by sampling the original MSA site-wise with replacement. Subsequently, we infer the corresponding parsimony (starting) tree using RAxML-NG to calculate the 200 replicate trees.

predictive power of EBG, we defined a baseline SBS value prediction based on the PBS of 200 replicates. If a branch is in the ML tree, but not in the PBS replicate tree set, we assigned a baseline prediction of zero to it, as this resembles the behavior of the SBS procedure. EBG outperformed this baseline across all four metrics. As the MBE indicates, the regressor exhibits no substantial systematic bias in either an over- or underestimation of the SBS values. The difference between MAE and MdAE, along with the higher RMSE value, suggests deviations of varying size between the EBG prediction and the true SBS value. The lower bound predictions of EBG can serve as a means to bound this prediction error.

Figure 5.2 illustrates, how the proximity of the lower bound predictions to the median prediction can be used to constrain the MdAE for the EBG median regression predictions. This approach effectively mitigates prediction uncertainty. As the lower bound predictions approach the median prediction more closely, the MdAE decreases. Thus, the inspection of the distance between the lower bound and the median SBS predictions limits and quantifies the prediction uncertainty.

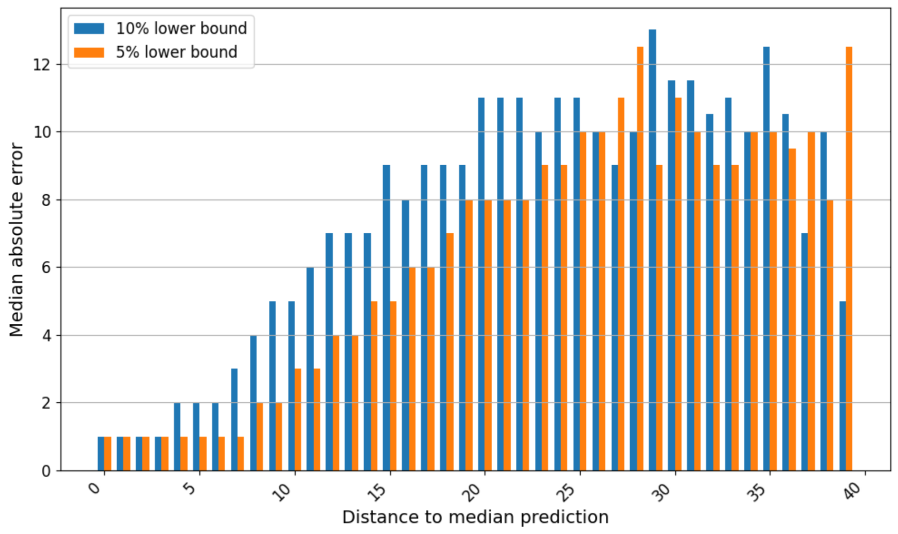


Figure 5.2.: Relationship between the distance of the lower bound predictions to the median prediction and their influence on the MdAE.

5.3.2. EBG Classifier Performance Evaluation

In analogy to the EBG regressor, we evaluated the EBG classifier based on 10 repeated holdout sets of 20%. Table 5.3 summarizes the resulting performance metrics for varying SBS thresholds t. As baseline performance, we used the different SBS thresholds $t \in \{70, 75, 80, 85\}$ on the PBS of 200 replicates. The baseline again is zero for all branches, that are present in the ML tree, but not in the PBS replicate trees. EBG outperformed the baseline for every metric.

In analogy to the lower bound prediction for the EBG regressor, EBG also provides a prediction uncertainty measure for its classifier. As the EBG classifier solves a binary classification problem, with one class defined as $SBS > t$ and the other as $SBS \leq t$, we can use the Shannon entropy of the two class probabilities to obtain a prediction uncertainty measure $u \in [0, 1]$. Here,

Metric	t := 70	t := 75	t := 80	t := 85	***baseline***
BAC	0.92 ± 0.00	0.91 ± 0.00	0.91 ± 0.00	0.92 ± 0.00	0.85 ± 0.01
AUC	0.97 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.00	0.85 ± 0.01
F1	0.90 ± 0.00	0.90 ± 0.00	0.89 ± 0.00	0.89 ± 0.00	0.82 ± 0.01

Table 5.3.: EBG classifier performance for different decision boundaries t. Mean and standard deviation of 10 repeated random holdouts against the baseline consisting of the PBS of 200 rounds.

$u = 0$ represents absolute certainty, and $u = 1$ corresponds to absolute uncertainty. Figure 5.3 provides an overview of the relationship between EBG classifiers with different SBS thresholds t and their Acc with increasing prediction uncertainty. Especially for the smaller uncertainties, a high class imbalance implies a better classification within the uncertainty interval of concern. Therefore, we decided not to compensate for class imbalances by using the BAC and instead used the Acc for this analysis. For cases with low $u \in [0.1, 0.3]$, the prediction Acc consistently remains at or above 90% across all SBS thresholds t. For moderate $u \in [0.4, 0.6]$, the Acc typically falls within the 80% to 90% range. We only observe prediction accuracies below 70% for $u > 0.8$.

In practical terms, when using EBG, we have the flexibility to determine an acceptable level of uncertainty to attain confident predictions. Thereby, users can tailor the approach to their specific needs and preferences.

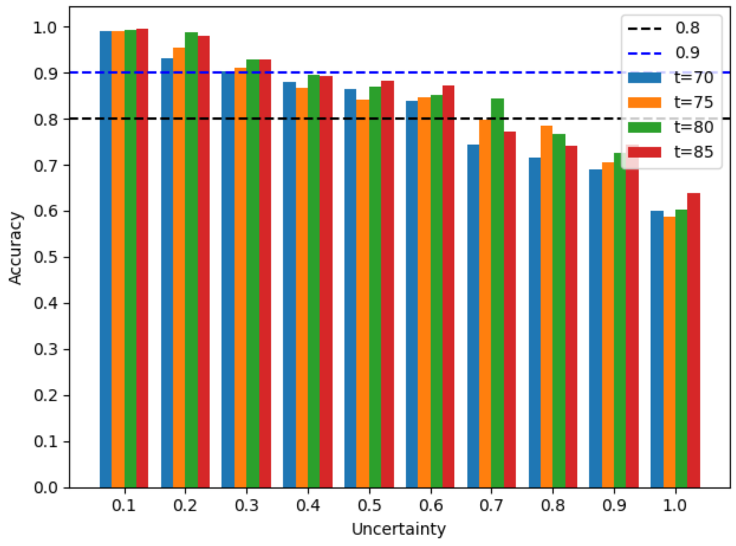


Figure 5.3.: Relationship between the prediction Acc of EBG classifiers and their prediction uncertainty for varying SBS thresholds t.

To demonstrate the performance of EBG on an edge case, we predicted the SBS values for an MSA that was shown to be difficult to analyze in terms of phylogenetic inference. Morel et al. [72] demonstrate the difficulties of obtaining a reliable phylogeny using ML inference on a set of SARS-CoV-2 genome sequences. This difficulty is caused by the combination of a large number of sequences and a relatively low mutation rate, resulting in numerous branches with low bootstrap values. For our experiment, we used a set of 1654 of those complete SARS-CoV-2 genomes, each with a length of 29 800 base pairs. We employed 100 RAxML-NG searches to determine the ML tree with the highest log-likelihood. We then performed an SBS run with 1000 replicates to establish the ground truth SBS values. This analysis shows, that only 13.9% of the inner branches of the ML tree yield an SBS value greater than 70 indicating an overall low support of the ML tree.

Predicting the branch support using our EBG regressor results in an overall good performance, with an MAE of 3, an MdAE of 0, and an RMSE of 9. Meanwhile, the EBG classifier, with an SBS threshold of $t := 70$, achieved a BAC of 0.82, an F1 of 0.77, and an AUC of 0.82. Even on an MSA that is known to be difficult-to-analyze, EBG can provide a good estimate of the SBS values for the corresponding ML tree. The prediction of the bootstrap support, using a mid-class laptop equipped with 4 cores (8 total threads) and 8 GB of memory, has a time-to-completion of approximately three hours. In contrast, computing the ground truth SBS values took 35 hours, utilizing 10 nodes of a large computing cluster, each equipped with an Intel Xeon Gold 6230 (20 cores, 40 threads, 2.1 GHz) and 96 GB memory. On the same amount of computing nodes, the RB computation using the RAxML with MPI-based parallelization took 5.5 hours, whereas the MPI-based UFBoot2 only took 31 minutes.

Another dataset that is known to be hard to analyze is the Internal Transcribed Spacer (ITS) 354 [39]. ITS 354 is a short alignment (348 MSA sites) extracted from the ITS genes from 354 maple tree genomes. Predicting the SBS values using the EBG regressor results in an MAE of 6, an MdAE of 4, and an RMSE of 9. Also on this difficult MSA EBG yields a good performance.

5.3.3. Performance Comparison with UFBoot2, SH-like aLRT, and Rapid Bootstrap

EBG, UFBoot2, and SH-like aLRT branch support values have all different interpretations. EBG approximates the SBS which is conservative, whereas UFBS values are unbiased [70]. SH-like aLRT is also conservative but not in the same sense as SBS, reasonable thresholds for SH-like aLRT can be between 0.8 and 0.9 [40]. Therefore, we needed a reliable ground truth phylogeny to compare all three branch supports against each other. Consequently, we performed the following performance comparison using simulated MSAs. We simulated a total of 979 DNA MSAs without gaps (i.e., without simulating indel events) based on TreeBASE trees using AliSim [66] under the GTR+G model. The corresponding tree (true tree) of the simulated MSAs served as the ground truth for our experiment. We compared the fraction of branches in the true tree for each branch support value of EBG, UFBoot2, and SH-like aLRT. A similar approach was used for the evaluation of the original UFBoot by Minh et al. [70]. For the following analyses, we used the IQ-TREE2 [70] implementation of the SH-like aLRT. The commands we used are shown in Appendix A.2.1. Figure 5.4 summarizes the results of our comparison analyses. An ideal branch support measure would yield the unbiased probability of the branch being in the true tree (dashed, red line). In our experiments, all three tools are too liberal with their estimation of the true branch probability. As already observed by Minh et al. [70], low SH-like aLRT values (< 50) are not informative concerning the true probabilities. For larger values, the SH-like aLRT behaves similarly to EBG. EBG is fairly unbiased for low support values (< 60). For branch supports > 60 it tends to overestimate the true probability of the branches being present in the true tree. In this experiment, UFBoot2 overestimated the true

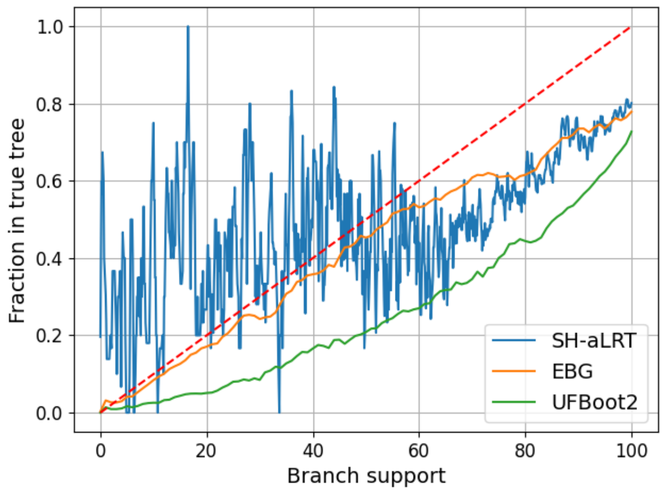


Figure 5.4.: Moving average with window size five of the fraction of branches in the true tree for all branch support values.

branch probability the most. Besides all three tools being overconfident in predicting the true support, EBG is the closest to the ideal branch support value line.

We also compared the performance of the different tools as a function of the difficulty of the simulated MSAs. We computed the BAC for specific branch support thresholds and compared them across all tools. The results suggest that EBG is best able to deal with varying MSA difficulties. We provide the detailed analyses in Appendix A.2.4.

Finally, we compared EBG directly with RB on empirical MSAs. This is possible since RB is highly correlated with the SBS values [97]. We randomly selected 220 MSAs (20% AA, 80% DNA) from TreeBASE and computed the ground truth SBS for each branch based on 1000 bootstrap replicates using RAxML-NG. Table 5.4 summarizes the respective classification and regression metrics for EBG and RB.

Tool	***MBE***	***MAE***	***MdAE***	***RMSE***	***BAC***	***F1***	***AUC***
EBG	0.0	8.7	6.0	12.8	0.89	0.87	0.89
EBG[*]	0.1	7.1	4.0	11.1	0.95	0.94	0.95
RB	0.0	4.5	2.0	8.0	0.97	0.96	0.97

[*] EBG with consideration of prediction uncertainty

Table 5.4.: Performance evaluation of the EBG regressor, EBG classifier with SBS threshold $t := 0.80$, and RB.

Based on our experimental findings, RB achieves the highest Acc when predicting SBS values. EBG's prediction performance falls short compared to RB, as evidenced by a higher MdAE (RB: 2.0, EBG: 6.0) and a lower BAC (RB: 0.96, EBG: 0.89). However, if we leverage EBG's prediction uncertainty measures, the performance becomes comparable, especially for the EBG classifier.

Note that EBG* in Table 5.4 summarizes the results for an uncertainty filtering of the predictions we conducted for both, the regression, and classification tasks. In the regression scenario, our focus is on branches where we expect the MdAE to be less than or equal to 8 (as Figure 5.2 depicts), that is, a 5% lower bound distance of 23 or less. This uncertainty filtering results in the exclusion of 28% of predictions that we deem too uncertain for consideration. For classification, we restrict our attention to predictions with an uncertainty level $u \leq 0.7$ (as Figure 5.3 depicts). That leads to excluding 21% of the predictions, that we consider as being too uncertain. While this approach may not provide predictions for every branch, it effectively constrains prediction errors. It represents a trade-off between the number of predictions we consider to be trustworthy and the level of certainty of those predictions.

In addition to the above accuracy analyses, we conducted a time-to-completion comparison of EBG with UFBoot2 and the SH-like aLRT (using IQ-TREE2) as fastest competitors. The SBS computation with RAxML-NG, UFBoot2 as well as the aLRT implementation of IQ-TREE2 can use multiple threads. Since the independent parsimony tree inferences necessary for the feature computation of EBG can also be parallelized straightforwardly, we performed our benchmark on a reference machine using multiple threads. This reference machine is equipped with an Intel Xeon Platinum 8260 Processor (48 cores, 96 threads, 2.4 GHz) and 754 GB memory. RAxML-NG and IQ-TREE2 provide the option to automatically determine the optimal number of threads for a given MSA, and we used this feature in both tools with up to a total of 60 threads. Figure 5.5 summarizes the results of the benchmark on the 220 empirical TreeBASE MSAs we used for the EBG/RB comparison. For the sake of simplicity, we define MSA size as

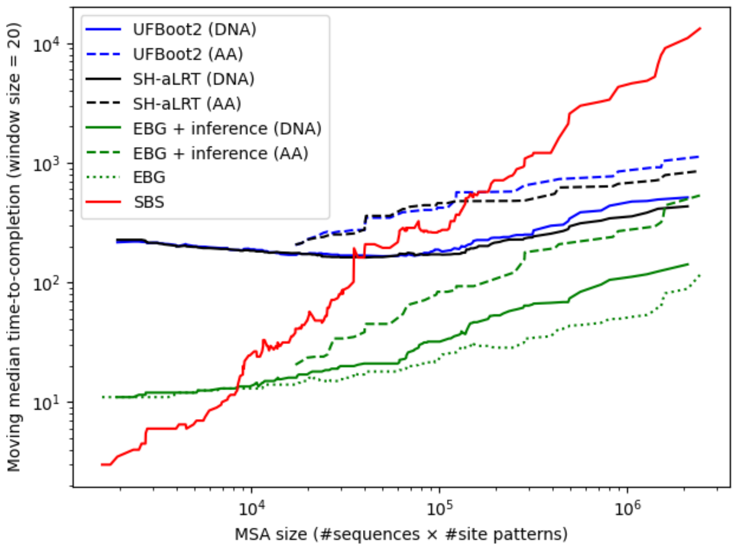


Figure 5.5.: Time-to-completion comparison for datasets of varying size with a moving average of window size 20.

the product of the number of sequences and the number of site patterns (unique MSA sites). Furthermore, we separately depict run times for AA (dashed lines) and DNA datasets (bold lines) to assess potential differences between data types. Since EBG requires an existing phylogenetic tree and substitution model parameters as input, we added the inference time of one adaptive RAxML-NG [102] search to the time-to-completion of the EBG prediction (EBG + inference). We observed that over all 220 MSAs EBG accounts for 19% and the inference for 81% of the total time-to-completion. SH-like aLRT and UFBoot2 have a similar time-to-completion both for DNA and AA MSAs. For 97% of the datasets (DNA and AA), EBG outperformed UFBoot2 in terms of time-to-completion, with an average speedup of 9.4. Considering MSAs of size ≥ 200 000, EBG yielded an average speedup of 2.8 in comparison to UFBoot2. With increasing MSA sizes, the time-to-completion differences between UFBoot2 and EBG gradually decrease for AA datasets. We do not observe an analogous trend for DNA data.

Additionally, we used a random sample of 84 MSAs from the total of 220 MSAs to assess the disparity in accumulated CPU time between EBG and UFBoot2. In the median, running EBG along with an adaptive RAxML-NG search requires 28% less accumulated CPU time in comparison to the corresponding UFBoot2 execution. However, in sum over all 84 MSAs EBG and the inference take 57% more accumulated CPU time (65 137 vs. 43 090 seconds) than UFBoots2. This suggests that there are some outliers where UFBoot2's time-to-completion is substantially smaller than EBG's time-to-completion when we include the ML inference time in the calculation. According to our analysis, this seems to primarily be the case with MSAs having a difficulty of ≥ 0.5 (see Appendix A.2.7 for a detailed visualization). Furthermore, we observe, that on the same 84 MSAs EBG only accounts for 4% of the accumulated CPU time while the remaining 96% are required for the adaptive RAxML-NG tree inference that is required as an input for EBG.

We conclude that it is challenging to devise a fair time-to-completion comparison between EBG and UFBoot2 as for EBG, we are undecided if the ML inference time should be included or not. In contrast, for UFBoot2 it represents an intrinsic requirement for the computation of support values as the ML search and the support calculations are necessarily intertwined. Hence, the above time comparisons reflect, to a large extent, a time comparison between the adaptive RAxML-NG and IQ-Tree ML search algorithms.

5.3.4. Feature Importances

Table 5.5 lists the five most important features of the EBG regressor. The feature importance quantifies, how much a feature contributes to achieving an improved prediction during the training. For a comprehensive overview of the feature importance, we refer to Appendix A.2.2.

Feature	***Importance in %***
PBS	82.2
PS	3.1
Normalized branch length	2.0
# child inner branches	1.7
Skewness PBS	1.5

Table 5.5.: Overview of the five most important features that EBG uses for the prediction and their respective importance in percent.

According to the feature importances of EBG, the PBS feature with a feature importance of 82.2% is by far the most important for the prediction of SBS values. We interpret the substantial importance of the PBS features with an analogy to ensemble methods in machine learning. These methods aggregate numerous weak learners, to create a robust, strong one. Similarly, we leverage the contributions of multiple “weak” parsimony inferences, to obtain a precise estimate of the SBS values.

5.4. Digression: Branch Prediction

Within the use case of SBS prediction, we explored the problem of branch prediction. More specifically, we tried to predict whether a branch we observe in the 1000 parsimony starting tree consensus tree occurs in the consensus tree of 100 ML tree searches we perform with RAxML-NG. We used 1060 TreeBASE MSAs with a total of 60 000 inner branches. 55% of the branches we extracted from the parsimony consensus tree are present in the ML consensus tree.

The feature set and the prediction pipeline are the same as for the SBS prediction. We performed RFE down to 10 features which, in our experiments, still yielded the best accuracy. We set the threshold of the class probability to 0.5 to predict whether the parsimony consensus tree branch is present in the consensus ML tree. Table 5.6 summarizes the prediction results for 10 random holdouts of size 20%. We established the baseline with the best split for the values of the minimum parsimony support of child branches on the training data. Our predictor outperforms the baseline in every metric and yields an overall good performance. Table 5.7 provides an overview of the feature importances of the final predictor. With a set of lightweight features, we can predict if a branch we observe in a parsimony consensus tree will occur in

Tool	***Acc***	***f1***	***AUC***
Classifier	0.79 ± 0.00	0.81 ± 0.01	0.88 ± 0.00
Baseline	0.65 ± 0.01	0.54 ± 0.01	0.68 ± 0.00

Table 5.6.: Test results for the branch predictor.

Feature	***Importance (%)***
Min. parsimony support child branches	80
Support parsimony consensus tree	12
Min. parsimony support child branches (weighted^2)	3
Skewness subtree imbalance ratio	1
Mean parsimony support child branches	1
Mean RF distance parsimony trees	1
Std. parsimony support child branches	0.5
Std. parsimony support child branches (weighted^2)	0.5
Skewness subtree imbalance ratio	0.5
Std. parsimony support parent branches	0.5

^{2} by the branch length

Table 5.7.: Feature importances for the branch predictor.

the ML consensus tree as well with good accuracy. This predictor could speed up the process of ML tree search by speeding up the candidate tree selection and, in general, provide a more guided topological search during the inference process.

5.5. Conclusion

We observe that RB remains the most accurate approximation method for the direct prediction of SBS values. We demonstrate how the EBG uncertainty measures can help to reduce the accuracy gap between EBG and RB.

Filtering the predictions of the EBG classifier to an uncertainty level $u \leq 0.7$ was able to close this gap in our experiments. EBG including the ML tree inference time requires substantially lower time-to-completion compared to the major competitor UFBoot2 with an average speedup of 9.4 ($\sigma = 5.5$). This speedup comes at the cost of an increase in total accumulated CPU time summed over all test MSAs of 57% due to outliers.

On 979 simulated MSAs, EBG, UFBoot2, and SH-like aLRT are generally too liberal and provide mixed results. Besides that, EBG is the closest to the ideal branch support value out of all three.

Currently, EBG implements the sampling part of the PB procedure in Python. The PB is a performance bottleneck that substantially contributes to the prediction time and often accounts for up to 70% of the overall time-to-completion of EBG excluding the ML inference time. Furthermore, EBG performs 204 RAxML-NG calls: One for each of the 200 parsimony bootstrap inferences, one for the parsimony inference, two for the support computation of both, and one for the nRF distance computation. Additionally, we store and retrieve the results of those calls in individual files, posing an I/O overhead. Unifying EBG's feature computation and the prediction as a command implemented within the RAxML-NG tool would likely result in a substantial speedup and streamline the entire prediction process. Since RAxML-NG is developed in our lab, the integration of EBG into RAxML-NG constitutes future work.

While EBG successfully establishes lower bounds for the EBG regression, we were not able to provide an upper bound to the SBS values. In our experiments, the attempts to optimize for any upper bound were unsuccessful, since they converged to the trivial upper bound of an SBS of 100. An upper bound for EBG regression would be useful for the construction of a comprehensive prediction interval. Future research could focus on the development of a method or model that is capable of reliably estimating upper bounds. This would provide a more informative prediction interval, which would be highly beneficial for the assessment of the range of potential SBS values.

Finally, we note that parsimony-based methods may experience a renaissance, since as we show here and as already demonstrated by the Pythia tool, they constitute the by far most important and computationally inexpensive feature for conducting predictions about ML method results.

6. Conclusion and Outlook

In this work, we explored two different use cases for predictive modeling in the context of phylogenetic inference and placement.

The first use case in Chapter 4, phylogenetic placement difficulty prediction, is insight-driven. Based on our definition of placement difficulty, we were able to provide a good predictor. The most predictive features are summary statistics of matches and mutations between the QS and conserved sites of the reference MSA. The predictor is available as command line tool BAD. BAD can provide SHAP value-based explanations for the difficulty of individual phylogenetic placements. A test we conducted on metagenomic test placements confirmed the performance of BAD we observed during training on non-metagenomic data. Thus, biologists conducting a metagenomic study can use BAD to hypothesize why specific sequences are easy or hard to place in a reference phylogeny. This use case demonstrates how techniques of explainable AI can be useful in analyzing otherwise inexplicable results. The major area of future research is incorporating metagenomic data into the BAD predictor training.

Chapter 5 presented the prediction of SBS values as our second use case for predictive modeling in phylogenetics. In contrast to the first use case, the SBS prediction is efficiency-driven. In our experiments, the SBS predictor EBG can beat state-of-the-art alternatives in terms of time-to-completion while predicting accurate branch support estimates. We were able to harness multiple parsimony tree inferences to obtain an accurate estimate of the SBS values. Thus, the idiom *strength in numbers* not only applies to EBG's GBT predictor combining multiple weak learners but also to the feature extraction from multiple parsimony trees. To support trustworthy decisions by biologists conducting phylogenetic analysis using EBG we put special emphasis on means to estimate the prediction uncertainty. Future research will focus on the efficient embedding of EBG in RAxML-NG, a more efficient feature generation, and the refining of the uncertainty estimates.

By successfully implementing both use cases, we reached our objective of exploring novel applications of predictive modeling in the realm of phylogenetic inference and placements. We have shown that ML can be used in these areas to elucidate previously unexplained mechanisms and help gain efficiency in common analysis setups.

Bibliography

[1] S. A. Ahmed, H. S. El-Mahallawy, S. F. Mohamed, M. C. Angelici, K. Hasapis, T. Saber, and P. Karanis. Subtypes and phylogenetic analysis of blastocystis sp. isolates from west ismailia, egypt. *Scientific Reports*, 12(1):19084, 2022. doi: 10.1038/s41598-022-23360-0.

[2] T. Aikenov, R. Hidayat, and H. Wicaksono. Power consumption and process cost prediction of customized products using explainable ai: a case in the steel industry. In *International Conference on Flexible Automation and Intelligent Manufacturing*, pages 1183–1193. Springer, 2023. doi: 10.1007/978-3-031-38165-2_135.

[3] B. Alberts, D. Bray, J. Lewis, M. Raff, K. Roberts, and J. Watson. *Watson Molekularbiologie*. Pearson, 6th edition, 2010. ISBN 978-3-86326-582-3.

[4] B. L. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Annals of Combinatorics*, 5(1):1–15, 2001. doi: 10.1007/s00026-001-8006-8.

[5] M. Anisimova and O. Gascuel. Approximate likelihood-ratio test for branches: A fast, accurate, and powerful alternative. *Systematic Biology*, 55(4):539–552, 2006. doi: 10.1080/10635150600755453.

[6] P. Barbera, A. M. Kozlov, L. Czech, B. Morel, D. Darriba, T. Flouri, and A. Stamatakis. EPA-ng: massively parallel evolutionary placement of genetic sequences. *Systematic Biology*, 68(2):365–369, 2018. doi: 10.1093/sysbio/syy054.

[7] L. Bassham, A. Rukhin, J. Soto, J. Nechvatal, M. Smid, S. Leigh, M. Levenson, M. Vangel, N. Heckert, and D. Banks. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2010. URL `https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762`.

[8] A. Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 2005. doi: 10.1121/1.1906679.

[9] S. A. Berger, D. Krompass, and A. Stamatakis. Performance, accuracy, and web server for evolutionary placement of short sequence reads under maximum likelihood. *Systematic Biology*, 60(3):291–302, 2011. doi: 10.1093/sysbio/syr010.

[10] K. Bibby, K. Crank, J. Greaves, X. Li, Z. Wu, I. A. Hamza, and E. Stachler. Metagenomics and the development of viral water quality tools. *npj Clean Water*, 2(1):9, 2019. doi: 10.1038/s41545-019-0032-3.

[11] A. Botchkarev. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14:45–79, 2019. doi: 10.28945/4184.

[12] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and regression trees*. Taylor & Francis, 1984. doi: 10.1201/9781315139470.

[13] R. P. Brent. An algorithm with guaranteed convergence for finding a zero of a function. *Comput. J.*, 14:422–425, 1971. doi: 10.1093/comjnl/14.4.422.

[14] L. L. Cavalli-Sforza and A. W. F. Edwards. Phylogenetic analysis: models and estimation procedures. *Evolution*, 21(3):550–570, 1967. doi: 10.2307/2406616.

[15] S. Challa and N. R. R. Neelapu. *Phylogenetic trees: applications, construction, and assessment*, pages 167–192. Springer International Publishing, 2019. doi: 10.1007/978-3-030-19318-8_10.

[16] M. Chantry, H. Christensen, P. Dueben, and T. Palmer. Opportunities and challenges for machine learning in weather and climate modelling: hard, medium and soft ai. *Philosophical Transactions of the Royal Society A*, 379(2194):20200083, 2021. doi: 10.1098/rsta.2020.0083.

[17] T. Chen and C. Guestrin. Xgboost: a scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, KDD '16, page 785–794. Association for Computing Machinery, 2016. doi: 10.1145/2939672.2939785.

[18] K. R. Clarke, L. Hor, A. Pilapitiya, J. Luirink, J. J. Paxman, and B. Heras. Phylogenetic classification and functional review of autotransporters. *Frontiers in Immunology*, 13, 2022. doi: 10.3389/fimmu.2022.921272.

[19] R. W. Crosby and T. L. Williams. A fast algorithm for computing the quartet distance for large sets of evolutionary trees. In *Bioinformatics Research and Applications*, pages 60–71. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-30191-9_6.

[20] N. CUI, B.-S. LIAO, C.-L. LIANG, S.-F. LI, H. ZHANG, J. XU, X.-W. LI, and S.-L. Chen. Complete chloroplast genome of salvia plebeia: organization, specific barcode and phylogenetic analysis. *Chinese Journal of Natural Medicines*, 18(8):563–572, 2020. doi: 10.1016/S1875-5364(20)30068-6.

[21] R. Daniel. The metagenomics of soil. *Nature Reviews Microbiology*, 3(6):470–478, 2005. doi: 10.1038/nrmicro1160.

[22] I. M. De Diego, A. R. Redondo, R. R. Fernández, J. Navarro, and J. M. Moguerza. General performance score for classification problems. *Applied Intelligence*, 52(10):12049–12063, 2022. doi: 10.1007/s10489-021-03041-7.

[23] J. de Goër de Herve, M. Kang, X. Bailly, and E. M. Nguifo. A perceptual hash function to store and retrieve large scale DNA sequences. *CoRR*, abs/1412.5517, 2014. doi: 10.48550/arXiv.1412.5517.

[24] D. Dickey. *Review of simple regression*, pages 1–36. Springer New York, 1998. doi: 10.1007/0-387-22753-9_1.

[25] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5):1792–1797, 2004. doi: 10.1093/nar/gkh340.

[26] A. W. F. Edwards. Statistical methods for evolutionary trees. *Genetics*, 183(1):5–12, 2009. doi: 10.1534/genetics.109.107847.

[27] B. Efron. Bootstrap methods: another look at the jackknife. *The Annals of Statistics*, 7(1): 1 – 26, 1979. doi: 10.1214/aos/1176344552.

[28] J. A. Eisen. Phylogenomics: improving functional predictions for uncharacterized genes by evolutionary analysis. *Genome Res*, 8(3):163–167, 1998.

[29] G. F. Estabrook, F. R. McMorris, and C. A. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Biology*, 34(2): 193–200, 1985. doi: 10.2307/sysbio/34.2.193.

[30] J. Felsenstein. Evolutionary trees from gene frequencies and quantitative characters: finding maximum likelihood estimates. *Evolution*, 35(6):1229–1242, 1981. doi: 10.1111/j.1558-5646.1981.tb04991.x.

[31] J. Felsenstein. Confidence limits on phylogenies: an approach using the bootstrap. *Evolution*, 39(4):783–791, 1985. doi: 10.2307/2408678.

[32] J. Felsenstein and H. Kishino. Is there something wrong with the bootstrap on phylogenies? a reply to hillis and bull. *Systematic Biology*, 42(2):193–200, 1993. doi: 10.1093/sysbio/42.2.193.

[33] M. Fernández-Delgado, M. Sirsat, E. Cernadas, S. Alawadi, S. Barro, and M. Febrero-Bande. An extensive experimental survey of regression methods. *Neural Networks*, 111: 11–34, 2019. doi: 10.1016/j.neunet.2018.12.010.

[34] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Zoology*, 20(4):406–416, 1971. URL `http://www.jstor.org/stable/2412116`.

[35] N. Galtier. Sampling properties of the bootstrap support in molecular phylogeny: influence of nonindependence among sites. *Systematic Biology*, 53(1):38–46, 2004. doi: 10.1080/10635150490264680.

[36] A. Galántai. The theory of newton's method. *Journal of Computational and Applied Mathematics*, 124(1):25–44, 2000. doi: 10.1016/S0377-0427(00)00435-0.

[37] S. Geisser. *Predictive inference.* Chapman and Hall/CRC, 2017. doi: 10.1201/9780203742310.

[38] P. E. Gill, W. Murray, and M. H. Wright. *Practical optimization.* Society for Industrial and Applied Mathematics, 2019. doi: 10.1137/1.9781611975604.

[39] G. W. Grimm, S. S. Renner, A. Stamatakis, and V. Hemleben. A nuclear ribosomal DNA phylogeny of acer inferred with maximum likelihood, splits graphs, and motif analysis of 606 sequences. *Evol Bioinform Online*, 2:7–22, 2007.

[40] S. Guindon, J.-F. Dufayard, V. Lefort, M. Anisimova, W. Hordijk, and O. Gascuel. New algorithms and methods to estimate maximum-likelihood phylogenies: assessing the performance of PhyML 3.0. *Systematic Biology*, 59(3):307–321, 2010. doi: 10.1093/sysbio/syq010.

[41] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46(1):389–422, 2002. doi: 10.1023/A:1012487302797.

[42] J. Haag, D. Höhler, B. Bettisworth, and A. Stamatakis. From easy to hopeless—predicting the difficulty of phylogenetic analyses. *Molecular Biology and Evolution*, 39(12):msac254, 2022. doi: 10.1093/molbev/msac254.

[43] M. Haber and J. Velasco. Phylogenetic Inference. In E. N. Zalta and U. Nodelman, editors, *The Stanford encyclopedia of philosophy.* Metaphysics Research Lab, Stanford University, Fall 2022 edition, 2022. URL `https://plato.stanford.edu/archives/fall2022/entries/phylogeneticinference`.

[44] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Sscience conference*, pages 11 – 15, 2008. URL `https://www.osti.gov/biblio/960616`.

[45] T. Hastie, R. Tibshirani, and J. Friedman. *Boosting and additive trees*, pages 337–387. Springer New York, 2009. doi: 10.1007/978-0-387-84858-7_10.

[46] D. M. Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004. doi: 10.1021/ci0342472.

[47] D. T. Hoang, O. Chernomor, A. von Haeseler, B. Q. Minh, and L. S. Vinh. UFBoot2: improving the ultrafast bootstrap approximation. *Molecular Biology and Evolution*, 35(2):518–522, 2018. doi: 10.1093/molbev/msx281.

[48] T. H. JUKES and C. R. CANTOR. Chapter 24 - evolution of protein molecules. In H. MUNRO, editor, *Mammalian Protein Metabolism*, pages 21–132. Academic Press, 1969. doi: 10.1016/B978-1-4832-3211-9.50009-7.

[49] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, A. Bridgland, C. Meyer, S. A. A. Kohl, A. J. Ballard, A. Cowie, B. Romera-Paredes, S. Nikolov, R. Jain, J. Adler, T. Back, S. Petersen,

D. Reiman, E. Clancy, M. Zielinski, M. Steinegger, M. Pacholska, T. Berghammer, S. Bodenstein, D. Silver, O. Vinyals, A. W. Senior, K. Kavukcuoglu, P. Kohli, and D. Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. doi: 10.1038/s41586-021-03819-2.

[50] P. Kapli, Z. Yang, and M. J. Telford. Phylogenetic tree building in the genomic age. *Nature Reviews Genetics*, 21(7):428–444, 2020. doi: 10.1038/s41576-020-0233-0.

[51] K. Katoh and M. C. Frith. Adding unaligned sequences into an existing alignment using MAFFT and LAST. *Bioinformatics*, 28(23):3144–3146, 2012. doi: 10.1093/bioinformatics/bts578.

[52] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14): 3059–3066, 2002. doi: 10.1093/nar/gkf436.

[53] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 3149–3157. Curran Associates Inc., 2017. doi: 10.5555/3294996.3295074.

[54] E. Keogh and A. Mueen. *Curse of dimensionality*, pages 314–315. Springer US, 2017. doi: 10.1007/978-1-4899-7687-1_192.

[55] M. Kimura. Estimation of evolutionary distances between homologous nucleotide sequences. *Proceedings of the National Academy of Sciences*, 78(1):454–458, 1981. doi: 10.1073/pnas.78.1.454.

[56] R. Koenker and G. Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978. doi: 10.2307/1913643.

[57] L. B. Koski and G. B. Golding. The closest blast hit is often not the nearest neighbor. *Journal of Molecular Evolution*, 52:540–542, 2001. doi: 10.1007/s002390010184.

[58] A. M. Kozlov, D. Darriba, T. Flouri, B. Morel, and A. Stamatakis. RAxML-NG: a fast, scalable and user-friendly tool for maximum likelihood phylogenetic inference. *Bioinformatics*, 35(21):4453–4455, 2019. doi: 10.1093/bioinformatics/btz305.

[59] L. S. Kubatko. *Inference of phylogenetic trees*, pages 1–38. Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-74331-6_1.

[60] M. K. Kuhner and J. Felsenstein. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Molecular Biology and Evolution*, 11(3): 459–468, 1994. doi: 10.1093/oxfordjournals.molbev.a040126.

[61] S. C. Larson. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology*, 22(1):45, 1931.

[62] S. Lipovetsky and M. Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001. doi: 10.1002/asmb.446.

[63] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi: 10.1007/BF01589116.

[64] J. B. Losos and R. E. Glor. Phylogenetic comparative methods and the geography of speciation. *Trends in Ecology & Evolution*, 18(5):220–227, 2003. doi: 10.1016/S0169-5347(03)00037-5.

[65] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, NIPS'17, page 4768–4777. Curran Associates Inc., 2017. ISBN 9781510860964.

[66] N. Ly-Trong, S. Naser-Khdour, R. Lanfear, and B. Q. Minh. AliSim: a fast and versatile phylogenetic sequence simulator for the genomic era. *Molecular Biology and Evolution*, 39(5):msac092, 2022. doi: 10.1093/molbev/msac092.

[67] F. Mahé, C. De Vargas, D. Bass, L. Czech, A. Stamatakis, E. Lara, D. Singer, J. Mayor, J. Bunge, S. Sernaker, et al. Parasites dominate hyperdiverse soil protist communities in neotropical rainforests. *Nature Ecology Evolution*, 1(1):0091, 2017. doi: 10.1038/s41559-017-0091.

[68] J. E. McCormack, S. M. Hird, A. J. Zellmer, B. C. Carstens, and R. T. Brumfield. Applications of next-generation sequencing to phylogeography and phylogenetics. *Molecular Phylogenetics and Evolution*, 66(2):526–538, 2013. doi: 10.1016/j.ympev.2011.12.007.

[69] M. S. Megan L. Porter, Holly Lutz and R. A. Chong. The complete mitochondrial genomes and phylogenetic analysis of two nycteribiidae bat flies (diptera: Hippoboscoidea). *Mitochondrial DNA Part B*, 7(8):1486–1488, 2022. doi: 10.1080/23802359.2022.2107450.

[70] B. Q. Minh, M. A. T. Nguyen, and A. von Haeseler. Ultrafast approximation for phylogenetic bootstrap. *Molecular Biology and Evolution*, 30(5):1188–1195, 2013. doi: 10.1093/molbev/mst024.

[71] D. Moreira. *phylogenetic tree*, pages 1885–1887. Springer Berlin Heidelberg, 2015. doi: 10.1007/978-3-662-44185-5_1208.

[72] B. Morel, P. Barbera, L. Czech, B. Bettisworth, L. Hübner, S. Lutteropp, D. Serdari, E.-G. Kostaki, I. Mamais, A. M. Kozlov, P. Pavlidis, D. Paraskevis, and A. Stamatakis. Phylogenetic analysis of SARS-CoV-2 data is difficult. *Molecular Biology and Evolution*, 38(5):1777–1791, 2020. doi: 10.1093/molbev/msaa314.

[73] M. Nei. *Molecular evolutionary genetics.* Columbia University Press, New York Chichester, West Sussex, 1987. doi: 10.7312/nei-92038.

[74] P. J. Oefner, L. Liu, Y. Li, S. Li, N. Hu, Y. He, R. Pong, D. Lin, L. Lu, and M. Law. Comparison of next-generation sequencing systems. *Journal of Biomedicine and Biotechnology*, 2012: 251364, 2012. doi: 10.1155/2012/251364.

[75] N. D. Pattengale, M. Alipour, O. R. P. Bininda-Emonds, B. M. E. Moret, and A. Stamatakis. How many bootstrap replicates are necessary? In *Research in Computational Molecular Biology*, pages 184–200. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-02008-7_13.

[76] T. Pearson, H. M. Hornstra, J. W. Sahl, S. Schaack, J. M. Schupp, S. M. Beckstrom-Sternberg, M. W. O'Neill, R. A. Priestley, M. D. Champion, J. S. Beckstrom-Sternberg, G. J. Kersh, J. E. Samuel, R. F. Massung, and P. Keim. When outgroups fail; phylogenomics of rooting the emerging pathogen, coxiella burnetii. *Systematic Biology*, 62(5):752–762, 2013. doi: 10.1093/sysbio/syt038.

[77] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Édouard Duchesnay. Scikit-learn: machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. doi: 10.5555/1953048.2078195.

[78] W. H. Piel, M. J. Donoghue, and M. J. Sanderson. Treebase v. 2: a database of phylogenetic knowledge. *e-BioSphere 2009*, 2009.

[79] D. Posada and K. A. Crandall. MODELTEST: testing the model of DNA substitution. *Bioinformatics*, 14(9):817–818, 1998. doi: 10.1093/bioinformatics/14.9.817.

[80] G. Rebala, A. Ravi, and S. Churiwala. *Classification*, pages 57–66. Springer International Publishing, 2019. doi: 10.1007/978-3-030-15729-6_5.

[81] D. Robinson and L. Foulds. Comparison of phylogenetic trees. *Mathematical Biosciences*, 53(1):131–147, 1981. doi: 10.1016/0025-5564(81)90043-2.

[82] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987. doi: 10.1093/oxfordjournals.molbev.a040454.

[83] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975.

[84] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990. doi: 10.1007/BF00116037.

[85] R. Schwartz and A. A. Schäffer. The evolution of tumour phylogenetics: principles and practice. *Nature Reviews Genetics*, 18(4):213–229, 2017. doi: 10.1038/nrg.2016.170.

[86] A. Scott and D. Baum. Phylogenetic tree. In R. M. Kliman, editor, *Encyclopedia of evolutionary biology*, pages 270–276. Academic Press, 2016. doi: 10.1016/B978-0-12-800049-6.00203-1.

[87] P. H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793, 1974. ISSN 00361399.

[88] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948. doi: 10.1002/j.1538-7305.1948.tb01338.x.

[89] M. R. Smith. Information theoretic generalized Robinson–Foulds metrics for comparing phylogenetic trees. *Bioinformatics*, 36(20):5007–5013, 2020. doi: 10.1093/bioinformatics/btaa614.

[90] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Evolution*, 147(1):195–197, 1981. doi: 10.1016/0022-2836(81)90087-5.

[91] R. R. Sokal and C. D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas science bulletin*, 38:1409–1438, 1958.

[92] P. S. Soltis and D. E. Soltis. Applying the bootstrap in phylogeny reconstruction. *Statistical Science*, 18(2):256 – 267, 2003. doi: 10.1214/ss/1063994980.

[93] C. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, 1904. doi: 10.2307/1422689.

[94] T. Sperlea. *Multiple sequence assignments: an introduction*, pages 3–15. Springer Berlin Heidelberg, 2022. doi: 10.1007/978-3-662-64473-7_1.

[95] S. Srinivasan, N. G. Hoffman, M. T. Morgan, F. A. Matsen, T. L. Fiedler, R. W. Hall, F. J. Ross, C. O. McCoy, R. Bumgarner, J. M. Marrazzo, et al. Bacterial communities in women with bacterial vaginosis: high resolution phylogenetic analyses reveal relationships of microbiota to clinical criteria. *PloS one*, 7(6):e37818, 2012. doi: 10.1371/journal.pone.0037818.

[96] A. Stamatakis. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics*, 30(9):1312–1313, 2014. doi: 10.1093/bioinformatics/btu033.

[97] A. Stamatakis, P. Hoover, and J. Rougemont. A rapid bootstrap algorithm for the RAxML web servers. *Systematic Biology*, 57(5):758–771, 2008. doi: 10.1080/10635150802429642.

[98] M. A. Steel and D. Penny. Distributions of tree comparison metrics—some new results. *Systematic Biology*, 42(2):126–141, 1993. doi: 10.1093/sysbio/42.2.126.

[99] E. Štrumbelj and I. Kononenko. Explaining prediction models and individual predictions with feature contributions. *Knowledge and Information Systems*, 41(3):647–665, 2014. doi: 10.1007/s10115-013-0679-x.

[100] S. Sunagawa, L. P. Coelho, S. Chaffron, J. R. Kultima, K. Labadie, G. Salazar, B. Djahanschiri, G. Zeller, D. R. Mende, A. Alberti, F. M. Cornejo-Castillo, P. I. Costea, C. Cruaud, F. d'Ovidio, S. Engelen, I. Ferrera, J. M. Gasol, L. Guidi, F. Hildebrand, F. Kokoszka, C. Lepoivre, G. Lima-Mendez, J. Poulain, B. T. Poulos, M. Royo-Llonch, H. Sarmento, S. Vieira-Silva, C. Dimier, M. Picheral, S. Searson, S. Kandels-Lewis, T. O. Coordinators, C. Bowler, C. de Vargas, G. Gorsky, N. Grimsley, P. Hingamp, D. Iudicone, O. Jaillon, F. Not, H. Ogata, S. Pesant, S. Speich, L. Stemmann, M. B. Sullivan, J. Weissenbach,

P. Wincker, E. Karsenti, J. Raes, S. G. Acinas, and P. Bork. Ocean plankton. structure and function of the global ocean microbiome. *Science*, 348(6237):1261359, 2015. doi: 10.1126/science.1261359.

[101] E. Susko. Bootstrap support is not first-order correct. *Systematic Biology*, 58(2):211–223, 2009. doi: 10.1093/sysbio/syp016.

[102] A. Togkousidis, O. M. Kozlov, J. Haag, D. Höhler, and A. Stamatakis. Adaptive RAxML-NG: accelerating phylogenetic inference under maximum likelihood using dataset difficulty. *Molecular Biology and Evolution*, 40(10):msad227, 2023. doi: 10.1093/molbev/msad227.

[103] J. Trost, J. Haag, D. Höhler, L. Nesterenko, L. Jacob, A. Stamatakis, and B. Boussau. Simulations of sequence evolution: how (un)realistic they really are and why. *bioRxiv*, 2023. doi: 10.1101/2023.07.11.548509.

[104] X. Wang, I. K. Jordan, and L. W. Mayer. A phylogenetic perspective on molecular epidemiology. *Molecular Medical Microbiology*, pages 517–536, 2015. doi: 10.1016/B978-0-12-397169-2.00029-9.

[105] D. E. Wood, J. R. White, A. Georgiadis, B. V. Emburgh, S. Parpart-Li, J. Mitchell, V. Anagnostou, N. Niknafs, R. Karchin, E. Papp, C. McCord, P. LoVerso, D. Riley, L. A. Diaz, S. Jones, M. Sausen, V. E. Velculescu, and S. V. Angiuoli. A machine learning approach for somatic mutation discovery. *Science Translational Medicine*, 10(457):eaar7939, 2018. doi: 10.1126/scitranslmed.aar7939.

[106] J. C. Wooley, A. Godzik, and I. Friedberg. A primer on metagenomics. *PLoS Comput Biol*, 6(2):e1000667, 2010. doi: 10.1371/journal.pcbi.1000667.

[107] Z. Yang. Maximum likelihood estimation on large phylogenies and analysis of adaptive evolution in human influenza virus a. *J Mol Evol*, 51(5):423–432, 2000. doi: 10.1007/s002390010105.

[108] Z. Yang. *Computational molecular evolution.* Oxford University Press, 2006. doi: 10.1093/acprof:oso/9780198567028.001.0001.

[109] W. Zaman, J. Ye, S. Saqib, Y. Liu, Z. Shan, D. Hao, Z. Chen, and P. Xiao. Predicting potential medicinal plants with phylogenetic topology: inspiration from the research of traditional chinese medicine. *Journal of Ethnopharmacology*, 281:114515, 2021. doi: 10.1016/j.jep.2021.114515.

A. Appendix

A.1. Use Case I: Phylogenetic Placement Entropy Prediction

A.1.1. Difficulty Distribution

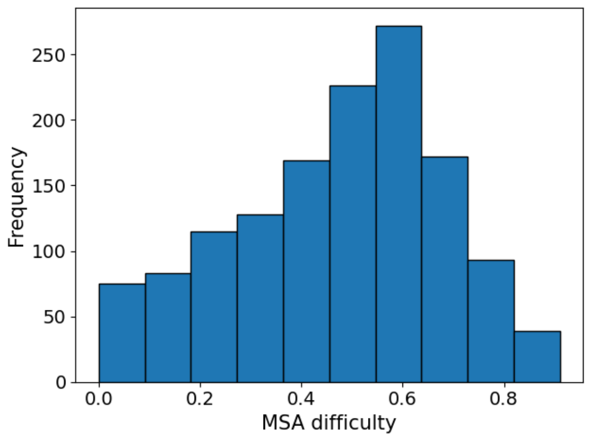


Figure A.1.: Difficulty distribution of the datasets for the placement difficulty prediction. Sample size: 1800 datasets.

A.1.2. Placement Entropy Scatterplots

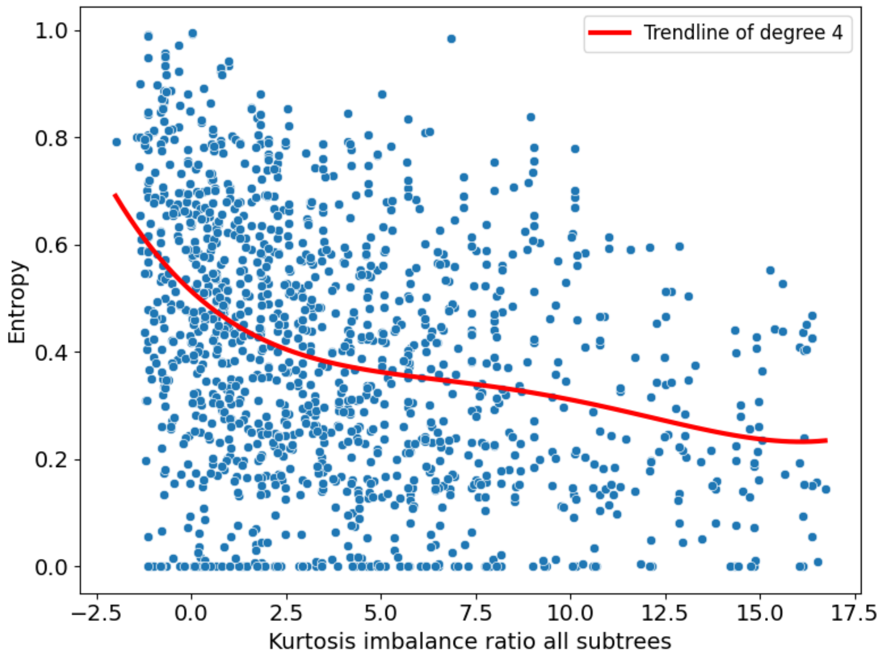


Figure A.2.: Placement entropy as a function of the kurtosis of the imbalance ratios of all subtrees. Sample size: 1500.

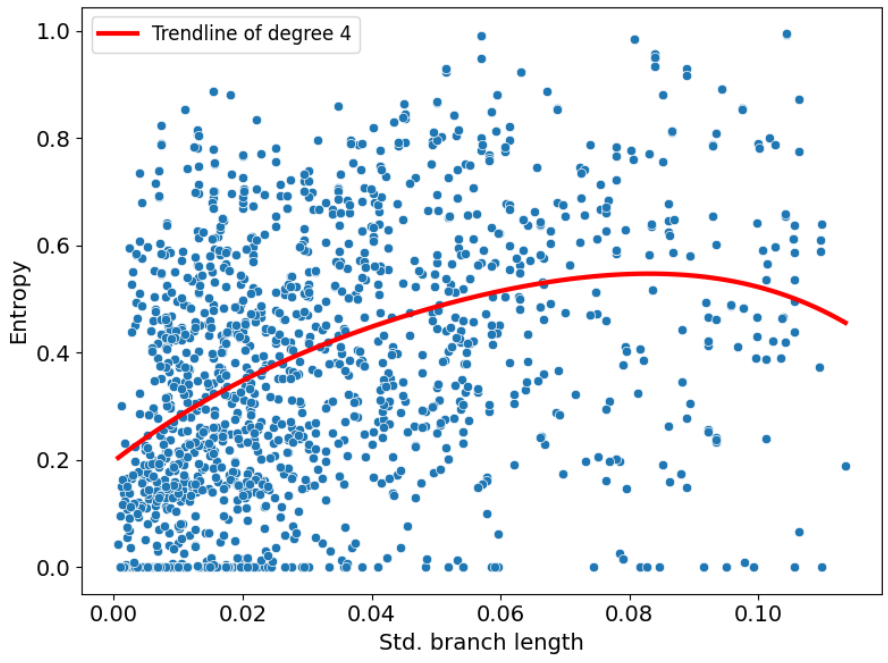


Figure A.3.: Placement entropy as a function of the standard deviation of the branch length. Sample size: 1500.

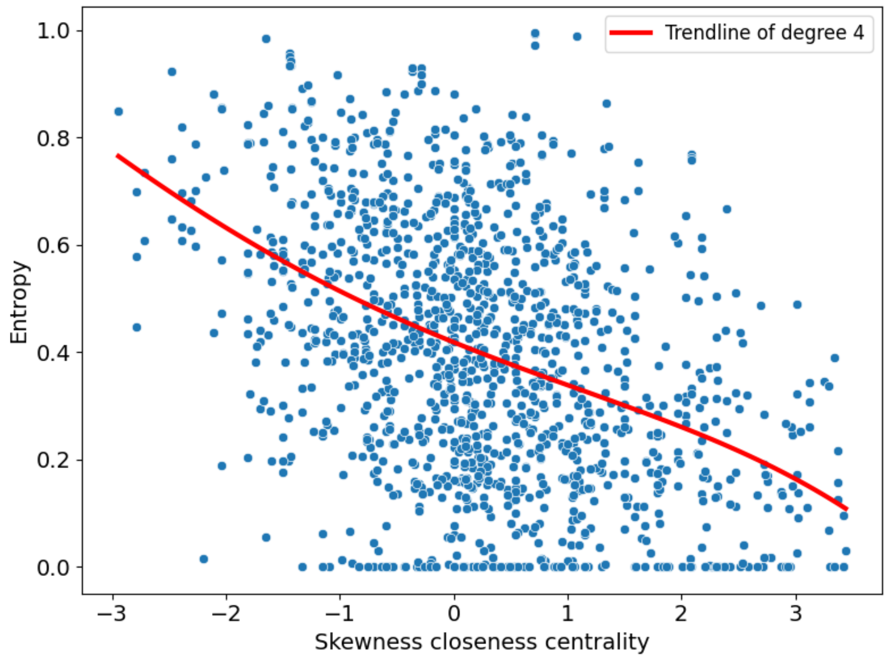


Figure A.4.: Placement entropy as a function of the skewness of the closeness similarity. Sample size: 1500.

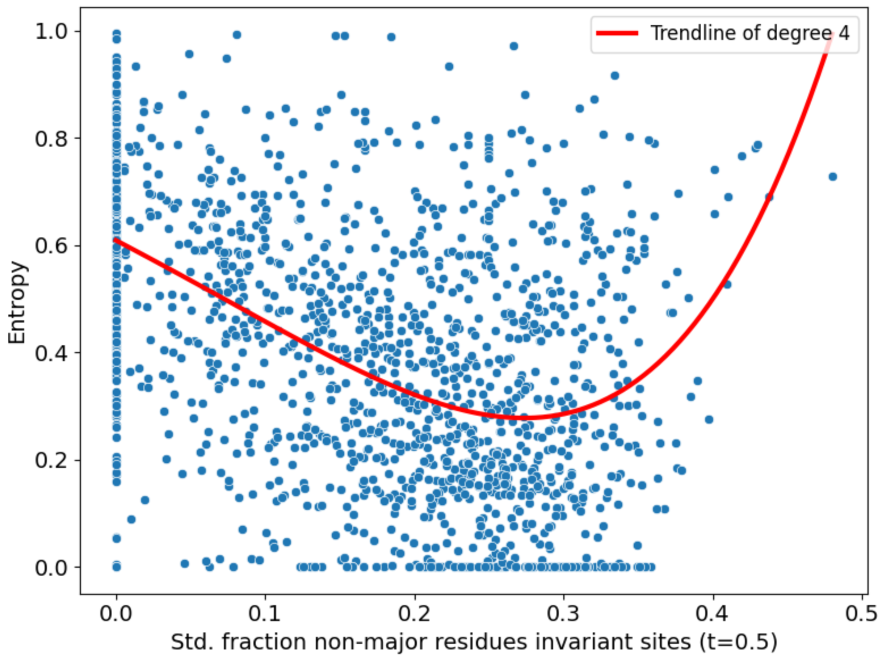


Figure A.5.: Placement entropy as a function of standard deviation of the fractions of non-major residues for invariant sites ($t := 0.5$). Sample size: 1500.

A.1.3. MSA Difficulty Correlations

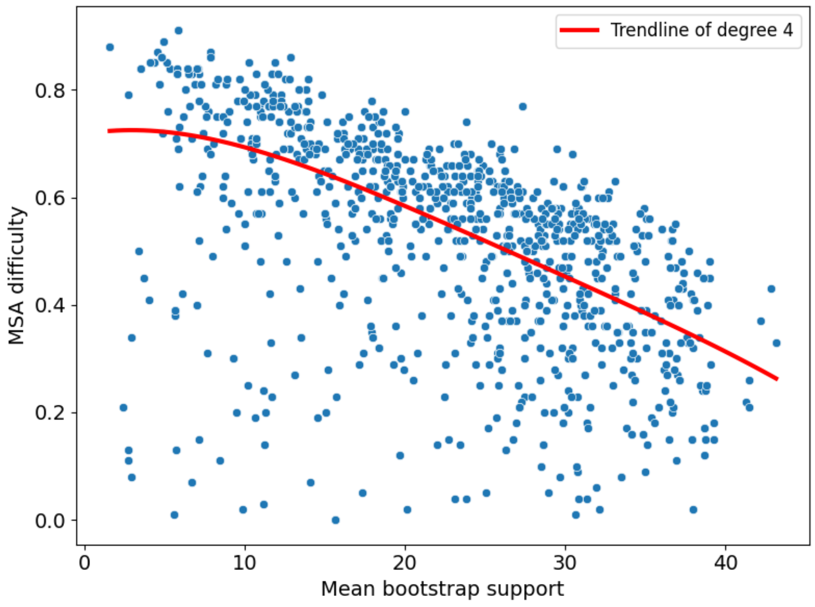


Figure A.6.: MSA difficulty as a function of mean bootstrap support. Sample size figure: 1500. Spearmans $\rho = -0.69$ with $p \ll 1 \times 10^{-10}$.

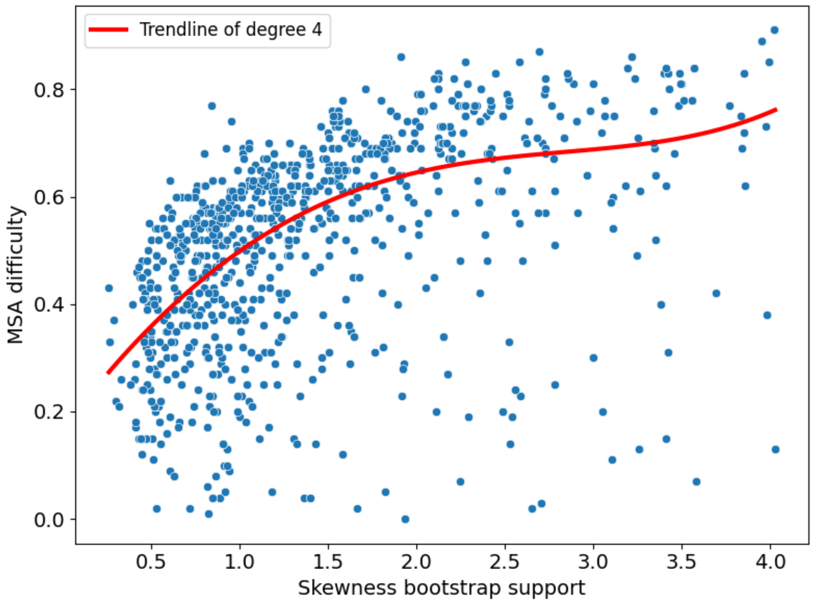


Figure A.7.: MSA difficulty as a function of skewness bootstrap support. Sample size figure: 1500. Spearmans $\rho = 0.68$ with $p \ll 1 \times 10^{-10}$.

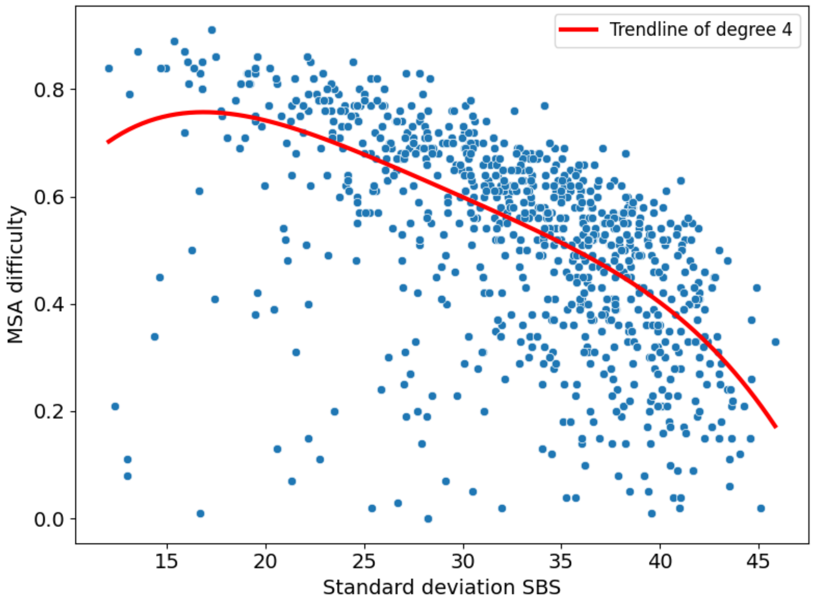


Figure A.8.: MSA difficulty as a function of mean bootstrap support. Sample size figure: 1500. Spearmans $\rho = -0.71$ with $p \ll 1 \times 10^{-10}$.

A.1.4. SHAP Plot Examples

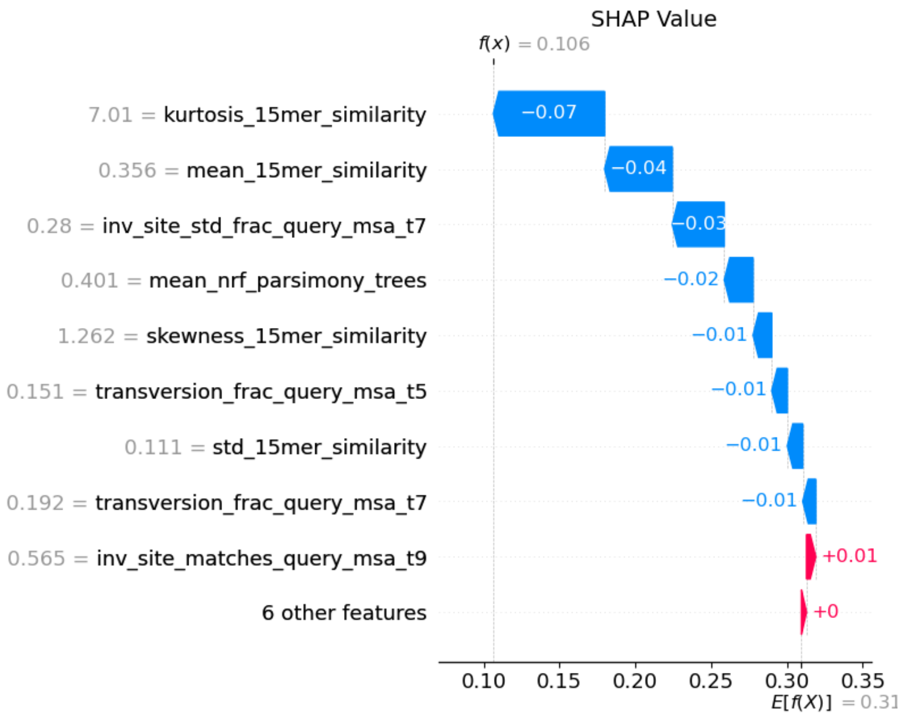


Figure A.9.: SHAP value waterfall plot for a placement with low placement entropy. The plot depicts how the feature values influence the predicted placement entropy. The base value is the expected prediction $E[f(x)] = 0.31$. Starting at the base value the plot lists the individual influences of the feature values which accumulate in the final prediction $f(x) = 0.106$.

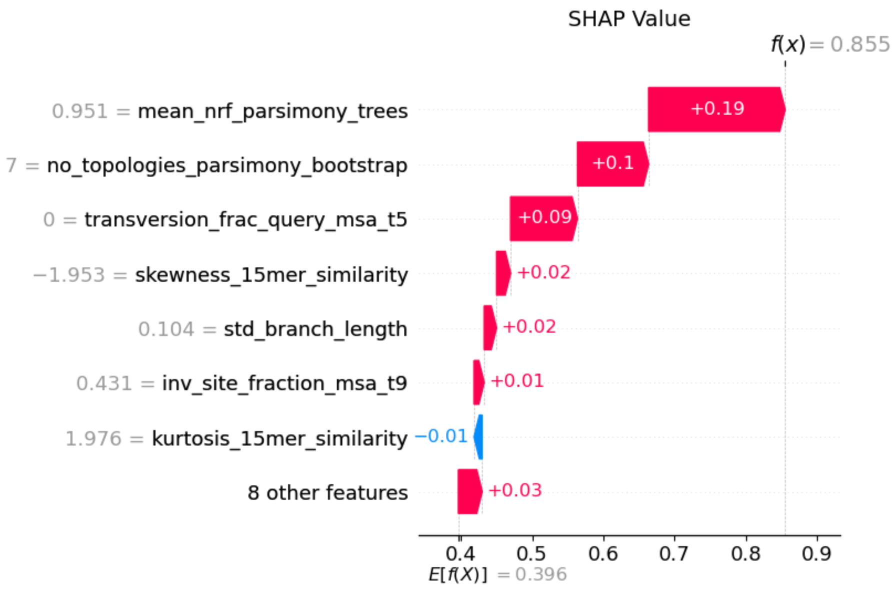


Figure A.10.: SHAP value waterfall plot for a placement with high placement entropy.

A.1.5. Prediction Error Distribution

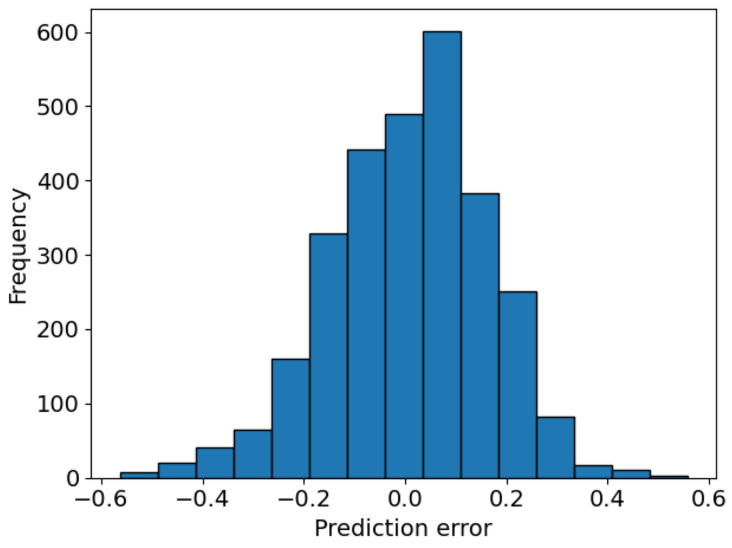


Figure A.11.: Prediction error distribution for the phylogenetic placement entropy prediction. 81% of the absolute prediction errors are ≤ 0.2.

A.2. Use Case II: Bootstrap Support Prediction

A.2.1. Command References

Tool	*Version*	*Commands*
RAxML	8.2.12	**Rapid Bootstrap**: raxmlHPC-PTHREADS -T 60 -m GTRGAMMA/PROTGAMMALG -s msa_filepath -# 1000 -p 12345 -x 12345 **Rapid Bootstrap (MPI)**: mpirun raxmlHPC-HYBRID-AVX -m GTRGAMMA/PROTGAMMALG -s msa_filepath -# autoMRE -p 12345 -x 12345
RAxML-NG	1.1.0	**Search**: raxml-ng −−adaptive −−msa msa_filepath −−model GTR+G/LG+G −−threads auto{60} **SBS**: raxml-ng −−bootstrap −−model model_filepath −−bs−trees 1000 −−msa msa_filepath **SBS (MPI)**: mpirun raxml-ng −−bootstrap −−model model_filepath −−bs−trees 1000 −−msa msa_filepath −−workers 50
IQ-TREE2	2.2.2.7	**UFBoot2**: iqtree2 -m GTR+G/LG+G -s msa_filepath -B 1000 -T AUTO −threads-max 60 **UFBoot2 (MPI)**: mpirun iqtree2-mpi -m GTR+G/LG+G -s msa_filepath -B 1000 -T AUTO −threads-max 60 **SH-like aLRT**: iqtree2 -m GTR+G/LG+G -s msa_filepath -alrt 1000 -T AUTO −threads-max 60

Table A.1.: Used commands for each tool

A.2.2. Features

- **Parsimony Support (PS)**
  We compute the PS using the −−start-option of RAxML-NG for generating 1000 parsimony starting trees. We calculate the support using RAxML-NG as well. More than 1000 parsimony starting trees do not yield a better predictor performance according to our experiments.

- **Parsimony Bootstrap Support (PBS)**
  We generate a PB by resampling the columns of the MSA with replacement. For each of the sampled MSAs, we compute the corresponding parsimony starting tree using the −−start-option in RAxML-NG. We infer a total of 200 PB trees. More than 200 PB trees do not yield a better predictor performance according to our experiments.

- **Normalized branch length**
  We normalize the branch lengths by the total sum of branch lengths of the tree.

- **# children inner branches**
  The number of inner branches in the subtree below the branch.

- **Skewness PBS**
  The skewness of the PBS of all inner branches of the input tree.

- **Mean Robinson-Foulds-distance PB**
  The mean Robinson-Foulds distance of the trees generated by the PB procedure.

Feature	*Importance (%)*
Parsimony Bootstrap Support (PBS)	82.2
Parsimony Support (PS)	3.1
Normalized branch length	2.0
# children inner branches	1.7
Skewness PBS	1.5
Mean Robinson-Foulds-distance PB	1.1
Mean Parsimony Mutations per Site (PMS)	1.1
Branch length	1.0
Max. PMS	1.0
Coefficient of variation PSF	0.7
Max. PBS children*	0.6
Mean PBS parents	0.6
Max. PS children*	0.5
Branch number (ordered by level-order traverse)	0.5
Skewness PMS	0.5
Branch length ratio bipartition	0.5
Max. PS children*	0.3
Min. PS children	0.3
Std. PBS parent branches	0.3
Std. PBS children branches	0.3
Mean closeness centrality bipartition ratio	0.3
Min. PS children*	0.2
Min. PBS children*	0.1

*: weighted by branch length

Table A.2.: Overview of the subset of features used for the prediction and their final feature importance for the EBG regressor. We obtained this subset via Recursive Feature Elimination.

- **Parsimony Mutations per Site (PMS) [statistic]**
  We compute the number of parsimony mutations per site using the tree and the MSA. Afterwards, we calculate the summary statistics.

- **[Min|Max|Mean|Std|Skewness] P(B)S [children|parents] [*]**
  We compute the statistics over the P(B)S for the inner branches below (children) or above (parents) the branch. As indicated by *, we also weigh the P(B)S values of the child/parent inner branches by their branch length for some features.

- **Mean closeness centrality bipartition ratio**
  We split the tree into two parts at the branch. We then transform the trees into graphs with networkx [44] and calculate their mean closeness centrality. We define the closeness centrality of node i as in Equation (A.1) [8] with N being the number of nodes in a graph

and $d(x, y)$ as the shortest branch distance between two nodes.

$$C(i) = \frac{1}{\sum_{i=1}^{N} d(x, i)} \tag{A.1}$$

By dividing the smaller by the larger closeness centrality we obtain the final feature.

A.2.3. Pearson Correlation EBG Regressor

MSA	*Pearson correlation*	*p-value*
10098_1	0.91	0.0
10115_3	0.88	0.0
10118_0	0.94	0.0
10169_0	0.95	0.0
10169_2	0.93	0.0
10196_0	0.94	0.0
10264_0	0.9	0.0
10268_3	0.96	0.0
10270_3	0.83	0.0
10271_5	0.92	0.0
10436_0	0.94	0.0
10436_8	0.84	0.0
10454_1	0.94	0.0
10542_1	0.93	0.0
10562_0	0.94	0.0
10568_0	0.95	0.0
10629_0	0.96	0.0
10652_0	0.92	0.0
10703_5	0.95	0.0
10714_0	0.95	0.0
10727_0	0.92	0.0
10749_0	0.97	0.0
10782_0	0.95	0.0
10791_9	0.87	0.0
10801_0	0.67	0.0
10856_1	0.87	0.0
10949_2	0.83	0.0
10983_0	0.78	0.0
10986_0	0.91	0.0
11032_4	0.9	0.0
11331_0	0.89	0.0
11376_0	0.94	0.0

11487_7	0.95	0.0
11712_1	0.92	0.0
11768_1	0.93	0.0
11777_0	0.86	0.0
11783_0	0.93	0.0
11966_0	0.83	0.0
11972_1	0.94	0.0
11988_0	0.88	0.0
11991_4	0.95	0.0
12013_0	0.9	0.0
12020_0	0.95	0.0
12165_2	0.92	0.0
12306_1	0.91	0.0
12306_13	0.97	0.0
12334_0	0.93	0.0
12339_0	0.93	0.0
12493_1	0.91	0.0
12493_7	0.86	0.0
12717_2	0.93	0.0
12746_0	0.91	0.0
12855_2	0.95	0.0
13184_0	0.89	0.0
13664_0	0.9	0.0
13801_2	0.92	0.0
13808_7	0.93	0.0
13815_2	0.97	0.0
13887_0	0.86	0.0
13909_0	0.93	0.0
13909_1	0.93	0.0
13985_6	0.92	0.0
14035_0	0.94	0.0
14151_0	0.98	0.002
14188_0	0.91	0.0
14232_0	0.83	0.0
14244_0	0.7	0.0
14504_0	0.94	0.0
14526_0	0.94	0.0
14534_25	0.95	0.0
14643_2	0.92	0.0
14663_0	0.85	0.0
14688_0	0.85	0.033
14688_13	0.92	0.0

14688_23	0.92	0.0
14725_5	0.95	0.0
14725_7	0.96	0.0
14833_4	0.94	0.0
14923_0	0.95	0.0
14954_0	0.9	0.0
14959_2	0.96	0.0
15019_5	0.89	0.0
15021_3	0.92	0.0
15021_7	0.94	0.0
15039_2	0.87	0.0
15097_1	0.93	0.0
15179_2	0.95	0.0
15253_0	0.82	0.0
15306_5	0.93	0.0
15368_0	0.91	0.0
15427_0	0.91	0.0
15635_1	0.89	0.0
15636_0	0.82	0.0
15639_0	0.87	0.0
15669_9	0.92	0.0
15682_0	0.92	0.0
15769_0	0.94	0.0
15828_0	0.95	0.0
15861_2	0.93	0.0
15908_1	0.86	0.0
16007_0	0.87	0.0
16009_1	0.89	0.0
16105_0	0.88	0.0
16141_1	0.78	0.041
16190_2	0.94	0.0
16269_0	0.92	0.0
16313_11	0.94	0.0
16453_0	1.0	1.0
16629_0	0.88	0.0
16632_2	0.96	0.0
16637_2	0.97	0.0
16675_0	0.88	0.0
16737_0	0.84	0.0
16748_0	0.84	0.0
16785_1	0.9	0.0
16855_2	0.97	0.0

17014_0	0.9	0.0
17168_0	0.95	0.0
17390_1	0.92	0.0
17443_0	0.92	0.0
17594_11	0.93	0.0
17594_13	0.95	0.0
17666_0	0.89	0.0
17723_0	0.94	0.0
17749_1	0.9	0.0
17761_0	0.91	0.0
17774_4	0.97	0.0
17791_0	0.85	0.0
17814_0	0.93	0.0
17878_9	0.85	0.0
17885_1	0.96	0.0
17896_31	0.92	0.0
18077_0	0.94	0.0
18131_0	0.84	0.0
18218_1	0.96	0.0
18258_2	0.92	0.0
18438_0	0.75	0.003
18448_0	0.89	0.0
18465_0	0.94	0.0
18638_0	0.87	0.0
18638_1	0.92	0.0
18654_0	0.95	0.0
18850_2	0.62	0.574
18883_0	0.92	0.0
19060_0	0.96	0.0
19447_0	0.91	0.0
19466_3	0.92	0.0
19509_1	0.99	0.0
19579_0	0.91	0.0
19740_5	0.82	0.0
19782_3	0.78	0.0
19797_0	0.95	0.0
19889_1	0.89	0.0
19925_0	0.96	0.0
19925_6	0.95	0.0
20079_4	0.94	0.0
20196_18	0.96	0.0
20196_19	0.96	0.0

20239_0	0.93	0.0
20239_3	0.95	0.0
20250_1	0.87	0.0
20736_0	0.94	0.0
20944_1	0.79	0.0
21191_0	0.83	0.0
21303_0	0.92	0.0
2180_2	0.95	0.0
21817_6	0.87	0.0
2191_2	0.94	0.0
21973_9	0.98	0.0
22052_0	0.93	0.0
22091_1	0.78	0.0
2217_0	0.93	0.0
22200_0	0.91	0.0
2224_0	0.92	0.0
22408_11	0.97	0.0
22429_0	0.9	0.0
22442_11	0.92	0.0
22442_6	0.85	0.002
22475_0	0.9	0.0
2248_0	0.96	0.0
2250_0	0.91	0.0
22552_1	0.91	0.0
2256_1	0.98	0.0
22751_1	0.95	0.0
22798_0	0.85	0.0
22805_0	0.93	0.0
22941_0	0.92	0.0
22999_3	1.0	0.017
23036_0	0.94	0.0
23279_0	0.96	0.0
23282_0	0.95	0.0
23436_0	0.82	0.0
23535_1	0.84	0.0
23593_0	0.95	0.0
23768_0	0.89	0.0
23884_0	0.93	0.0
25031_0	0.8	0.0
25084_2	0.9	0.0
25181_0	0.94	0.0
25256_20	0.93	0.002

25256_23	1.0	0.0
25284_1	0.9	0.0
25341_1	0.92	0.0
25554_1	0.92	0.0
25635_1	0.83	0.0
25818_4	0.94	0.0
25829_8	0.87	0.0
26085_4	0.95	0.0
26188_0	0.85	0.0
26212_1	0.91	0.0
26551_4	0.95	0.0
26628_4	0.97	0.0
26669_46	0.96	0.0
26988_12	0.79	0.0
26988_8	0.97	0.0
27016_1	0.94	0.0
27176_0	0.93	0.0
27689_0	0.91	0.0
28112_0	0.85	0.0
28258_0	0.98	0.0
28360_17	0.98	0.0
28360_18	0.96	0.0
28360_2	0.95	0.0
28360_30	0.95	0.0
28360_8	0.96	0.0
362_1	0.89	0.0
684_1	0.96	0.0
688_1	0.95	0.0
9936_1	0.97	0.0
9972_0	0.96	0.0
9987_1	0.89	0.0

Table A.3.: EBG regressor correlation with the SBS ground truth of 1000 replicates. In three cases the Pearson correlation is < 70 (10801_0, 18850_2). For 10801_0 and 18850_2 the mean SBS support is extremely low (5 and 23) suggesting very uncertain phylogenies.

A.2.4. Tool Balanced Accuracy and MSA Difficulty

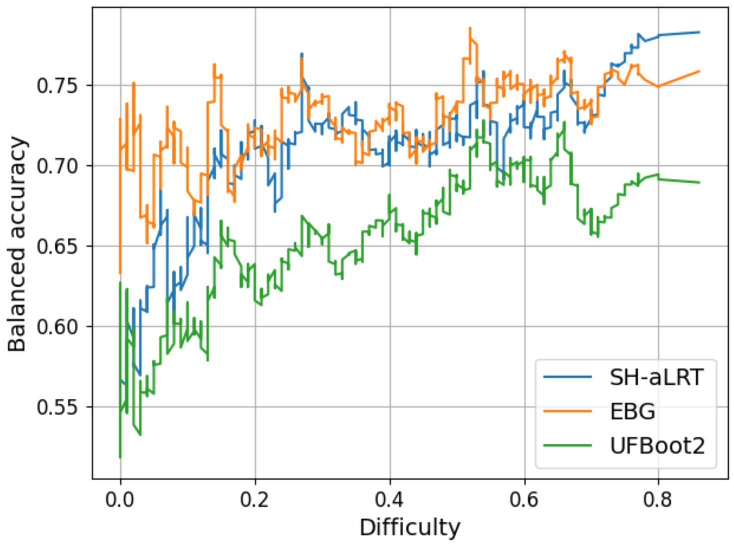


Figure A.12.: Balanced accuracy comparison of EBG, UFBoot2, and SH-like aLRT concerning MSA difficulty.

We compared the BAC of the three tools in predicting whether a branch is in the true tree or not. We set the decision threshold for EBG and SH-like aLRT to 80, as they behave similarly on the simulated MSAs. For UFBoot2 we set the threshold for predicting the branch is in the true tree to 95. According to the authors, this yields a 5% false positive bound [70]. It is difficult to compare the BAC values of the three tools due to the definition of individual thresholds. At least with this set of thresholds, EBG yields the most consistent performance on the simulated MSAs.

A.2.5. Different Models

Tool	*MAE*	*BAC*
Logistic/Ridge Regression	10 ± 0.2	0.89 ± 0.00
Random Forest	12 ± 1.6	0.89 ± 0.00
LightGBM	8.3 ± 0.2	0.91 ± 0.00

Table A.4.: Performance of different machine learning models on the regression and classification task formulation. The table shows the mean and standard deviation of the metrics based on 10 repeated random holdouts of size 20%.

A.2.6. Number of Parsimony Trees

Configuration	*MdAE*	*RMSE*
100 parsimony bootstrap starting trees	7.9	13.8
200 parsimony bootstrap starting trees	7.6	13.1
500 parsimony bootstrap starting trees	7.5	13.0
100 parsimony starting trees	9.3	14.9
1000 parsimony starting trees	8.5	14.2
10000 parsimony starting trees	8.4	14.0

Table A.5.: Performance comparison of the EBG regressor with different numbers of parsimony (bootstrap) trees as the basis for the P(B)S features. The table shows the mean of the metrics based on 10 repeated random holdouts of size 20%. As 500 PB trees did not yield a significantly better MdAE compared to 200 PB trees, we chose to use 200 PB trees for a better runtime. The same rationale led to the choice of 1000 parsimony starting trees.

A.2.7. CPU times EBG and UFBoot2

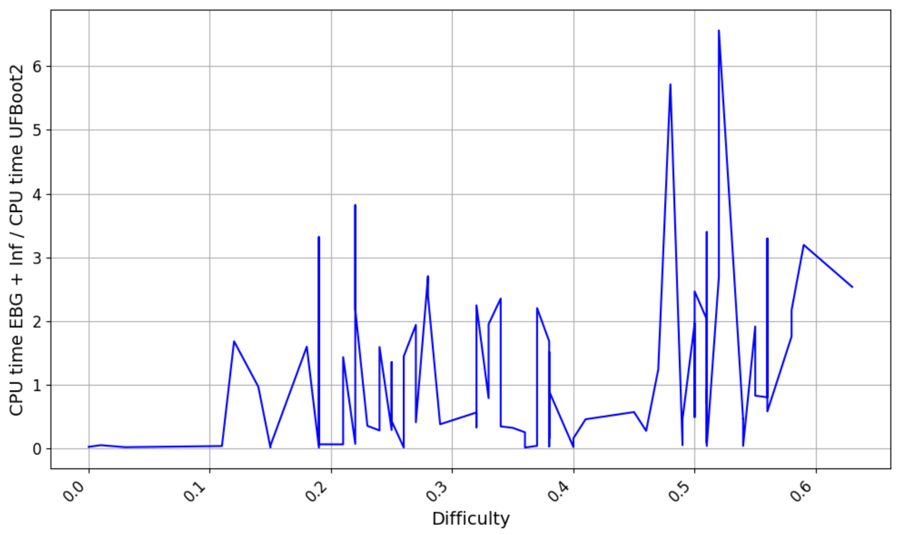


Figure A.13.: Ratio of the EBG and inference CPU time against the UFBoot2 CPU time on 84 MSAs.