Deggendorf Institute of Technology

Master Thesis:

Detecting Genomic Anomalies:

A Machine Learning/ Deep Learning Approach

Vishnu Vijayan

Matr. - Nr. 00806007

29 April 2024

Supervised by:

Prof. Dr. Ingo Ebersberger

Gökçe Aydos

Abstract

With the rapid advancements in genome sequencing technology, there has been a significant increase in the number of genome assemblies stored in public databases. At the same time, the incidence of *genome contamination* which refers to the unintentional inclusion of foreign sequences into genuine ones, has also escalated. These undesired anomalies in genomic assemblies can distort downstream analysis results. Differentiating between contamination and horizontal gene transfer (HGT) poses a significant challenge, as both stem from foreign species and can lead to variations in the considered sequence, potentially resulting in misattributed functions or false HGT claims. Existing tools often focus on single characteristics of genes to classify it as contamination or HGT, which may not be sufficient for reliable distinction.

The aim of this project was to create a machine learning classification model capable of predicting Target Contamination genes (CT), Reference Contamination genes (CR), HGT genes, and Target genes (TG), and to effectively distinguish among them. Two labeling approaches were tested and evaluated to determine the most effective method for creating training data for a machine learning model: the 5% quantile labeling technique and the upstream/downstream labeling technique, both using a feature vector table that describe each gene in the genomic assembly of the target species. The 5% quantile labeling method entails examining different features of genes within a genomic assembly and classifying these genes according to how their characteristics and variations differ from the overall assembly's norm. On the other hand, the upstream/downstream labeling method focuses on the genes immediately adjacent to a specific gene, classifying them according to their observed patterns and assigned taxon.

When tested on the genomic assembly of *Lasallia pustulata*, a species of lichen, the 5% quantile technique identified 40.41% more TG genes, 22 times more CT genes, 1.64 times more CR genes, and 3.52 times fewer HGT genes compared to the Up/downstream technique. Despite its limitations, the Up/downstream technique demonstrated better performance, especially considering its applicability to fragmented assemblies and the consistency of gene labels within various clusters observed within an assembly. Based on this, a supervised Random Forest machine model to predict different classes of genes was developed and it exhibited accuracy of 0.81 and precision of 0.70. When tested on *Lasallia pustulata* assembly, the machine learning model successfully identified 70.86% of the labeled TG as such but failed to recognize the remaining genes. Additionally, none of the CT and CR genes, which were marked using the labeling method, were predicted by the model. The model was only able to correctly predict 1.58% of the genes labeled as HGT by the labeling method. In general, the model was biased toward the majority class, which is the TG even after up sampling the underrepresented classes, which are HGT, CR and CT.

Contents

Abbreviations

BLAST	Basic Local Alignment Search Tool
bp	base pairs
CR	Contamination in reference
CT	Contamination in target
DL	Deep Learning
DNA	Deoxyribonucleic acid
EDA	Exploratory Data Analysis
HGT	Horizontal gene transfer
MILTS	Minimum Information Standard for Laterally Transferred Gene Sequences
ML	Machine Learning
NCBI	National Centre for Biotechnology Information
NCBI-nr	NCBI non-redundant protein [database]
NGS	Next Generation Sequencing
nr	non-redundant
PC	Principal Component
PCA	Principal component analysis
RF	Random Forest
RefSeq	Reference Sequence
RNA	Ribonucleic acid
SMOTE	Synthetic Minority Oversampling Technique
SVM	Support Vector Machine
TG	Target Gene

1. INTRODUCTION

1.1 Genome assembly and Taxonomic assignment

The genome compiles all the information in the DNA of an organism [1]. A segment of DNA or gene is replicated into numerous single-stranded ribonucleic acid (RNA) molecules to serve as a blueprint for synthesizing proteins. These proteins are vital for the proper functioning of all cells within the organism [2]. Multiple sequencing techniques can be utilized to acquire the genome sequence of a species, as explored in various studies [3][4][5][6]. However, a key challenge arises in fragmenting the genome into smaller segments, impacting the sequencing process [7][8]. Reconstructing the complete genome from these fragments, which vary in length from around 150 base pairs to several tens of thousands of base pairs, necessitates their orderly assembly. A common strategy involves identifying overlaps between the reads and assembling them accordingly [8]. The resultant sequence of the genome is known as the *assembly*.

Taxonomic assignment is the process of identifying the taxonomic classification of a biological sequence by comparing it to a database of known classifications. Phylogenetic taxonomy illustrates the evolutionary relationships among organisms through names and graphical representations [9]. Taxonomies convey details about evolutionary history using names and graphical structures like branching diagrams. Clear and consistent meanings of taxonomy names are essential for their utility [10][11]. This assignment helps analyze the diversity and abundance of organisms [12]. It is also crucial for inferring functions, as closely related organisms often share similar metabolic pathways and traits [13]. Accurate taxonomic assignment depends on high-quality reference databases covering a wide range of taxonomic diversity.

We currently lack a comprehensive database that fully maps DNA sequences to their taxonomic origins, as many microorganisms remain undiscovered. However, various methods have been developed to classify newly discovered genomic sequences by comparing them to sequences from known species [14]. When examining a specific region of interest within the

genome, comparing the similarity and dissimilarity between known and unknown sequences can help identify the organism of origin. Two commonly used algorithms for aligning and scoring sequences are the Needleman-Wunsch algorithm [15] and the Smith-Waterman algorithm [16]. The Needleman-Wunsch algorithm considers the entire length of sequences and optimizes alignment between two sequences, known as global alignment, while the Smith-Waterman algorithm focuses on aligning segments of sequences with high similarity, known as local alignment.

1.2 Contamination in Genomic assemblies

The quality of genome assemblies is critical for accurate analysis, with contamination being a key factor that can affect results. Contamination occurs when a genome contains foreign sequences alongside its genuine sequence, and it can be categorized as biological, experimental, or computational [17]. These categories contribute to both redundant and non-redundant contaminations. Redundant contamination occurs when a genomic segment is duplicated within the genome, while non-redundant contamination involves the replacement of a genomic region from the main organism with that of a foreign organism. Additionally, any additional DNA segment not belonging to the main organism contributes to non-redundant contamination. Foreign sequence introduction can happen at various stages of the sequencing process, spanning from organism culture to data processing [17]. Common causes of genomic contamination include the presence of unwanted organisms and the inadvertent inclusion of extraneous DNA during DNA extraction or sequencing on shared platforms [18]

1.3 Overview of Algorithms for detection of contamination

Detecting contaminants within genome assemblies is critical due to their potential to compromise data quality and introduce artifacts. Figure 1 provides an overview of various algorithms used for this purpose, categorized into database-free and reference database-dependent approaches. In 2013, Blobology was introduced as a tool for visualizing sequences from low complexity metagenomic assemblies, employing Guanosine+Cytosine (GC) content and read coverage to plot contigs on a two-dimensional graph, with sequence taxonomy

retrieval supported by BLAST and DIAMOND blast [19]. Anvi'o, another established visualization tool, clusters contigs based on a combination of k-mer frequencies and read coverage, facilitating contamination detection, albeit requiring separate protein annotation [20]. ProDeGe, developed in 2015, calibrates k-mer frequency profiles based on the expected organism's taxonomy provided by the user [21]. Taxonomy is assigned to contigs through a homology search against a curated version of the Integrated Microbial Genome (IMG) database, with contigs considered uncontaminated if their taxonomy matches the user-provided label. In contrast, PhylOligo constructs profiles without relying on taxonomy, offering a Neighbor-Joining tree for user selection of calibration contigs [22]. However, these tools require manual inspection by users on a case-by-case basis, limiting their scalability for large-scale projects.

Another straightforward approach, which relies on commonly found gene markers, is SINA, where the presence of SSU rRNA genes in a genome assembly is evaluated [23]. This method has been employed to assess the level of contaminants in cyanobacterial genomes. In 2020, a method called Conterminator [18] was introduced for detecting contamination in genome assemblies, facilitating the processing of extensive nucleotide and protein sequence datasets on a single server. This method was utilized to quantify the extent of contamination in the nucleotide databases GenBank [24] and the comprehensive NR protein database. Conterminator identified 114,035 and 2,161,746 contaminated sequences affecting 2,767 and 6,795 species in RefSeq and GenBank, respectively.

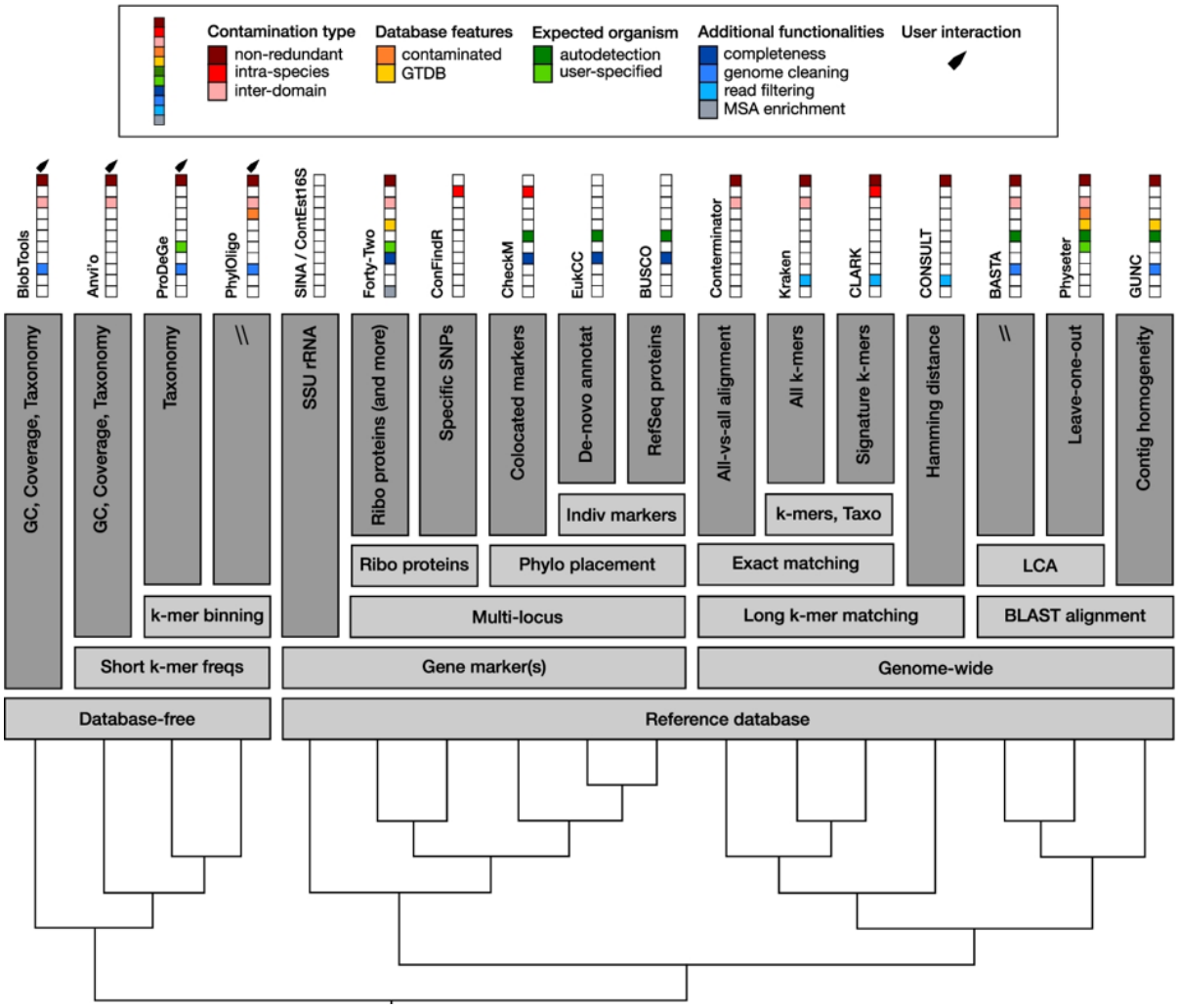

***Figure 1**: Overview of algorithms used to detect contamination in genomic assemblies. The algorithms are classified based on their operating principles [17].*

1.4 Horizontal gene transfer (HGT)

Horizontal gene transfer (HGT), also known as lateral gene transfer, is the process by which genetic material moves across normal mating barriers between organisms that are distantly related, contrasting with the typical vertical transmission of genes from parent to offspring [25]. In this process, genetic material is not transferred vertically from parent to child but is instead taken up by the organism and integrated into its genome. This phenomenon was first observed by Griffith in 1928 through an experiment demonstrating the transfer of virulence factors from a virulent to a non-virulent strain of Streptococcus pneumoniae [26]. HGT is a well-known phenomenon in bacteria and serves as a crucial facilitator for their rapid adaptation to changing environments [27]. Transduction, conjugation, and natural

transformation are the primary mechanisms of horizontal gene transfer (HGT) in bacteria (Figure 2). Transduction involves bacteriophages accidentally transferring genetic material from one infected bacterium to another. Conjugation requires the proximity of two bacterial cells, with a mating pilus facilitating the transfer of genetic material from a donor to a recipient bacterium. Natural transformation occurs when a bacterium enters a state of natural competence, allowing it to uptake free DNA from its surroundings, often originating from the lysis of other bacteria. If the incoming DNA integrates into the bacterium's chromosome through recombination, the bacterium is considered naturally transformed [28].

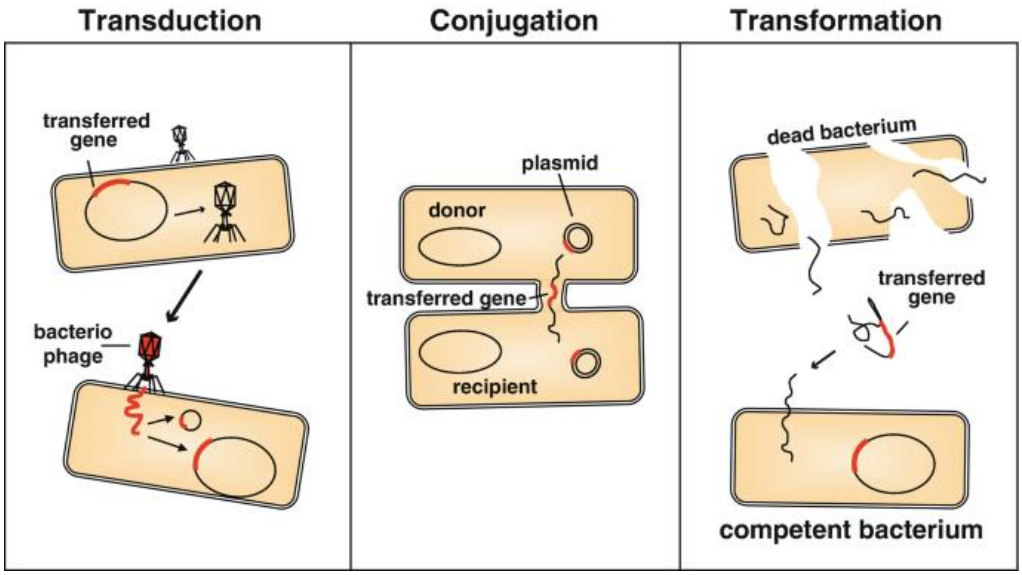

***Figure 2**: Overview of Horizontal Gene Transfer Mechanisms in Bacteria.[28] This diagram illustrates the 3 main methods by genetic material is acquired by bacteria from external sources: transduction, where a bacteriophage carries a gene from one bacterium to another; conjugation, the direct transfer of DNA from a donor to a recipient bacterium through a connecting pilus; and transformation, where a competent bacterium takes up a gene from the environment, often from lysed, dead bacteria*

1.5 Objectives

Advancements in sequencing technology have significantly reduced the cost of genome sequencing and assembly, enabling the analysis of many organisms. However, concerns persist regarding the quality of these assemblies due to contamination present in public databases [18]. Detecting contamination and HGT within genomic assemblies is crucial, prompting the need for highly sensitive and reliable identification tools.

In this project, a supervised machine learning model is introduced, building upon the capabilities of taXaminer which is a tool aimed at predicting target genes (TG), contaminations, and HGT genes within a genomic assembly. Various labeling techniques are explored for the gene dataset to train the supervised model. The optimal labeling technique is determined by comparing the principal component analysis (PCA) plot of the labeled dataset with that of taXaminer's output on *Lasallia pustulata*, a lichen-forming fungus.

Through an extensive analysis of different gene characteristics within the assembly, two subtypes of contamination are identified: Contamination in Target species (CT) and Contamination in Reference species (CR). Distinguishing between these clusters is a key objective. taXaminer's output file (*gene_tabele_taxon_assignment.csv*) (Table 1) provides numerous gene features, including taxon assignments, which aid in this differentiation.

	g_name	c_name	c_num_of_genes	c_len	c_pct_assemby_len	c_genelenm	c_genelensd	c_cov_7	c_cov_8	c_covsd_7
0	FUN_08148	scaffold_15	276	838651	0.0255	1444.2355	1006.609	42.9521	198.6266	31.8069
1	FUN_08149	scaffold_15	276	838651	0.0255	1444.2355	1006.609	42.9521	198.6266	31.8069
2	FUN_08150	scaffold_15	276	838651	0.0255	1444.2355	1006.609	42.9521	198.6266	31.8069
3	FUN_08151	scaffold_15	276	838651	0.0255	1444.2355	1006.609	42.9521	198.6266	31.8069
4	FUN_08152	scaffold_15	276	838651	0.0255	1444.2355	1006.609	42.9521	198.6266	31.8069
...	...	...	...	...	...	...	...	...	...	...
9820	FUN_01121	scaffold_1	1119	3307933	0.1005	1656.5648	1208.835	40.2567	191.8628	13.3554
9821	FUN_01122	scaffold_1	1119	3307933	0.1005	1656.5648	1208.835	40.2567	191.8628	13.3554
9822	FUN_01123	scaffold_1	1119	3307933	0.1005	1656.5648	1208.835	40.2567	191.8628	13.3554
9823	FUN_01124	scaffold_1	1119	3307933	0.1005	1656.5648	1208.835	40.2567	191.8628	13.3554
9824	FUN_01125	scaffold_1	1119	3307933	0.1005	1656.5648	1208.835	40.2567	191.8628	13.3554

9825 rows × 64 columns

***Table 1**: An overview of the feature vector table of Lasallia pustulata assembly. g_name is the name of the gene and the corresponding values describe the attributes of that gene including its length(c_num_of_genes), scaffold on which it is residing on(c_name), coverage(c_cov), taxon assignment (is_target) etc.*

The *is_target* column which contains a binary value denotes whether a gene is assigned to the target taxon (1) or not (0), allowing for further investigation to differentiate between CT, CR, and HGT. CT genes denote contamination within the target assembly, likely exhibiting distinct characteristics compared to TG. Conversely, CR genes represent contaminations within the

reference species, indicating genes not fully integrated into the reference species genomic assembly. It's plausible that CT genes in the reference species correspond to CR genes in the target species. Similarly, the target genes well integrated into the reference species may serve as potential candidates for HGT in the target assembly.

taXaminer provides vectors that describe a gene. These vectors throw light on the gene characteristics and its taxonomic assignment. Certain features like *c_len, c_num_of_genes, c_cov* etc. are very crucial in determining the character of a gene. In this project the relevance of these feature vectors is also studied.

2. MATERIALS and METHODS

The following chapter demonstrates the methodologies and resources utilized for data collection in this project, including the programming language, libraries, and ML/DL learning techniques employed.

2.1 taXaminer

taXaminer is a tool designed to differentiate between contamination and horizontal gene transfer (HGT) within genes in annotated assemblies. It incorporates a variety of alignment-free metrics, such as coverage data and sequence composition, alongside taxonomic assignment via similarity to sequences in the NCBI-nr database. The objective is to precisely ascertain the true source of genes by analyzing multiple parameters. As part of its functionality, taXaminer generates an interactive 3D plot that condenses 16 alignment-free criteria (Table 2) using principal component analysis. Each point on the plot corresponds to a gene of interest, with color denoting its taxonomic assignment. It evaluates each gene's reliability within a genome based on various characteristics, including sequence composition, coverage-related features, and the gene's integration into the assembly. By considering these feature vectors, a three-dimensional PCA can identify genes that deviate from the average assembly.

		column	descriptor name	explanation
		1	g_name	gene name (as given in the GFF file, 9th column, ID attribute)
		2	c_name	name of the contig the given gene is located on
contig	positional	**3**	**c_num_of_genes**	**number of genes annotated on the given contig**
		4	**c_len**	**contig length**
		5	c_pct_assembly_len	percentage of assembly length: proportion of total assembly length contained in the given contig
		6	**c_genelenm**	**mean gene length on the given contig**
		7	**c_genelensd**	**standard deviation of the gene lengths in the set of genes annotated on the given contig**
	coverage	**8**	**c_cov**	**mean read coverage of the given contig**
		9	**c_covsd**	**standard deviation of read coverage along the given contig**
		10	c_covdev	extent to which c_cov deviates from the mean assembly coverage, in units of total assembly coverage SD
		11	c_genecovm	mean read coverage in the set of genes annotated on the given contig
		12	c_genecovsd	standard deviation of the read coverage in the set of genes annotated on the given contig
	composition	**13**	**c_pearson_r**	**Pearson's correlation coefficient for the tetranucleotide-derived z-score vectors of the given contig and the overall assembly**
		14	c_pearson_p	probability (p-value) for two random sequences to show a correlation that is at least as high as c_pearson_r
		15	c_gc_cont	percentage of GC content in the given contig
		16	c_gcdev	extent to which c_gc_cont deviates from the mean assembly GC content, in units of total assembly GC SD
gene	positional	**17**	**g_len**	**length of the given gene**
		18	**g_lendev_c**	**extent to which g_len deviates from the corresponding c_genelenm, in units of c_genelensd**
		19	g_lendev_o	extent to which g_len deviates from the average gene length in the assembly, in units of the length SD over the complete set of genes in this assembly
		20	**g_abspos**	**absolute position of the gene (the closer to contig centre, the closer g_abpspos to 0; the closer to either contig end, the closer g_abpspos to 1)**
		21	**g_terminal**	**classifier of terminal genes (1 if gene is terminal, 0 if not)**
		22	g_single	classifier of single genes (1 if the given gene is the only gene annotated on its contig, 0 if not)
	coverage	**23**	**g_cov**	**mean read coverage of the given gene**
		24	**g_covsd**	**standard deviation of the read coverage along the given gene**
		25	**g_covdev_c**	**extent to which g_cov deviates from the corresponding c_genecovm, in units of c_genecovsd**
		26	g_covdev_o	extent to which g_cov deviates from the average gene coverage in the assembly, in units of the coverage SD over the complete set of genes in this assembly
	composition	**27**	**g_pearson_r_o**	**Pearson's correlation coefficient for the tetranucleotide-derived z-score vectors of the given gene and the overall assembly**
		28	g_pearson_p_o	probability (p-value) of a random data set to show a correlation with the overall assembly composition that is at least as high as g_pearson_r_o
		29	g_pearson_r_c	Pearson's correlation coefficient for the tetranucleotide-derived z-score vectors of the given gene and its respective contig
		30	g_pearson_p_c	probability (p-value) for two random sequences to show a correlation that is at least as high as g_pearson_r_c
		31	g_gc_cont	percentage of GC content in the given gene
		32	g_gcdev_c	extent to which g_gc_cont deviates from the mean gene GC content, in units of GC SD over the set of genes on the corresponding contig
		33	g_gcdev_o	extent to which g_gc_cont deviates from the mean GC content in overall gene set, in units of GC SD over the complete set of genes in this assembly

***Table 2**: Feature vector table (gene_table_taxon_assignment.csv) by taXaminer. Each vector describes the gene characteristic.*

2.2 Python

Python is an open-source and cross-platform programming language, that has become increasingly popular over the last ten years. It was first released in 1991. The latest version is 3.12.3. Currently, Python is one of the most widely used programming languages, especially in scientific computing.

The scripts for both labeling techniques as well as the machine model are developed using Python in this project. The list of packages used in the script and a short description of the usage is given in Table 3.

Package	Usage
Pandas	Data manipulation and cleaning.
NumPy	Numerical python provides support for large, multi-dimensional arrays and matrices.
Seaborn and Matplotlib	Used for data visualization.
Scikit-learn	Machine learning library that provides tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction.
Joblib	For saving and loading Python objects that make use of NumPy data structures.
tensorflow	TensorFlow library is used for a variety of applications, including DL,Ml , data analysis etc.
plotly	Graphing library to build complex interactive plots.

Table 3: *List of main python packages employed in the project.*

2.3 Visual studio code

Visual Studio Code (VS Code) is an open-source code editor for development and debugging cloud and web applications which is available on Linux and Windows. VS Code supports more a variety of programming, markup, and database languages, some of which are JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, Less, Sass, JSON, XML and Python.

2.4 Spyder

Spyder is a free, open-source scientific platform developed in Python It integrates the editing, analysis, debugging, and profiling features of a complete development tool with the data exploration, execution, deep inspection.

2.5 Jupyter notebook

Notebooks works on the console-based approach to interactive computing providing the user with a web application suitable for capturing the whole coding process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook is a combination of two components: A web application and Computational notebook documents.

2.6 Principal components analysis (PCA)

PCA is one of the techniques for taking high-dimensional data and using the dependencies between the variables to represent it in a lower-dimensional form, without losing too much information. PCA is one of the simplest and most robust ways of doing such dimensionality reduction. It is also one of the oldest, and has been rediscovered many times in many fields, so it is also known as the Karhunen-Loève transformation, the Hotelling transformation, the method of empirical orthogonal functions, and singular value decomposition.

In this project, PCA plays a crucial role as it gives the idea about the characteristics of a genome assembly at a glance. Both the labeling techniques are compared with the help of 3D PCA. It helps mainly is studying the distribution of different genes in a genome and how the labels are associated to it and the accuracy of labels.

2.7 Machine Learning

Machine learning (ML), which can be termed a subset of Artificial Intelligence, is a system specifically developed to automatically extract, combine, and synthesize ideas from massive data sets. It then expands this knowledge on its own by finding new information. The following, to put it briefly, are some applications for ML algorithms: To take appropriate corrective action beforehand, it is necessary to:

- gain a deeper understanding of the event that generated the data under study.
- capture the event's understating in the form of a model.

- predict future values that the event will generate based on the constructed model; and
- proactively detect any anomalous behavior of the phenomenon.

Machine learning is a rapidly evolving field. Recent technological advancements, particularly in the form of intelligent algorithms, computational capabilities like quantum computing and improved hardware and storage systems, have made it feasible to carry out numerous tasks more accurately and efficiently than they were even a few decades ago. Deep learning (DL), a specialized branch of machine learning that uses increasingly intricate architectures, algorithms, and models to solve challenging problems and forecast complicated event outcomes, has also developed during the last several years. The utilization of various machine learning approaches has become necessary to tackle the growing challenges in biology and clinical research due to the rapid expansion of biomedical and sequenced data in recent times. These techniques make it possible to automatically pick, extract features, and create predictive models, which makes it possible to research complex biological systems effectively. To improve training and validation, find the most interpretable features, and facilitate feature and model exploration, machine learning techniques are often combined with bioinformatic approaches, curated databases, and biological networks [29].

Machine learning algorithms can be classified into four types.

- Supervised learning: training is undergone using labeled data.
- Unsupervised learning: training is undergone without labeled data.
- Semi-Supervised Learning: training is undergone using both labeled and unlabeled data.
- Reinforced learning: training is undergone via a feedback-based process.

2.7.1 Supervised Learning

Supervised learning is a type of machine learning where the aim is to understand the relationship between input and output data using a set of input-output pairs known as training sample [30]. These samples are labeled, meaning each input is paired with an expected output, which acts as a guide or "supervision" for the learning process. The labeling throws light on the characteristics of the data set we have. This method is sometimes known as

Learning with a Teacher, Learning from Labeled Data, or Inductive Machine Learning. The aim is to develop a system that can infer the input-output mapping to predict outputs for new, unseen inputs. When the outputs are discrete, the system performs classification, categorizing input data into classes. When outputs are continuous, the system performs regression, predicting a numerical value. The relationship between inputs and outputs is usually captured through model parameters, which are estimated from the training data when not directly observable. Unlike unsupervised learning, which uses unlabeled data, supervised Learning requires data with labels. Semi-supervised Learning combines both supervised and unsupervised approach, while active learning involves the algorithm actively seeking labels from a user.

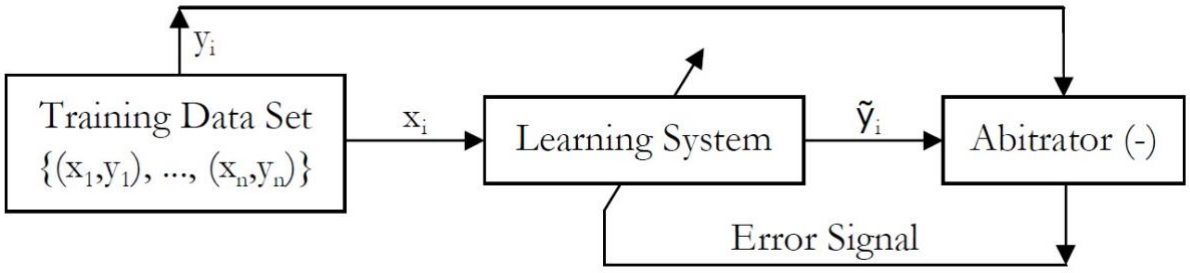

***Figure 3**: Schematic representation of supervised machine learning algorithm*

This diagram (Figure 3) delineates a supervised training sample as (x_1, y_1), where 'x' signifies the input to the system, 'y' denotes the output or the labeling of input x, and 'i' indicates the sample index. Within the Supervised Learning framework, an input x_i is fed into the Learning System, which then produces an output $\tilde{y}_i$. This output $\tilde{y}_i$ is subsequently compared with the actual label y_i by an evaluator, which calculates the discrepancy between them. This discrepancy, referred to as the Error Signal in the diagram, is utilized to inform adjustments within the Learning System to refine the learner's parameters. The overarching objective of this learning mechanism is to refine the Learning System's parameters to minimize the difference between the produced output $\tilde{y}_i$ and the actual label y_i across all samples, thereby reducing the aggregate error across the training dataset.

2.7.2 Unsupervised Learning

In unsupervised learning the machine receives inputs *x1, x2*, . . ., but obtains neither supervised target outputs, nor rewards from its environment. However, it is possible to develop of formal framework for unsupervised learning based on the notion that the machine's goal is to build representations of the input that can be used for decision making, predicting future inputs, efficiently communicating the inputs to another machine, etc. In a sense, unsupervised learning can be thought of as finding patterns in the data above and beyond what would be considered pure unstructured noise. Two very simple classic examples of unsupervised learning are clustering and dimensionality reduction.

2.7.3 Random Forest

The main idea behind the classical Random Forest (RF) algorithm is to create a big and diverse group of basic trees that decide between two categories [31]. These trees are made using different samples from the training data (Figure 4). At the decision-making time, the algorithm looks at what all these individual trees say and uses that information to make a smoother, more accurate decision. This usually makes the algorithm better at deciding between categories. While the trees are growing, for each place where the tree decides between two options, the algorithm picks a new set of predictors (or factors) to look at. This helps make sure that no single predictor is always the most important one, which makes each tree in the forest different from the others. This is different from another method called bagging, where every possible predictor is considered at every decision point.

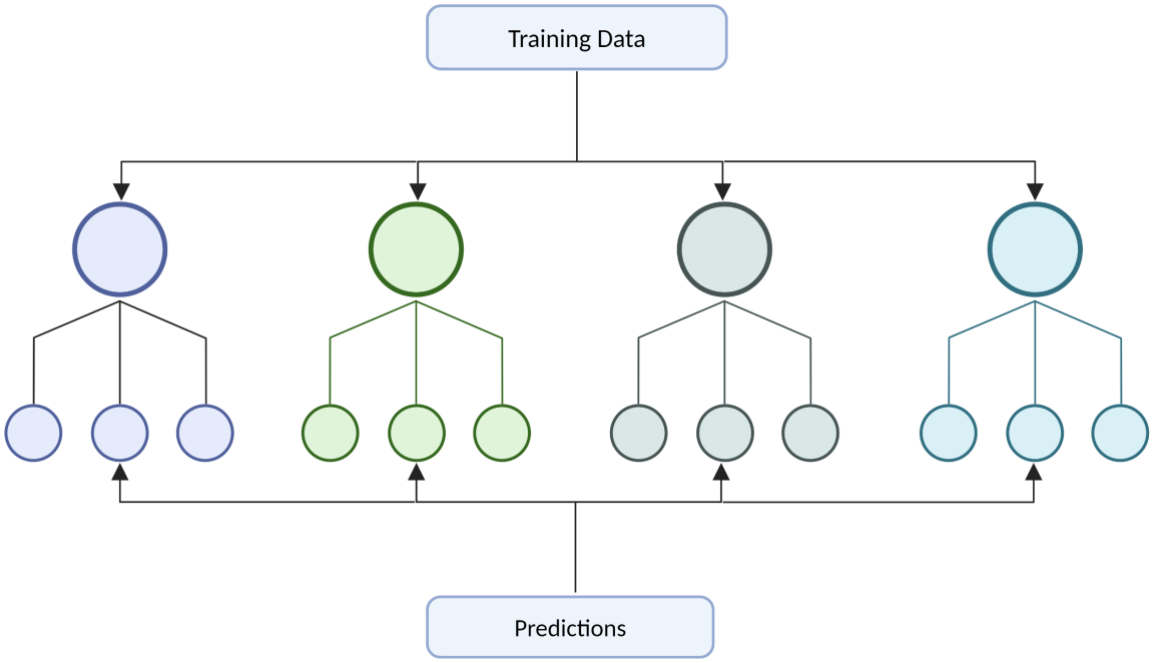

***Figure 4**: Schematic representation of random forest algorithm*

2.7.4 Decision Tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.

2.7.5 Logistic regression

Logistic regression is essentially a classification algorithm. The word *regression* in its name comes from its close sister in the regression domain known as *linear regression*. Given that the classes are discrete in supervised classification problems, the goal for the algorithms is to find the *decision boundaries* among the classes. Decision boundaries separate examples of one class from another. Depending on the problem instance, decision boundaries may be complex and nonlinear in geometric shape. In general, different machine learning algorithms have different assumptions regarding the shape of decision boundaries.

2.7.6 Support Vector Machine

A support vector machine (SVM) is a supervise machine learning algorithm that can classify data by finding an optimal line or hyperplane that maximizes the distance between each class in an N-dimensional space. SVMs are commonly used within classification problems. They distinguish between two classes by finding the optimal hyperplane that maximizes the margin between the closest data points of opposite classes. The number of features in the input data determine if the hyperplane is a line in a 2-D space or a plane in a n-dimensional space. Since multiple hyperplanes can be found to differentiate classes, maximizing the margin between points enables the algorithm to find the best decision boundary between classes. This, in turn, enables it to generalize well to new data and make accurate classification predictions.

2.7.7 Autoencoder

Autoencoder is an unsupervised learning technique in which neural networks are used for the tasks of representation learning [32]. A neural network architecture is designed in a way to impose a bottleneck in the network which forces a compressed knowledge representation of the original input (Figure 5). If the input features vectors are each independent of one another, this compression of this data and subsequent reconstruction would be a difficult task. However, if some sort of patterns exist in the dataset (i.e. correlations between input), this structure can be learned and consequently leveraged when forcing the input through the network's bottleneck.

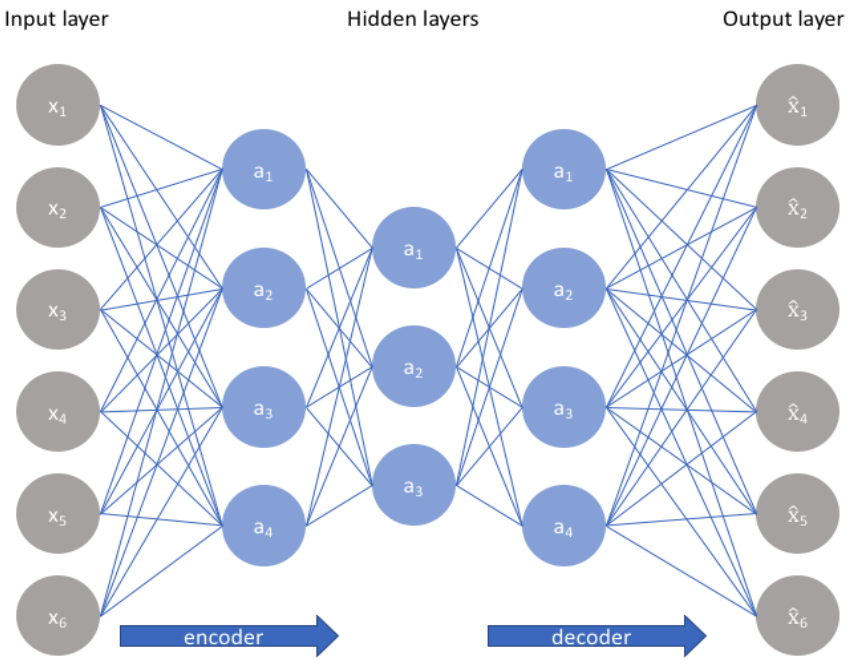

Figure 5: *Schematic representation of Autoencoders*

2.8 Labeling datasets for supervised training model

To develop a supervised machine learning model, labeled training datasets categorized into TG, HGT, CT, and CR were required. This necessitated extensive data collection, which was facilitated by taXaminer, a tool specifically designed for identifying contamination and horizontal gene transfer (HGT) among genes in annotated assemblies. Among the output files generated by taXaminer, two files, namely *gene_table_taxon_assignment.csv* and *taxonomic_hits.txt* were utilized for this purpose.

The *gene_table_taxon_assignment.csv* file contained crucial information, referred to as 'features,' about all genes within a species. Key features included *c_num_of_genes*, indicating the number of genes annotated on a given contig; *c_len*, representing the length of the contig; and *c_cov*, denoting the mean read coverage of the given contig. Data outputs from taXaminer were collected for species spanning various kingdoms, such as fungi, plants, mammals, bacteria, protozoa, invertebrates, and vertebrates and archaea.

For each species within these categories, both the *gene_table_taxon_assignment.csv* and *taxonomic_hits.txt* files were collected and saved for further analysis. Additionally, a csv file was generated acting as a database, which contained species names and their taXaminer run

status, indicating whether their data was available or not. This database facilitated the labeling process.

2.8.1 Workflow of 5% quantile labeling technique

A quantile can be termed as a value below which a certain percentage of data falls. Quantiles are widely used in statistics for various purposes, including summarizing data distributions, identifying outliers, and comparing different data sets.

The initial method adopted for labeling was the *5% quantile technique* (Figure 6). To accomplish this, the gene_table_taxon_assignment files containing various gene feature vectors obtained from the taXaminer run was examined and analyzed. The first feature examined was the *is_target* column. This column contains binary values, where a value of 1 indicates that the target gene is assigned to its target taxon. Genes were classified as TG if the *is_target* value was 1.

Before proceeding with further categorization, a subset of feature vectors was chosen based on their significant role in distinguishing between HGT, CT, CR, and TG. The 5% quantile method involves a simple binary decision for labeling. Initially, a target gene is examined, and if it is assigned to its target taxon, it is identified as a TG. Subsequently, the remaining genes are inspected, and those supporting the contamination hypothesis are designated as CT. This is determined by sorting the columns *c_len* and *c_num_of_genes* in ascending order and calculating a cutoff value. All the genes that have c_*len* or *c_num_of_genes* number below the cutoff values are considered contamination (Listing 1).

```
def calculate_percentile_cutoffs(df, columns_to_check=['c_len', 'c_num_of_genes']):
    cutoffs = {}
    for column in columns_to_check:
        if column in df.columns:
            percentile_cutoff = df[column].quantile(0.05)
            cutoffs[column] = percentile_cutoff
    return cutoffs
```

***Listing 1**: Python code for calculating the cutoff values. The function returns cutoff values which is further used to divide the genes as contaminations or not.*

For the genes that are not initially labeled, their taxonomic assignments are cross verified using reference species. If available, the corresponding genes are identified based on the best hit from the *taxonomic_hits.txt* file. Subsequently, the specific feature vectors of these genes are analyzed. The 5% quantile method is then applied to these feature vectors. Following sorting of the feature vectors of the reference species, the characteristics of the target gene are evaluated. If the target gene supports the contamination hypothesis in the reference species table, it is classified as CR, while the remaining genes are considered potential candidates for HGT. Genes without available reference feature vector tables are categorized as unknown genes. The resulting output of this labeling technique was a CSV file containing all feature vectors extracted from the taXaminer run, augmented with an additional column titled *target* appended at the end of the table, encompassing four distinct labels.

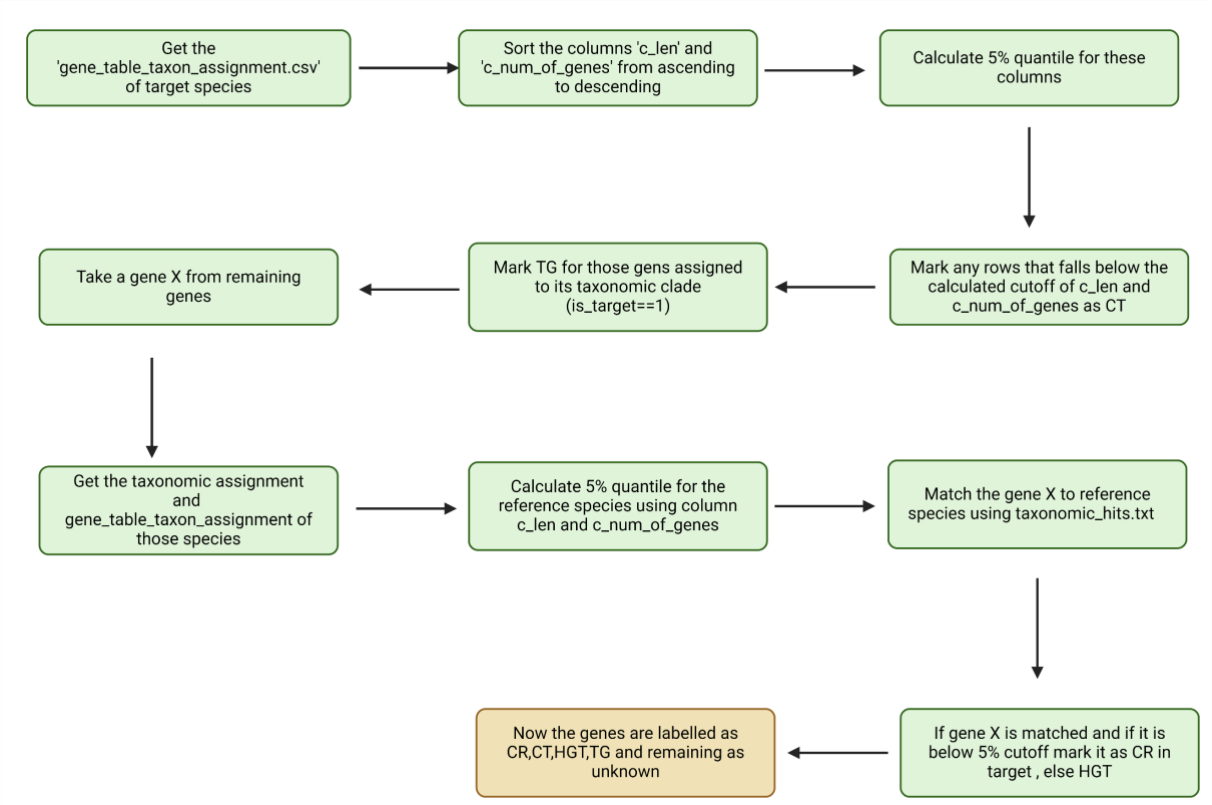

***Figure 6**: Workflow of 5% quantile labeling technique*

2.8.2 Workflow of Up/Downstream Labeling technique

Another approach to labeling genes is the upstream/downstream labeling method (Figure 7). An upstream gene is located immediately before the target gene, while a downstream gene is

situated immediately after the target gene. Analysis of the 5% quantile labeling technique revealed a clustering tendency among TG, HGT, and CR genes, whereas CT genes often appeared distant from the primary cluster. If a gene X is assigned to its target taxon, it will have an *is_target* value of 1. Therefore, if a target gene X has three upstream/downstream genes assigned to its own taxon, it was labeled as a TG. If a gene was not assigned to its target taxon (i.e., *is_target* value equals 0) and all six neighboring genes have *is_target* equal to 1, then the gene under consideration was flagged as a potential candidate for HGT. If a gene has an *is_target* value of 1 and is surrounded by neighboring genes with *is_target* values of 0, they were classified as CT. Genes with an *is_target* value of 0 that were not classified as HGT underwent further analysis. The taxonomic assignment of these genes was examined, and their feature vector table was studied. The rationale behind this was to assess the behavior of these non-target genes in their own genome assembly. If these genes are well integrated into their own genome assembly, it could indicate potential HGT in our target assembly. However, if these genes are contaminated or not well integrated with their own assembly, it suggests that these specific genes are contaminants from the reference (CR) in our assembly. All other genes that do not meet any of these criteria were labeled as unknown genes (Figure 8).

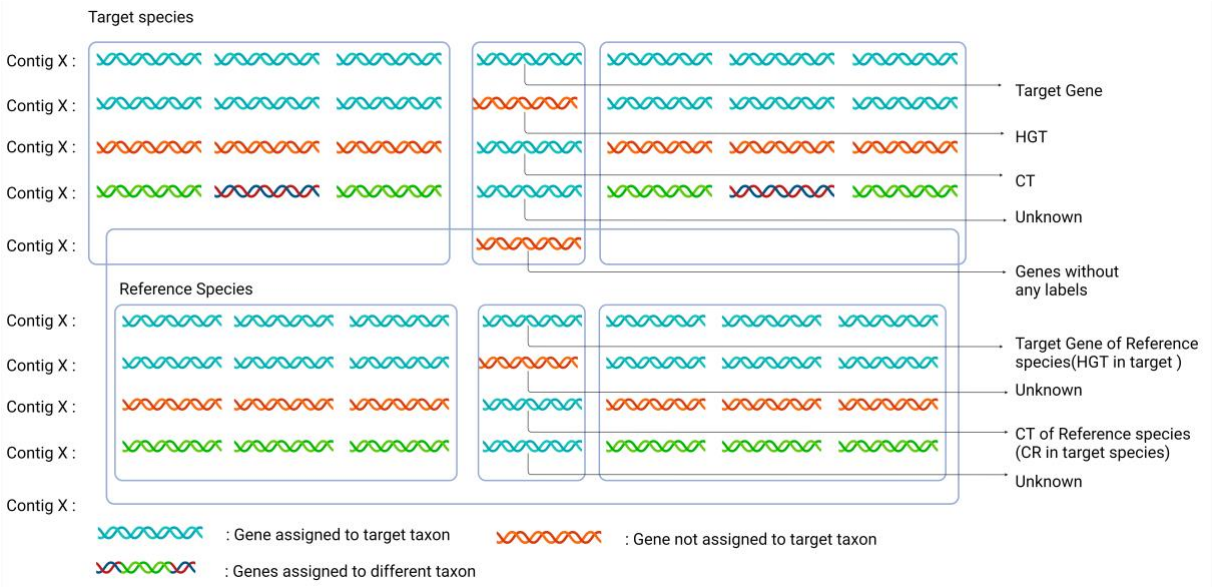

Figure 7: *Schematic depiction illustrating the labeling process of different genomic patterns for upstream and downstream genes within a genomic assembly. Blue strands represent genes assigned to the same taxon, while green strands indicate genes not assigned to the target taxon. The upper section illustrates the labeling of genes for the target species genomic table, while the lower section depicts labeling corresponding to the reference species genomic table.*

The procedure for examining unknown genes within the reference species mirrored that of the 5% quantile labeling technique. Similarly, a final csv output file was generated, like the 5% quantile labeling model. Fifty random species from various kingdoms were selected, and this labeling technique was applied to generate the labeled dataset needed for training.

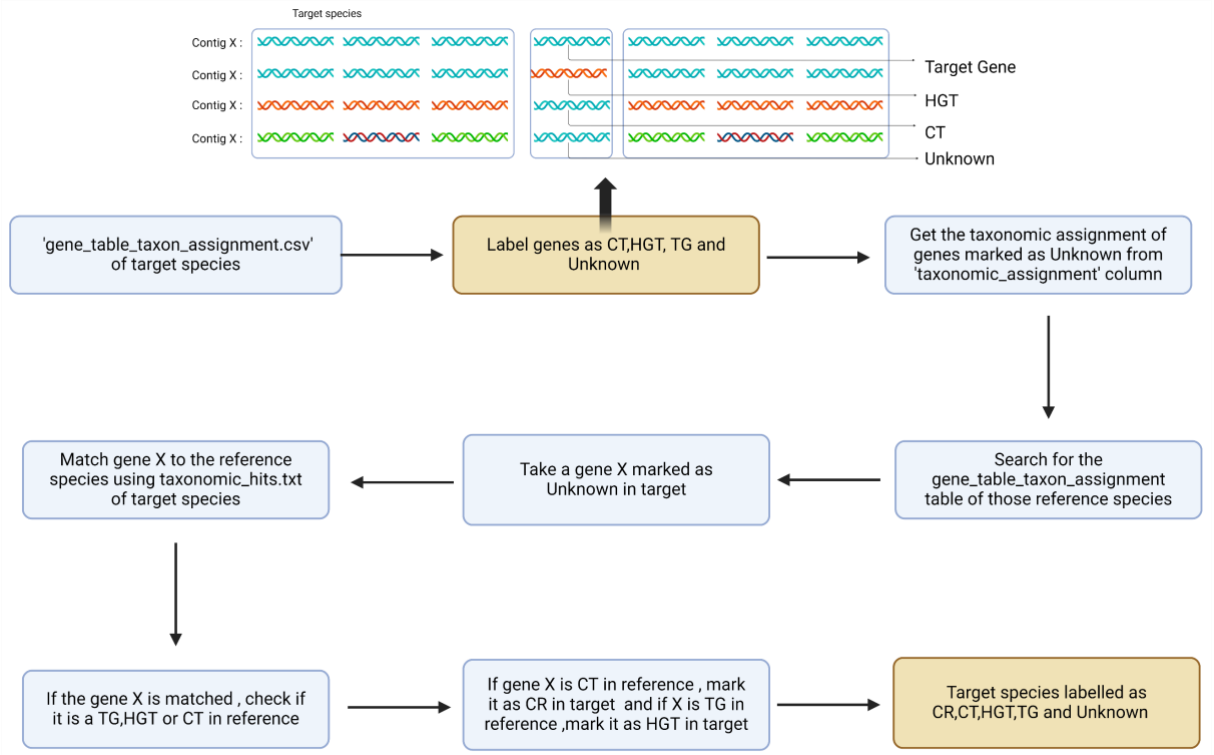

Figure 8: *Workflow of Up/Downstream Labeling technique*

2.8.2.1 Short contig approach using up/downstream technique.

A different and straightforward approach was adopted for the classification of contigs containing six or fewer genes. This threshold was chosen because the focus was on the six neighbors corresponding to the gene under study. The *is_target* column of these genes was examined, and they were labeled as TG only if all genes in the short contig had an *is_target* value of 1. Conversely, if all these genes had an *is_target* value of 0, they were classified as CT.

3. RESULTS

In this section, the comprehensive results of the studies, encompassing the analyses and findings derived from the collected data using different labeling techniques and conducted validation and comparison experiments are presented.

3.1 Performance of 5% quantile labeling technique

The PCA plot of *Lasallia pustulata*, (Figure 9), was generated using a 5% quantile labeling technique to assign various labels specific to genes. Unknown genes, for which taXaminer output was unavailable or unsuccessful, are represented in the plot. This labeling method effectively differentiated between three clusters of genes, as evident from the figure. Blue dots indicate target genes (TG) of *Lasallia pustulata*, totaling 5903 genes. Red dots represent CT, comprising 193 genes identified by the labeling technique. These contaminated genes exhibit distinct feature vectors compared to target genes, with many having short contig lengths. Additionally, a small cluster of target genes was observed to be mixed with these contaminated genes. CR, consisting of 18 genes, and HGT, comprising 253 genes, were integrated with the main cluster, indicating similar feature vectors. Another cluster on the left side of the plot contained contaminated genes along with some target genes.

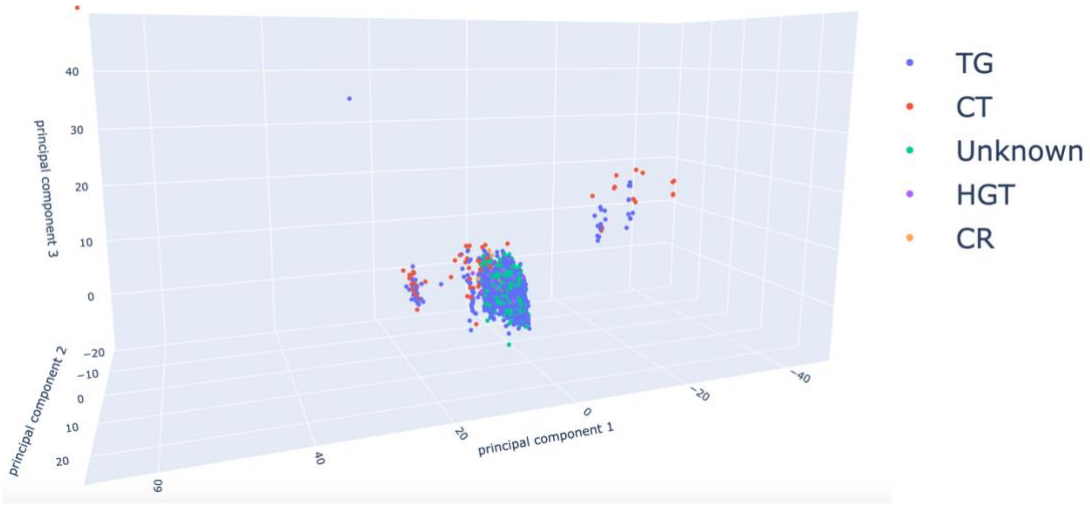

Figure 9: *3D PCA plot of Lasallia pustulata genomic assembly with 4 different labels generated by 5% quantile labeling technique. Blue dots indicate TG, red dot indicate CT, yellow dots indicate CR, purple dots indicate HGT, and green dots indicate unknown genes.*

3.1.2 Supervised machine model with 5% quantile labeling technique

Initially, a supervised machine learning model was constructed using labeled gene sets obtained through the 5% quantile technique, merging all labeled genes from two species, *Lasallia pustulata* and *Gigantopelta aegis*. Various models, including Decision Tree, Random Forest, Logistic Regression, and Support Vector Machine, were deployed.

Despite achieving high accuracy, the precision and recall of all models were notably low, except for TG and HGT, mainly due to the underrepresentation of the CT and CR datasets (Table 4). Notably, the Random Forest model outperformed the other three models. Hence, a larger dataset was deemed necessary for developing an effective supervised model. Consequently, since the 5% quantile method was only suitable for long-read continuous assembly, an alternative approach for labeling the dataset was devised.

a

Algorithm : Decision Tree :
Accuracy : 0.93

Class	Precision	recall	f1-score
TG	0.98	0.95	0.97
CT	0.33	1	0.50
CR	0.0	0.0	0.0
HGT	0.12	0.32	0.18

b

Algorithm : Random Forest :
Accuracy : 0.97

Class	Precision	recall	f1-score
TG	0.98	0.98	0.98
CT	0.33	1	0.50
CR	0.0	0.0	0.0
HGT	0.17	0.35	0.16

c

Algorithm : Logistic regression
Accuracy : 0.68

Class	Precision	recall	f1-score
TG	0.99	0.68	0.81
CT	0.14	1.00	0.25
CR	0.01	0.33	0.01
HGT	0.05	0.65	0.10

d

Algorithm : Support Vector Machine
Accuracy : 0.90

Class	Precision	recall	f1-score
TG	0.99	0.91	0.95
CT	0.25	1.00	0.40
CR	0.00	0.0	0.0
HGT	0.08	0.38	0.14

Table 4*: Performance comparison of machine learning models, a) Decision Tree, b) Random Forest, c) Logistic Regression, and d) Support vector machine, trained using the dataset labeled via the 5% quantile technique. Class 1 represents TG, 2 for CT, 3 for CR, and 4 for HGT. Random Forest showed superior performance overall, followed by Decision Tree, while Support Vector Machine displayed moderate accuracy. Logistic Regression exhibited the lowest accuracy among all the models.*

3.2 Performance of Up/Downstream Labeling technique

The Up/Downstream Labeling technique was validated by testing it with the *Lasallia pustulata* genomic assembly (Figure 10). The expected outcome was for TG, HGT, and CR genes to cluster together, while CT genes formed a distinct cluster. This expectation was confirmed in the genomic assembly using the new labeling approach.

The primary cluster, labeled as cluster 1, comprised TG, HGT, CR, and Unknown genes. Genes marked as "Not found" were those absent from the reference taXaminer output. Since these unknown and not found genes integrated well with the main cluster, it suggests they are not CT genes. The second cluster, situated near the main cluster, also included TG, HGT, and Unknown genes. CT genes formed a separate cluster, distinct from the main two clusters (cluster 1 and 2 in Figure 10), indicated in green. Additional examination indicated that these labeled genes were nearly identical to those categorized using the 5% quantile labeling technique, albeit with significantly fewer genes labeled using the Up/Downstream Labeling technique.

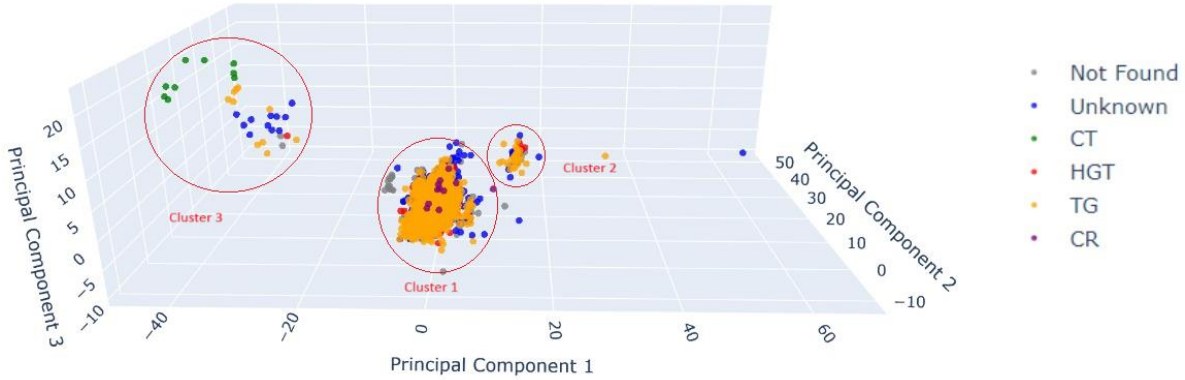

***Figure 10**: A 3D PCA plot illustrating the Lasallia pustulata genomic assembly, categorized into four labels using the Up/Downstream labeling technique. Each dot represents an individual gene within the genomic assembly. Violet dots within cluster 1 signify CR genes, while orange dots represent TG genes. Red dots denote HGT genes, and grey dots represent genes not found in the reference species assembly. Green dots indicate CT genes.*

To assess the effectiveness of the labeling technique across various genomic assemblies, a random set of species was chosen and labeled using this method. The investigation involved analyzing the distribution of gene clusters using 3D PCA. The results demonstrated varied characteristics among the assemblies; for instance, some only had genes labeled as TG or HGT, while others contained all gene categories except for CR. The genomic assembly of the species *Loxodonta africana* from the mammalian kingdom have CT genes distinctly separated from the main cluster of TG and HGT genes (Figure 11). No CR genes were identified in this assembly, and many unknown genes were grouped with the main cluster, suggesting they might include CR genes that weren't identified due to potentially lacking specific patterns in the reference taXaminer feature vector table. This absence of CR genes could be attributed to their rarity or non-appearance in the assembly.

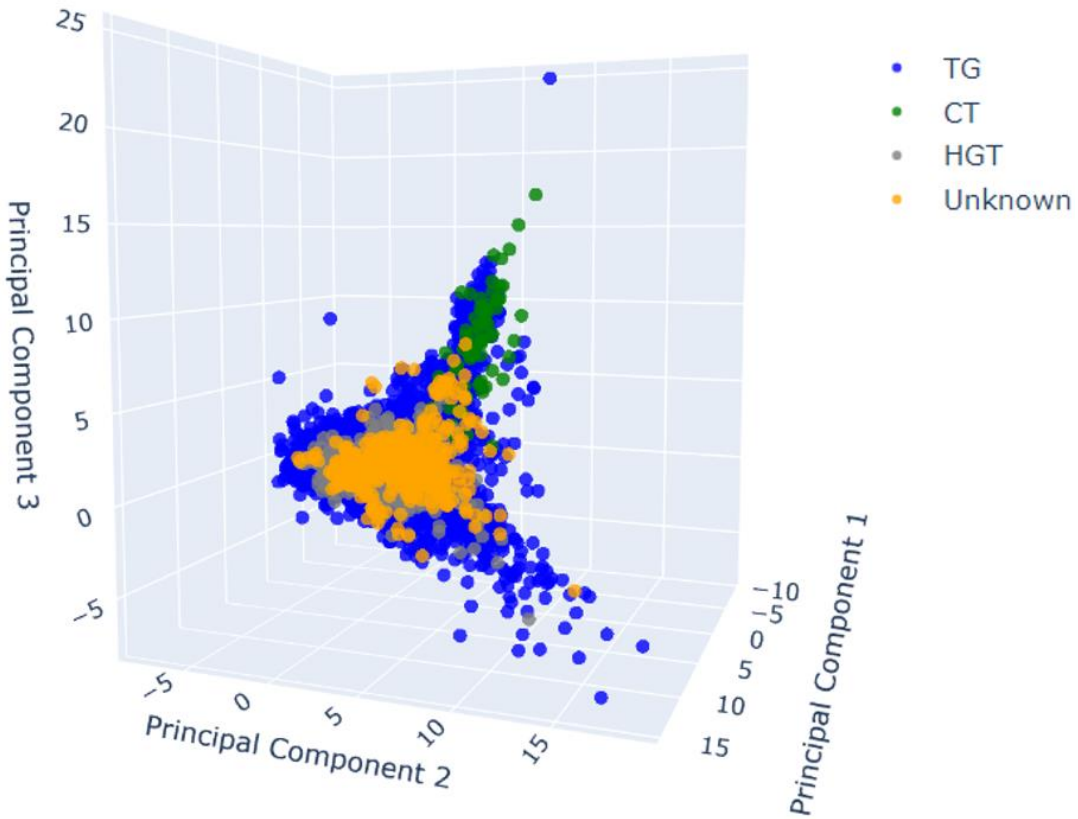

Figure 11: *3D PCA of Loxodonta africana labeled assembly. Each dot represents genes in the genomic assembly of Loxodonta Africana. The blue dots represent TG, grey dots represent HGT genes, green dot represent CT, and yellow dots represent unknown genes. The green dots are seen away from the center of the main cluster, an indication of potential candidates for contamination. The grey dots that are expected to be HGT are seen well integrated into the main cluster as expected.*

A similar situation was observed in the species *Monodelphis domestica* (Figure 12), where CT genes formed a separate cluster distinct from TG and HGT, with no CR genes detected. It's crucial to note that the absence of CR genes in these cases may be linked to missing gene information in the reference species taXaminer outputs, not necessarily due to the absence of the reference species files themselves. The automated labeling code produces two different outputs: one when the reference species information file is missing and another when all processes, including those involving the reference species file, are complete.

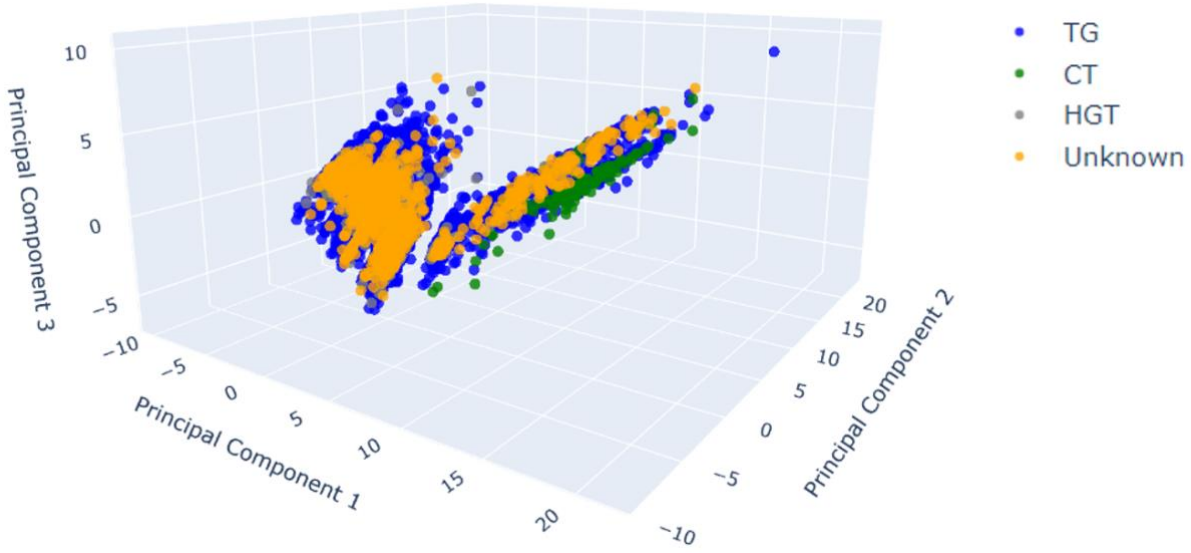

Figure 12: *3D pca of Monodelphis domestic labeled assembly: Two main clusters are observed in this assembly. The green dots, which are potential candidates for target contamination, is observed slightly away from the small cluster in right side. Some of the HGT genes are found inside the big cluster.*

3.3 Random Forest machine model with training data generated using Up/Downstream Labeling technique

3.3.1 Creating training data

Using the Up/downstream technique, 50 random taxa from various taxonomic clades including Archaea, Bacteria, Protozoa, Mammals, Fungi, non-mammalian Vertebrates, Invertebrates, and Plants were selected for labeling. However, only about 90 taxa ended up

being labeled. The lower number of labeled assemblies can be attributed to the unavailability of reference tables for some species and the lack of taxonomic hits tables for others.

3.3.2 Exploratory Data Analysis

The created dataset was analyzed, and it is considered a crucial step precedes to making a machine model. EDA helps in getting familiar with the data's format, the number, and data type of features and the first inklings of the relationships and underlying patterns between them (Figure 13).

EDA is also useful for solving data quality issues and where decisions about how to handle missing or invalid data are made. In general EDA helps in understanding the overall character istics of data. The data can be analyzed in graphical or non-graphical methods. Scatterplots, bubble charts, heatmaps are some of the methods used to depict the values by color.

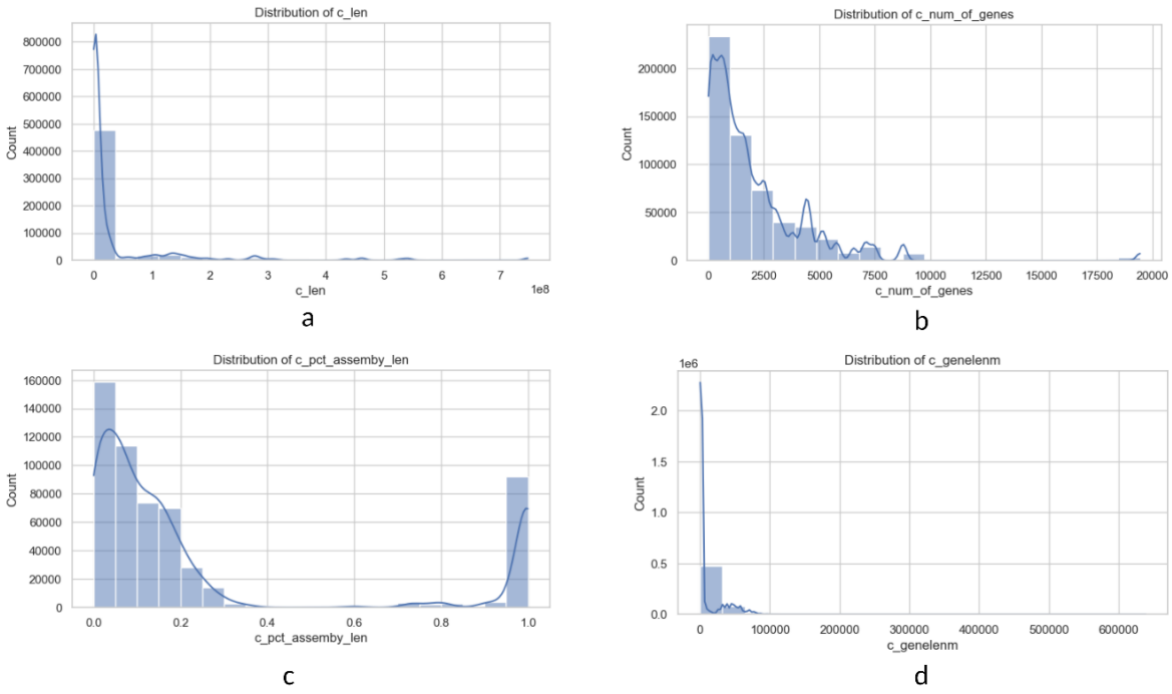

Figure 13: *The histograms representation of the distribution of different vectors a) C_len, b) c_num_of _genes, c) c_pct_assembly, d) c_genelenm.*

3.3.3 Spearman correlation coefficients heatmap of the vectors

Spearman's correlation coefficient is a statistical measure of the strength of a monotonic rela tionship between paired data (Figure 14).

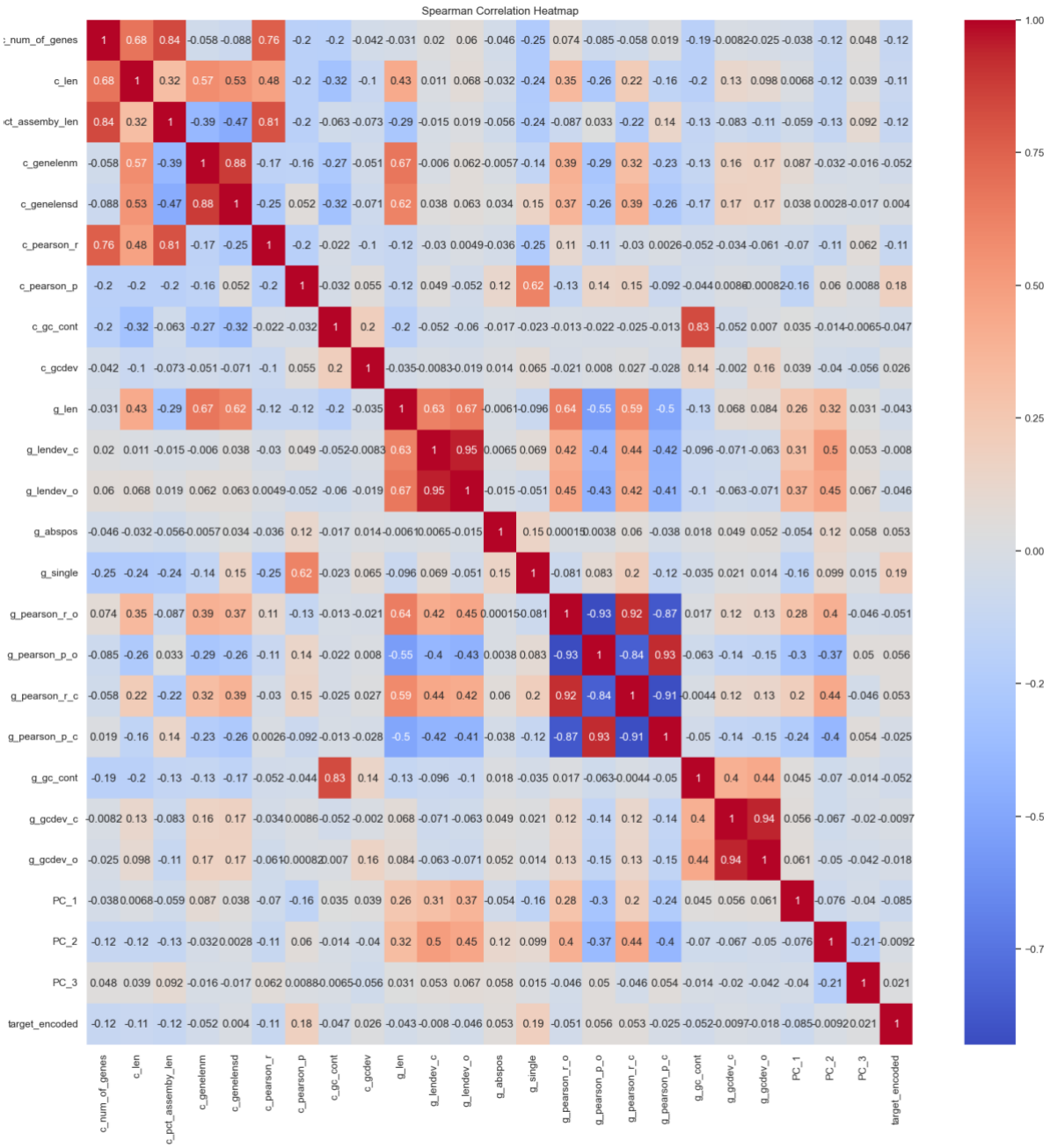

Figure 14: *In the heatmap a value close to 1 indicates a strong positive correlation, meaning as one feature value increases, the other also tends to increase. A value of -1 indicates that when one feature increases, other decreases and 0 indicates no correlation the given heatmap there are several features with strong positive or negative correlations with others. For example, c_num_of_genes seems to have a moderately strong positive correlation with c_len (0.68), indicating that longer contig s tend to have more genes. Also, c_pearson_r has a strong negative correlation with c_gc_cont (-0.82, suggesting that as the Pearson correlation coefficient increases, the GC content tends to decrease, or v ice versa.*

3.3.4 Target Label distribution

All the labeled csv files are loaded as a panda data frame and concatenated to form a big data frame that would act as the training data (Figure 15). The labels distribution are as follows:

- TG: 536022
- HGT: 15969
- CT: 8928
- CR: 312

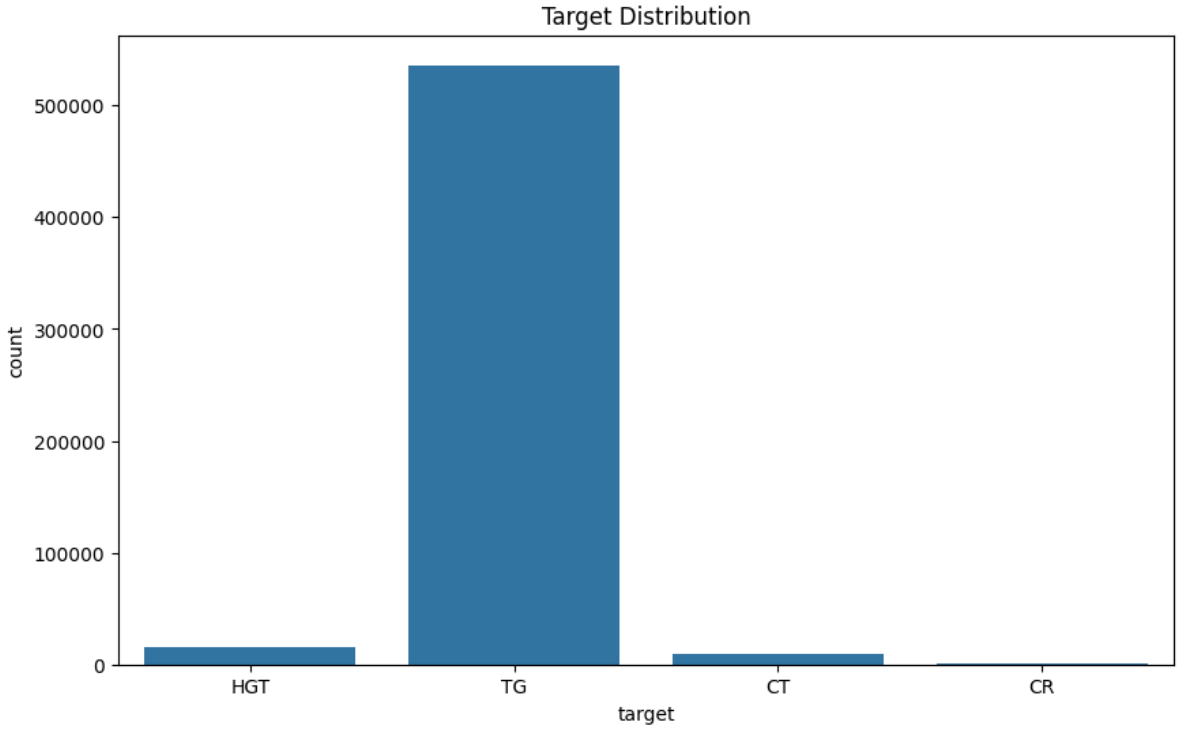

Figure 15: *Histogram representation of Target distribution of training data frame generated by Up/Downstream Labeling technique.*

In this target distribution the 'TG' category significantly outnumbers the other categories with 536,022 instances, indicative of a highly imbalanced dataset. In contrast, 'HGT' has 15,969 instances, 'CT' has 8,928, and 'CR' has the fewest with only 312 instances. This data was highly imbalanced, which could lead to a model bias favoring the more frequently occurring class. This imbalance is somewhat expected since efforts have been made to minimize contamination and HGT in various assembly databases, resulting in the least representation

of CR. To address this issue, Synthetic Minority Over-sampling Technique (SMOTE) was employed.

3.3.5 Data balancing using synthetic samples

SMOTE, accessed from the `imblearn.over_sampling` library, is initialized with a specific random state to ensure reproducibility. The `fit_resample` method is then applied to both the features and the target vector. This method generates synthetic samples in the feature space for underrepresented classes until a balanced class distribution is achieved (Listing 2). Consequently, the resampled dataset features an equal number of samples for each class, mirroring the count of the originally over-represented class.

```
from imblearn.over_sampling import SMOTE
smote=SMOTE(random_state=42)
X_resampled, y_resampled = smote.fit_resample(X,y)
```

Listing 2: *Python code used for data balancing.*

Scaling: The StandardScaler from the sklearn.preprocessing module was utilized to standardize features by removing the mean and scaling to unit variance. Standardization is a common requirement for ML models. The scaler is fitted on the resampled dataset, which means it calculates the mean and standard deviation to be used for later scaling (Listing 3). The transform method was then applied to scale the resampled dataset.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_resampled)
X_resampled_scaled = scaler.transform(X_resampled)
```

Listing 3: *Python code used for data scaling.*

After applying SMOTE, the class distribution became balanced, with 536,022 samples per class. This balanced distribution helps mitigate bias toward the majority class in learning algorithms and may enhance performance on minority classes (Table 5).

Target	Target encoded as:	Frequency
TG	4	536022
HGT	3	536022
CT	2	536022
CR	1	536022

Table 5: *Target labels are encoded into numerical values and underrepresented classes are sampled up to reduce bias.*

3.3.6 Random Forest model training

The random forest model was chosen because, it showed better results when a model was made on limited dataset, and this paved way for further developing the model with the help of random forest.

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model_fit(X_resampled_scaled, y_resampled)
```

Listing 4: *Python code used for random forest model training.*

RandomForestClassifier from the sklearn.ensemble module is instantiated with 100 trees (n_estimators=100) and a random_state of 42 for reproducibility (Listing 4).

Model Training: The fit method is called with the X_resampled_scaled dataset, which is the feature set that has been both oversampled and scaled. The target y_resampled is used, containing the class labels which are now evenly distributed after the SMOTE application.

3.3.7 Testing the model performance

The random model forest was tested using a previously labeled dataset of *Lasallia pustulata* (Figure 16). The testing data is data which is not seen by the model during the training phase. The labels in the testing data are hidden and after the prediction using the random forest model, the predicted labels are compared to the original label to create the evaluation matrices.

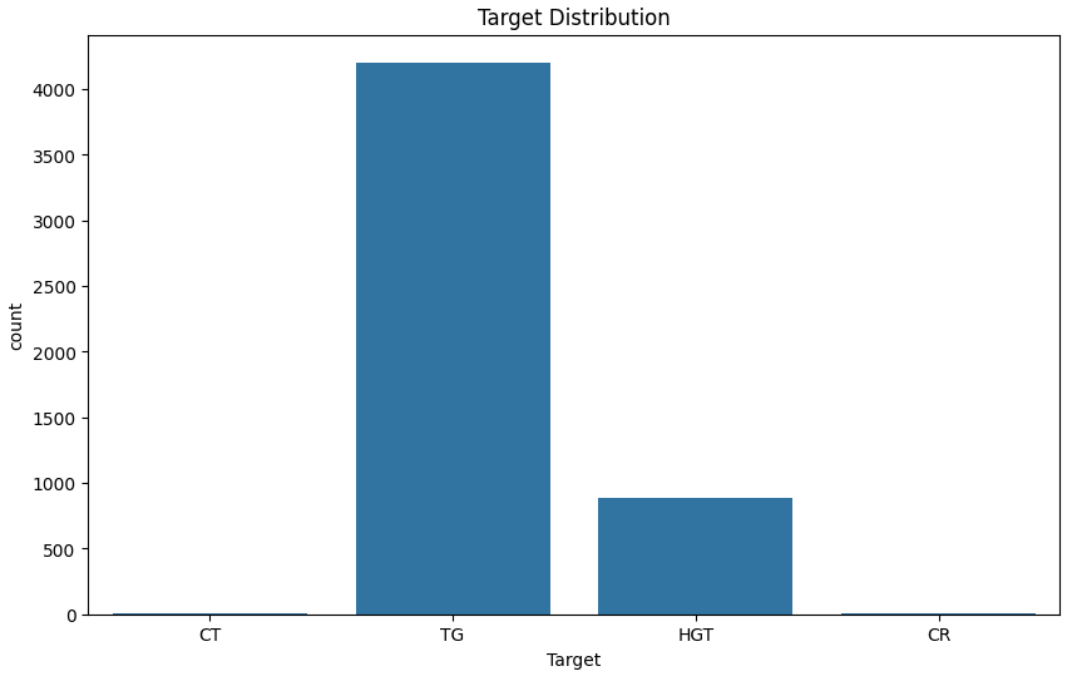

Figure 16: *Histogram representation of Target label distribution of testing data frame of Lasallia pustulata.*

3.3.8 Test dataset

The test dataset was also generated using up/downstream labeling technique. This dataset is not seen by model to avoid any bias. The test dataset labels are also encoded into the same numerical values. It consists of 4204 TG, 889 HGT, 11 CR and 9 CT.

3.3.9 Evaluation Metrics

Evaluation metrics are crucial aspects in assessing the performance of models. They provide quantitative measures that guide the selection and tuning of models. Different tasks require different metrics, and understanding which metric to use is key to interpreting model results effectively.

i. Accuracy:

It is the ratio of correctly predicted observation to the total observations.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

ii. Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision denotes low false positives.

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

iii. Recall

It is the ratio of correctly predicted positive observations to all observations in the actual class. High recall denotes low false negative rate.

$$\text{Recall} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

iv. F1 Score:

It is the weighted average of Precision and Recall. It takes both false positives and false negatives into account. It is a measure of a test's accuracy and is useful when the class distribution is not balanced. The F1 score is a good way to show that a model has a good balance between recall and precision.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

After testing the random forest model in an unseen dataset of *Lasallia pustulata* the results were as follows (Figure 17).

TG: This class had the highest number of correct predictions, with 4181 instances correctly identified as TG. There were 9 CT instances incorrectly predicted as TG, which shows a one-way misclassification from CT to TG. Furthermore, 11 TG instances were incorrectly classified as CR, indicating some confusion between TG and CR classes.

HGT (Class 3): A small number of instances (4) were correctly identified as HGT. However, 23 instances that were TG were misclassified as HGT. This suggests some degree of confusion between the HGT and TG classes. This is also because the HGT vectors are, to an extent, similar to the genes of Target species, except they might have a different taxonomic assignment.

CR: The model did not predict any instances as CR, which could imply that the model is not recognizing features pertinent to this class, or there are no instances of CR in the test dataset.

CT: Similar to CR, there were no instances predicted as CT, indicating a possible issue with the model's capability to discern this class or absence of this class in the test data.

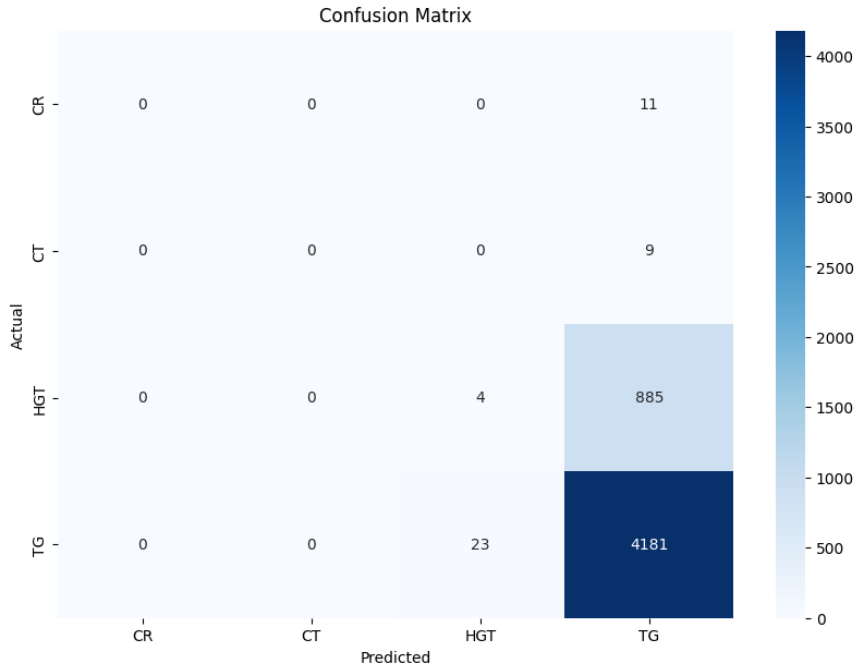

Figure 17: *Confusion matrix of the random forest model after testing of Lasallia pustulata dataset. The model predicts TG and some HGT but struggles with CR and CT as seen from the figure.*

The dataset was highly imbalanced, and these preliminary results generally show a bias toward the majority class which is the TG (Table 6).

Accuracy: 0.81
Precision:0.71
Recall: 0.81
F1 Score:0.74

Classification report:

Class	Precision	recall	f1-score
TG	0.82	0.99	0.90
CT	0.00	0.00	0.00
CR	0.00	0.00	0.00
HGT	0.15	0.00	0.01

Table 6: *Matrices showing model performance on testing data.*

4. DISCUSSION

This section demonstrates the findings from a genomic analysis performed using two distinct labeling techniques on the *Lasallia pustulata* genomic assembly: the 5% quantile method and the up/downstream labeling method. These methods were assessed for their effectiveness in classifying different gene types and for their integration with machine learning models to predict gene characteristics.

A supervised machine learning model can predict various categories of genes within a genomic assembly, including TG, CT, CR, and HGT. The critical aspect lies in optimizing the labeling technique used for both labeling and training the model. This task presents challenges, particularly with genomic assemblies exhibiting diverse behaviors. It's observed that labeling datasets or gene sets with long reads is relatively straightforward. However, for fragmented genomic assemblies consisting of numerous contigs, labeling becomes more challenging. The initial labeling technique employed, termed "5% quantile labeling technique," performed well with long-read genomic assemblies like *Lasallia pustulata*, revealing three distinct gene clusters as expected. This enabled comparison with the performance of the second labeling technique, Up/Downstream Labeling technique.

In the 5% quantile method of the 8,665 genes analyzed, 68.1% (5,903 genes) were categorized as TG, 2.3% (198 genes) as CT, 0.21% (18 genes) as CR, and 2.92% (253 genes) as HGT. In contrast, the up/downstream method identified 48.5% (4,204 genes) as TG, 0.10% (9 genes) as CT, 0.13% (11 genes) as CR, and 10.26% (889 genes) as HGT. The analysis revealed that the 5% quantile method identified 40.41% more TG genes than the up/downstream method. Moreover, the 5% quantile method labeled CT genes at a rate 22 times higher than the up/downstream method and identified 1.64 times more CR genes compared to the up/downstream method. Conversely, the up/downstream method identified 3.52 times more HGT genes than the 5% quantile method (Figure 18).

Significantly, the up/downstream method left approximately 41% (3,552 out of 8,665) of genes unlabeled, either due to unrecognizable patterns or the absence of taXaminer output from reference species. The 5% quantile method also encountered challenges, with about 26% (2,293 out of 8,665) of genes remaining unlabeled for similar reasons. These figures

underscore the substantial challenges posed by incomplete genomic data and complex genetic structures, which complicate effective gene classification.

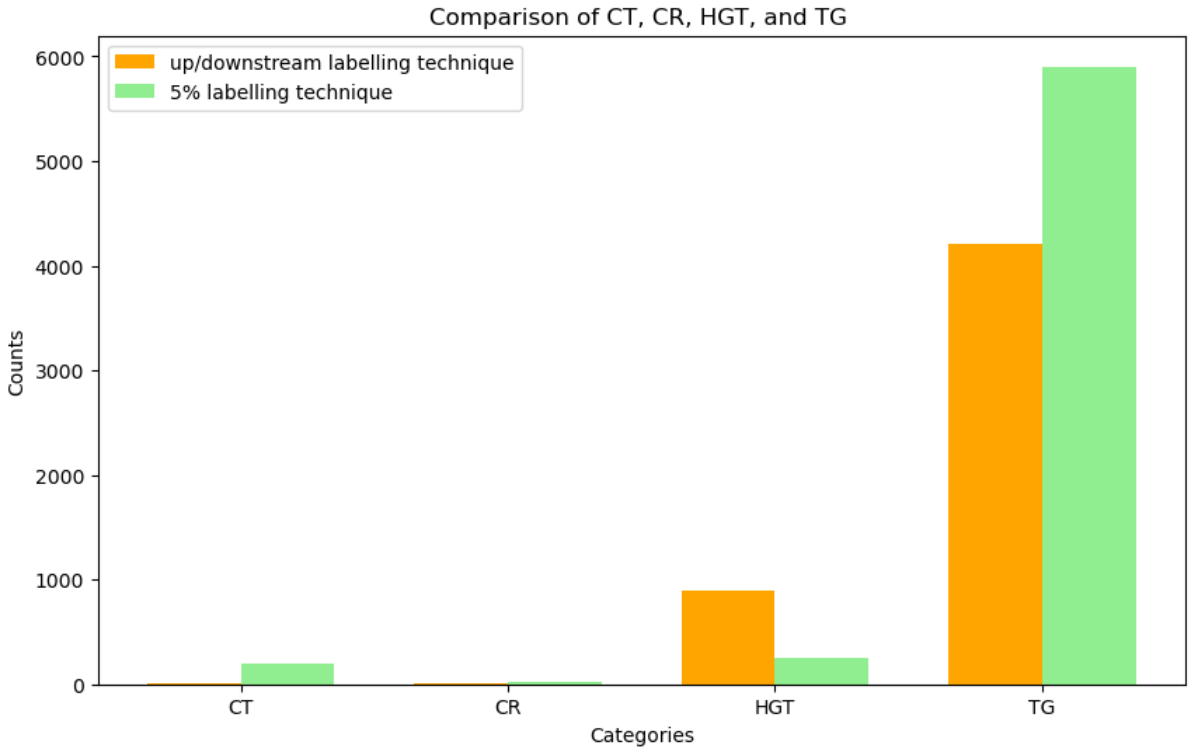

Figure 18: *Distribution of different categories of genes within Lasallia pustulata genomic assembly by two different labeling methods.*

The evaluation of four distinct machine learning models applied to data labeled by the 5% quantile technique showed varying degrees of accuracy. The Random Forest model achieved the highest accuracy at 0.97, followed by the Decision Tree at 0.93, SVM at 0.90, and Logistic Regression at 0.68. This outcome indicates that tree-based models, specifically Random Forest and Decision Tree, are more effective in managing the intricate genetic patterns and inherent noise present in genomic data, as compared to the linear models.

The labeled TG from both the 5% quantile and up/downstream methods were largely similar, with a significant majority forming a single large cluster. Additionally, the characteristics of

clusters from different labeled gene categories showed consistency between the two methods. For example, the CR genes identified by both methods, such as FUN_05301, FUN_06760, and FUN_09449, demonstrated a high level of consistency, reinforcing the reliability of these labels and highlighting the potential biological importance of these genes in the genomic assembly.

While the up/downstream method identified 536 more HGT genes than the 5% quantile technique, the PCA analysis showed that these genes integrated well into the main cluster of TG, suggesting that these additional labels might still represent valid data.

Contrastingly, the 5% quantile method labeled a wider array of CT genes, from FUN_09676 through FUN_09654 to FUN_09673, whereas the up/downstream method identified only a few, such as FUN_09856 and FUN_09847. This variance implies that the up/downstream method may be more attuned to specific genomic contexts or configurations not as easily detected by the 5% quantile method. Given the importance of accurate gene labeling and classification for training machine learning models, this sensitivity makes the up/downstream method potentially more advantageous, despite it generating fewer labels.

This analysis underscores the need to produce labeled datasets for a greater number of genomic assemblies, enhancing the pool of labeled data available for machine model training. Additionally, the 5% quantile method served as a validation tool for the up/downstream method, emphasizing its role in cross-verifying labeling accuracy.

A random forest machine model was developed based on the up/downstream labeling technique. The model exhibits high precision for the TG class, indicating its accuracy in predicting TG instances. However, it performs poorly in identifying CT and CR classes, as evidenced by zero precision and recall. Additionally, the HGT class shows low precision and recall, revealing a significant challenge in distinguishing it from other classes.

Although the model achieves high accuracy, it is mainly influenced by its performance on the TG class, which may be misleading when evaluating its effectiveness across all classes.

The training data suffered from significant imbalance due to low contamination and HGT instances in most assemblies. Despite attempts to address this through synthetic data generation, the model's performance remained unsatisfactory.

The imbalance stemmed from limitations in the labeling process, where the absence of feature vector and diamond hits tables for reference and target species resulted in skipped assemblies. Consequently, out of 400 selected assemblies, only 90 were successfully labeled. Scaling up this number would lead to a more balanced representation of CT, CR, and HGT classes, thereby improving overall accuracy and other metrics.
To effectively test the model, more datasets with adequate representation of each class are required.

As an addition to this, an unsupervised machine learning model using Autoencoders was developed and tested on *Lasallia pustulata* and *G. aegis*. The Autoencoder effectively differentiated between various gene clusters within the assemblies. However, despite attempts at fine-tuning with labeled datasets, minimal improvement was observed. Compared to the 5% labeled data, Autoencoders missed some CR genes within the assembly, predicting them as HGT.

To improve the ML model further, more assemblies from different taxa should be used so that the training data will be balanced organically. To improve the performance of autoencoders the number of hidden layers can be improved, and different feature selection methods can be employed to find better performing features in the training dataset.

5. OUTLOOK

This chapter demonstrates the overall perspective of the project, future directions, and recommendations for enhancing the outcomes.

The project focused on filtering out contaminations from genomic datasets, a crucial step due to its significant impact on data interpretation and research accuracy. To tackle this, a machine learning model was devised to categorize genes within a genomic assembly into four specific groups: TG, HGT, CT, and CR. Two distinct labeling methods were explored to train this model.

Initially, genes were labeled using a feature vector table derived from the taXaminer analysis of the genomic assembly. This table was rich with various feature vectors that described cellular characteristics, which were crucial for understanding gene classes. Important features included *c_len* for contig length and *c_num_of_genes* for the number of genes, along with a parameter for coverage. However, this method was primarily suitable for long-read genomic assemblies and faced limitations with short-read assemblies or where coverage data was unavailable.

To address these shortcomings, a second labeling technique was developed, focusing on the pattern of genes upstream and downstream. This new method was tested on various species, including *Lasallia pustulata*. Comparative PCA analysis of the results from both labeling methods for *Lasallia pustulata* demonstrated that the second method could effectively classify genes and was adaptable to short contig genes as well.

A Random Forest model was then implemented to predict the gene categories within the assemblies. This model showed proficiency in predicting TG and to some extent HGT but struggled with CT and CR due to their underrepresentation in the dataset. To improve this, automated labeling was executed across genomic assemblies from 400 species spanning various kingdoms, aiming to enrich the labeled dataset. However, because the CR category was infrequent, obtaining a sufficient number of genes labeled as CR for training the model presented a challenge.

Despite these challenges, the model was still effective in identifying whether a gene corresponded to its own assembly, primarily classifying TG accurately. Efforts to balance the

dataset through sampling the labeled dataset with the help of SMOTE were undertaken but did not yield substantial improvements.

Given the high dependency and specific conditions required by the supervised labeling process, an unsupervised learning approach using an Autoencoder was introduced. This model demonstrated potential in predicting CT, and HGT, (Figure 19-20) indicating a promising avenue for future research and analysis to refine and enhance genomic data classification.

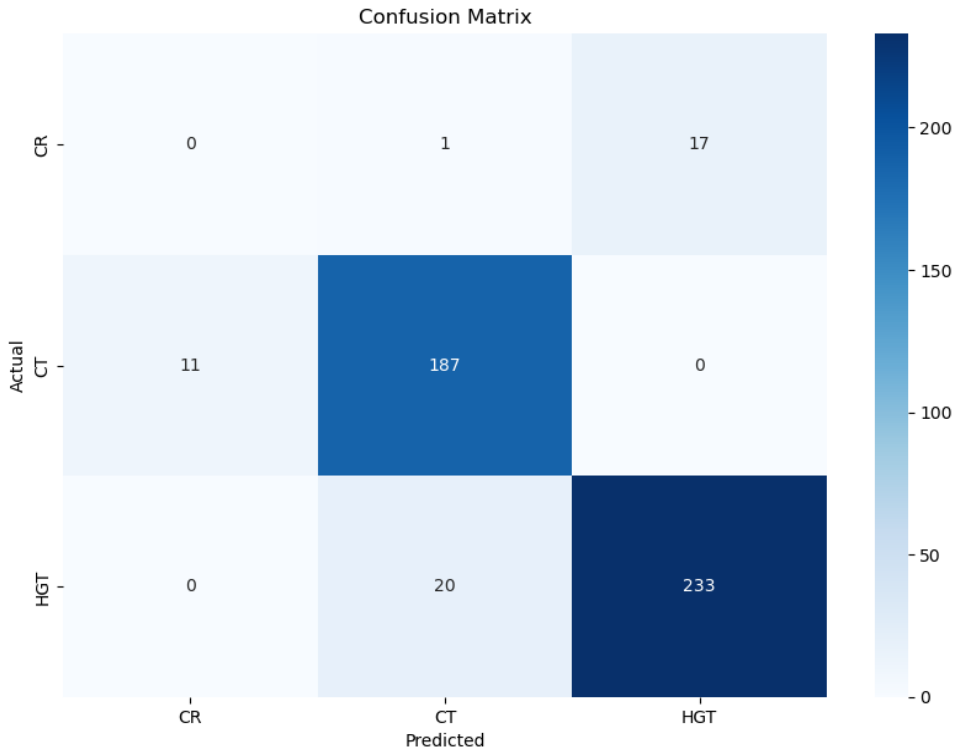

Figure 19: *The confusion matrix generated with the trained unsupervised autoencoder after testing on Lasallia pustulata data. The autoencoder predicted 233 HGT and 187 CT correctly as seen in figure but struggled to predict CR.*

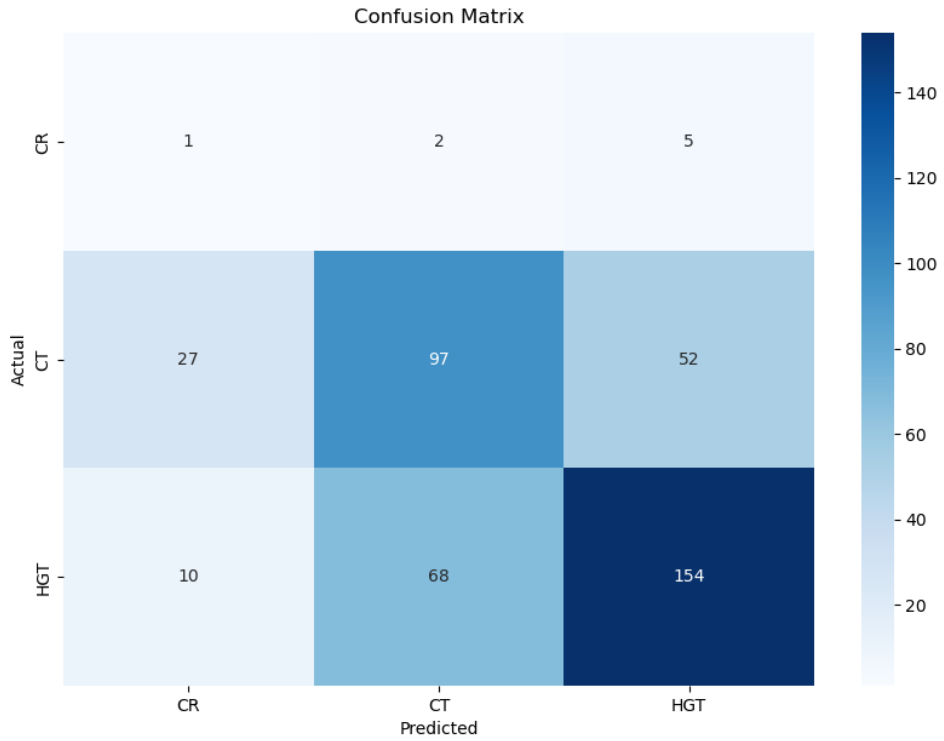

Figure 20: *The confusion matrix generated with the trained unsupervised autoencoder after testing on G. aegis data. The autoencoder predicted 154 HGT and 97 CT correctly as seen in figure but struggled to predict CR. It also classified 68 HGT as CT and 52 CT as HGT.*

The most critical step for enhancing the model is to increase and diversify the training dataset, particularly by obtaining more labeled data for the less common CT and CR categories. Although optimizing the labeling technique might not significantly impact these rare events compared to TG and HGT, it could still be beneficial to refine the method to label more genes, especially those that are currently unidentified in our genomic assemblies. Including these unknown genes could improve both the comprehensiveness and accuracy of the labeling technique and the training data. Another potential improvement could be to consistently incorporate coverage information across all genomic assemblies. This addition is vital as coverage data plays a crucial role in defining the characteristics of a gene.

This project revealed an inherent pattern among genes within a genomic assembly to an extent. The different gene clusters within a genomic assembly demonstrated the effectiveness of the Upstream/Downstream labeling method in distinguishing genes. The Random Forest model's ability to predict TG from the genomic assembly proved beneficial. This project could potentially pave way for a better classifier that can predict the target genes within an assembly further enhancing the process of genomic downstream analysis.

APPENDIX

All the codes used in this project is included in the GitHub repository.

REFERENCES

1) Alberts B, Johnson A, Lewis J, et al (2002). Molecular Biology of the Cell. 4th edition. New York: Garland Science; https://www.ncbi.nlm.nih.gov/books/NBK21054/
2) Alberts B, Johnson A, Lewis J, et al. (2002). Molecular Biology of the Cell. 4th edition. New York: Garland Science; From DNA to RNA. https://www.ncbi.nlm.nih.gov/books/NBK26887/
3) Ravi, et al (2018). MiSeq: A Next Generation Sequencing Platform for Genomic Analysis. Methods in Molecular Biology, 1706:223–232; DOI: 10.1007/978-1-4939-7471-9_12
4) Rhoads et al., (2015). PacBio Sequencing and Its Applications. Genomics, Proteomics & Bioinformatics, 13(5):278. DOI: 10.1016/j.gpb.2015.08.002
5) Wenger et al., (2019). Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. Nature biotechnology, 37(10):1155–1162. DOI: 10.1038/s41587-019-0217-9
6) Levy et al., (2019). Next-Generation Sequencing Strategies. Cold Spring Harbor Perspectives in Medicine, 9(7). DOI: 10.1101/cshperspect.a025791
7) Yoshinaga et al., (2018). Genome Sequencing. Methods in Molecular Biology, 1775:37–52. DOI: 10.1007/978-1-4939-7804-5_4
8) Sohn et al., (2018). The present and future of de novo whole-genome assembly. Briefings in Bioinformatics, 19(1):23–40. https://doi.org/10.1093/bib/bbw096
9) D G Fautin (1992). Annual Review of Ecology, Evolution and Systematics. Annual Reviews, https://archive.org/details/sim_annual-review-of-ecology-evolution-and-systematics_1992_23_contents
10) Crowson, R.A. (1970). Classification and Biology (1st ed.). Routledge. https://doi.org/10.4324/9781315081090
11) Hubbs, C. L. (1956). Ways of stabilizing zoological nomenclature. Proc. XIV Int. Congr. Zool.,Copenhagen1953:548-53. https://www.nativefishlab.net/library/textpdf/20061.pdf
12) Mugnai F et al., (2023). Be positive: customized reference databases and new, local barcodes balance false taxonomic assignments in metabarcoding studies. PeerJ. 9;11:e14616. doi: 10.7717/peerj.14616

13) Monica Santamaria et al., (2012). Reference databases for taxonomic assignment in metagenomics, Briefings in Bioinformatics, Volume 13, Issue 6, Pages 682–695, https://doi.org/10.1093/bib/bbs036

14) Koonin, Eugene & Galperin, Michael. (2003). Evolutionary Concept in Genetics and Genomics. DOI:10.1007/978-1-4757-3783-7_3

15) Needleman, S. B., & Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology, 48(3), 443–453. https://doi.org/10.1016/0022-2836(70)90057-4

16) Smith, T. F., & Waterman, M. S. (1981). Identification of common molecular subsequences. Journal of molecular biology, 147(1), 195–197. https://doi.org/10.1016/0022-2836(81)90087-5

17) Cornet et al., (2022) Contamination detection in genomic data: more is not enough. Genome Biol 23, 60. https://doi.org/10.1186/s13059-022-02619-9

18) Steinegger M et al., (2020). Terminating contamination: large-scale search identifies more than 2,000,000 contaminated entries in GenBank. Genome Biol. https://doi.org/10.1186/s13059-020-02023-1

19) Kumar S et al., (2013). Blobology: exploring raw genome data for contaminants, symbionts and parasites using taxon-annotated GC-coverage plots. Front Genet. 2013;4:237. doi: 10.3389/fgene.2013.00237

20) Eren AM et al., (2015). Anvi'o: an advanced analysis and visualization platform for omics data. PeerJ. PeerJ Inc.;3:e1319. DOI: 10.7717/peerj.1319

21) Tennessen K et al., (2016). ProDeGe: a computational protocol for fully automated decontamination of genomes. ISME J. 2016;10:269–72. DOI: 10.1038/ismej.2015.100

22) Mallet L et al., (2017). PhylOligo: a package to identify contaminant or untargeted organism sequences in genome assemblies. Bioinformatics. 2017;33:3283–5. https://doi.org/10.1093/bioinformatics/btx396

23) Pruesse E et al., (2012). SINA: Accurate high-throughput multiple sequence alignment of ribosomal RNA genes. Bioinformatics. 2012;28:1823–9 https://doi.org/10.1093/bioinformatics/bts252

24) Sayers, E. W et al., (2020). GenBank. Nucleic acids research, 48(D1), D84–D86. https://doi.org/10.1093/nar/gkz956

25) Keeling et al., (2008) Horizontal gene transfer in eukaryotic evolution. Nat Rev Genet 9, 605–618. https://doi.org/10.1038/nrg2386

26) Griffith, F. (1928). The Significance of Pneumococcal Types. The Journal of hygiene, 27(2):113–159. DOI: 10.1017/s0022172400031879

27) Ochman, H et al., (2000). Lateral gene transfer and the nature of bacterial innovation. Nature 2000 405:6784, 405(6784):299–304. DOI: 10.1038/35012500

28) Blokesch, M et al., (2015) Leben und sterben lassen – horizontaler Gentransfer in Vibrio cholerae . Biospektrum 21, 273–276 https://doi.org/10.1007/s12268-015-0572-0

29) Auslander N et al., (2021). Incorporating Machine Learning into Established Bioinformatics Frameworks. Int J Mol Sci. 12;22(6):2903. doi: 10.3390/ijms22062903. PMID: 33809353; PMCID: PMC8000113. DOI: 10.3390/ijms22062903

30) Liu, Qiong & Wu, Ying. (2012). Supervised Learning. 10.1007/978-1-4419-1428-6_451. DOI:10.1007/978-1-4419-1428-6_451

31) Daniel Ollech et al., (2020). A random forest-based approach to identifying the most informative seasonality tests∗. Bundesbank Discussion Paper No 55/2020. https://dx.doi.org/10.2139/ssrn.3721055

32) Umberto Michelucci (2022). An Introduction to Autoencoders . https://doi.org/10.48550/arXiv.2201.03898