Knowledge Engineering and Data Science (KEDS)
Vol 6, No 2, October 2023, pp. 231–248

pISSN 2597-4602
eISSN 2597-4637

Comparison of Machine Learning Algorithms for Species Family Classification using DNA Barcode

Lala Septem Riza [a,1,*], M Ammar Fadhlur Rahman [a,2], Yudi Prasetyo [a,3], Muhammad Iqbal Zain [a,4], Herbert Siregar [a,5], Topik Hidayat [b,6], Khyrina Airin Fariza Abu Samah [c,7], Miftahurrahma Rosyda [d,8]

[a] *Department of Computer Science Education, Universitas Pendidikan Indonesia*
Jl. Dr. Setiabudi No.229, Bandung 40154, Indonesia
[b] *Department of Biology Education, Universitas Pendidikan Indonesia*
Jl. Dr. Setiabudi No.229, Bandung 40154, Indonesia
[c] *Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Melaka*
110 off, Jalan Hang Tuah, Malaysia
[d] *Universitas Ahmad Dahlan*
Jl. Kapas No.9, Yogyakarta 55166, Indonesia
[1] *lala.s.riza@upi.edu*; [2] *mafr@student.upi.edu;* [3] *yudiprasetyo@upi.edu;* [4] *iqbalzain99@upi.edu;* [5] *herbert@upi.edu;* [6] *topikhidayat@upi.edu;* [7] *khyrina783@uitm.edu.my;* [8] *miftahurrahma.rosyda@tif.uad.ac.id*
* *corresponding author*

ARTICLE INFO

Article history:
Received 25 October 2023
Revised 27 October 2023
Accepted 03 November 2023
Published online 07 November 2023

Keywords:
Machine Learning
Supervised Classification
Species Classification
DNA Barcode
rbcL Gene
Data Analysis
Bioinformatics

ABSTRACT

Classifying plant species within the Liliaceae and Amaryllidaceae families presents inherent challenges due to the complex genetic diversity and overlapping morphological traits among species. This study explores the difficulties in accurate classification by comparing 11 supervised learning algorithms applied to DNA barcode data, aiming to enhance the precision of species family classification in these taxonomically intricate plant families. The ribulose-1,5-bisphosphate carboxylase-oxygenase large sub-unit (rbcL) gene, selected as a DNA barcode locus for plants, is used to represent species within the Amaryllidaceae and Liliaceae families. The experimental results demonstrate that nearly all tested models achieve accurate species classification into the appropriate families, with an accuracy rate exceeding 97%, except for the Naïve Bayes model. Regarding computational time, the Random Forest model requires significantly more time for training than other models. Regarding memory usage, the Least Squares Support Vector Machine with a polynomial kernel, and Regularized Logistic Regression consume more memory than other models. These machine learning models exhibit strong concordance with NCBI's classifications when predicting families using the test dataset, effectively categorizing species into the Amaryllidaceae and Liliaceae families.

I. Introduction

The development of living specimen processing technology [1] in recent decades has created many biological data, including *Deoxyribonucleic Acid* (DNA) sequence data. The collection of DNA sequences starts with taking samples from living organisms. The sample is then processed through various stages such as extraction, enumeration, and amplification to obtain pieces of DNA. These DNA fragments are then collected and sequenced to obtain the nucleic acid symbols (such as *adenine* (A), *guanine* (G), *cytosine* (C), and *thymine* (T)), which compose the DNA sequence [2]. The pieces of DNA sequences are then analyzed to obtain a genome that has been restructured so that it becomes a complete genome. That part of the genome is then selected as a barcode representing the species [3][4]. All these stages are depicted in Figure 1.

It has long been known that DNA sequences can be used to identify species, and nowadays, this activity is better known as DNA barcoding [5][6]. DNA barcoding is a method for identifying unknown specimens. It sequences in certain gene regions/loci that represent species in each kingdom, namely: *cytochrome C Oxidase subunit I* (COI) for animals [7] obtained from mitochondria in cells, *ribulose-1,5-bisphosphate carboxylase -oxygenase large sub-unit* (rbcL) and *megakaryocyte-associated tyrosine kinase* (matK) for plants [8] obtained from chloroplast cells, and internal transcribed spacer (ITS) for fungi [9] found in nucleus cells.

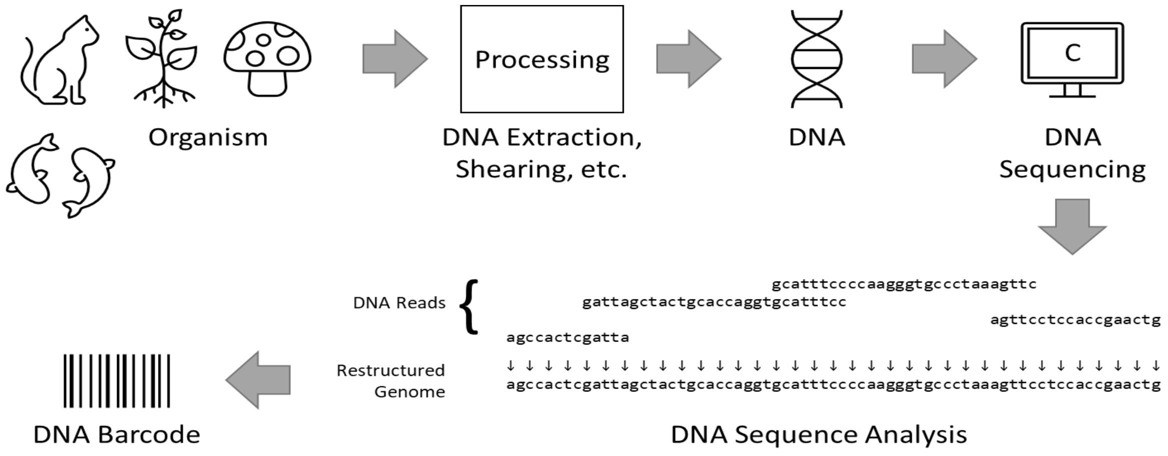

Organism
Processing
DNA Extraction, Shearing, etc.
DNA
C
DNA Sequencing
DNA Reads
gcatttccccaagggtgccctaaagttc
gattagctactgcaccaggtgcatttcc
agttcctccaccgaactg
agccactcgatta
Restructured Genome
agccactcgattagctactgcaccaggtgcatttccccaagggtgccctaaagttcctccaccgaactg
DNA Sequence Analysis
DNA Barcode

Fig. 1. Process of processing living specimens into DNA barcodes

The process of identifying species in DNA barcoding is done by analyzing the similarity of a barcode belonging to a specimen with another barcode belonging to a species already known in the database. The specimen can be classified as an existing species if the barcode has a high degree of similarity. If no barcode pairs are found with a high degree of similarity, then the specimen may be a new species and needs to be verified by a taxonomist.

Several approaches are commonly used to classify species in DNA barcodes: tree-based, similarity-based, and character-based [10][11]. The tree-based method classifies a barcode into species based on its membership in the DNA barcode tree. The similarity-based method classifies barcodes based on the number of similar characters in the DNA barcode. At the same time, the character-based method relies on the presence or absence of specific characters in the DNA barcode. In addition to these three approaches, species classification using DNA barcodes can also be treated as a case of machine learning problems with supervised learning [12][13][14][15][16].

The Liliaceae family, colloquially called the 'Lily Family', predominantly consists of *monocotyledonous* plants characterized by notable morphological diversity. Encompassing approximately 16 genera and over 610 species [17], members of this family manifest primarily as herbs and shrubs. They are predominantly distributed across temperate and subtropical regions [18]. The amphipathic properties inherent to certain compounds within Liliaceae render them effective as surfactants. Beyond their ecological significance, these plants exhibit multifaceted utility: they are esteemed for ornamental purposes and utilized as vegetables, and certain species are acknowledged for their medicinal properties. Given the vast potential inherent to the Liliaceae family, they hold promise in cosmetics and pharmaceutical development [19].

The Amaryllidaceae family, a prominent member of the order Asparagales, is distinguished by its bulbous flowering plants. These plants are celebrated for their visually captivating flowers, making them famous for ornamental cultivation [20]. From a taxonomic perspective, the Amaryllidaceae family is stratified into three subfamilies: Agapanthoideae, Allioideae, and Amaryllidoideae [21]. Historically, these were regarded as distinct families. The term "Amaryllidaceae" is recurrently cited in phytochemical and pharmaceutical literature, particularly in discussions centered on the Amaryllidoideae subfamily [20][22].

The medicinal potential of the Amaryllidaceae family is both historical and contemporary. Tracing back to the Classical period, luminaries like Hippocrates and Dioscorides harnessed the therapeutic properties of Narcissus oil, particularly for conditions believed to be associated with uterine tumors. In modern traditional medicine, the applications are diverse. For instance, Ammocharis is employed for blood purification and wound treatment, Brunsvigia for respiratory and hepatic ailments, Clivia for snakebites and facilitation of childbirth, and Crinum for a spectrum of conditions ranging from tumors to rheumatism [23].

In previous research, the Amaryllidaceae family was classified under the Liliaceae family. However, advancements in phylogenetics have led to a taxonomic reorganization. A team of scientists, spearheaded by Rolf Dahlgren [24], extensively examined monocot characteristics, including numerous microscopic features, culminating in a revised classification.

Historically, taxonomic experts such as Bentham and Hooker [25], Engler and Prantl [26], Bessey [27], Rendle [28], and Hutchinson [29] categorized Amaryllidaceae with an inferior ovary and Liliaceae with a superior ovary into distinct families based on ovary position differences. Despite these distinctions, both families exhibited numerous shared characteristics. Consequently, Cronquist [30] and Takhtajan [31] integrated the Amaryllidaceae family into Liliaceae. Further research regarded 'lilies' as a heterogeneous collection of genera and positioned them in families grouped under two orders: Asparagales and Liliales [32].

The problem in both families is depicted in the classification of *Allium albopilosum*. *Allium albopilosum*, indigenous to Turkestan, is cultivated for its notable utility as a cut flower. While traditionally, *Allium* species have been categorized under the Liliaceae family due to the presence of superior ovaries in their flowers, there exists a divergence of opinion among botanists. Some propose their reclassification to the Amaryllidaceae family, citing the characteristic umbellate inflorescence. Conversely, others advocate for a distinct classification, suggesting establishing a unique family, Alliaceae, to accommodate them [33].

The Consortium Barcode of Life [8] advocated the rbcL gene as a barcode for plant taxonomy and phylogenetic analysis. This gene is pivotal in plant species identification, phylogenetics, and relationships. The rbcL gene is located in chloroplast DNA [8]. Several studies have employed the rbcL gene for plant relationship research. For instance, the rbcL gene elucidates the relationships within Selaginellaceae [34]. Similarly, another research combined the rbcL gene with trnL-F for a phylogenetic study on Rhamnaceae [35].

Machine learning is a study attempting to extract knowledge from available data using computer programs that can learn and get smarter automatically based on experience [36][37]. Currently, the application of machine learning can be found in various activities in everyday life, such as recommendations for goods in Amazon e-commerce services [38], recommendations on the music streaming platform Spotify [39], and recommendations in education assessment [40][41][42]In bioinformatics, machine learning has been widely used to solve problems in various areas, including genomics, proteomics, systems biology, evolution, microarrays, and text mining [43][44] [45]. The application of machine learning in each case handles the different characteristics of the input data.

Based on the type of feedback from the input data, there are three forms of learning: supervised learning, unsupervised learning, and reinforcement learning [46]. Of the three forms of machine learning, bioinformatics case studies generally use supervised learning and unsupervised learning to solve problems. For example, supervised learning is used in genomics for the case of gene finding [47]. Another example is the application of Support Vector Machines (SVM) [48] and Random Forests (RF) [49] for the prediction of phenotypic effects [50]. An example of the application of unsupervised learning in bioinformatics is microarray science for clustering genes into groups with specific biological meanings [51].

This study attempts to compare supervised machine learning algorithms to predict families of species based on DNA barcode sequences in the R programming language. By predicting the family, we can more accurately place the species in the correct family in the taxonomy. Machine learning algorithms that are used in this research are Random Ferns, SVM Linear, SVM Poly, SVM Radial, SVM Radial Weights, LSSVM Poly, Naïve Bayes, Random Forest, C5.0, K-Nearest Neighbours, and Regularized Logistic Regression.

The DNA barcode sequence employed in this study is derived from a segment of the chloroplast gene specific to the rbcL gene region of each examined species. This research contributes to resolving the existing classification ambiguity between the Liliaceae and Amaryllidaceae families. It accomplishes this by applying various machine learning methodologies, the results of which are juxtaposed with contemporary, state-of-the-art classification systems from NCBI to yield more definitive insights into the precise familial categorizations.

II. Methods

A. Data Collection

The data used are DNA barcode sequence data obtained from GenBank [52] (ncbi.nlm.nih.gov, accessed August 15, 2023). The dataset contains rbcL enzyme sequences from the chloroplast gene of plants in the Amaryllidaceae and Liliaceae families. Information on the number of species, sequences, and file size of each dataset is listed in Table 1.

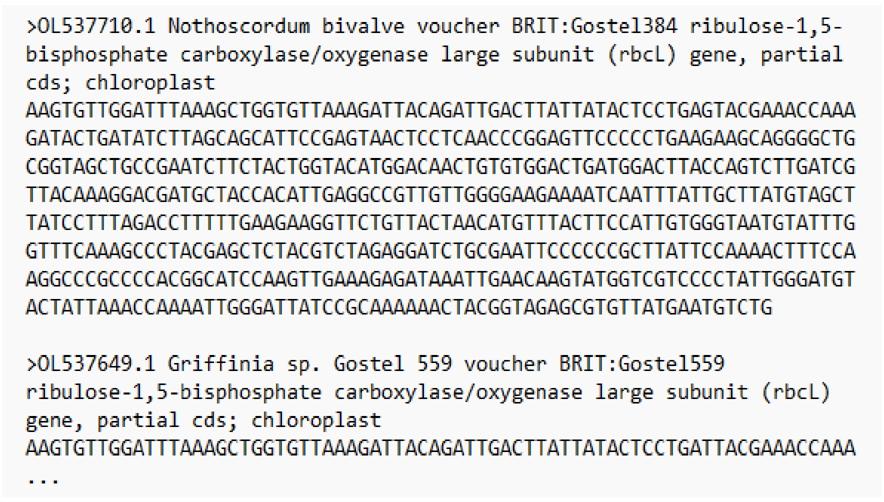
Table 1. Descriptions of the used datasets

Dataset	Number of Species	Number of DNA Sequences	File Capacity (kB)
		Training Data	
Amaryllis	308	689	708.4
Lily	331	713	784.3
		Testing Data	
Amaryllis	23	113	114.7
Lily	28	140	136.5
Total	690	1,655	1,743.9

The Amaryllis dataset contains 802 samples from the Amaryllidaceae family, of which 689 were used for training and 113 for testing. Meanwhile, the Lily dataset comes from the Liliaceae family and contains 853 samples, with details of 713 used for training and 140 for testing. All sequences in the dataset have varying sequence lengths (base pair; bp), with the most extended sequence having 1,458bp and an average sequence length of 903bp.

The training dataset was obtained by downloading all species sequences in the family and omitting several selected species in the Amaryllidaceae and Liliaceae families. The complete list of species omitted from the training dataset can be seen in Table 2. The testing dataset is a sequence of species omitted from the training dataset. The difference in the number of species in the testing dataset in Table 1 with the species in Table 2 is due to (1) not all species have samples of the rbcL gene sequence in GenBank at the time of data collection (example: *Allium chrysanthum*) and (2) GenBank distinguishes main species from varieties/sub-species (example: *Crinum asiaticum* and *Crinum asiaticum var. Japonicum*). All species collected in the testing dataset are listed in Table 3.

The entire dataset is downloaded and saved in FASTA format. Figure 2 shows an example of dataset content containing the GenBank accession number, species name, sequence description, and DNA sequence. Each sequence is indicated by a line starting with the greater than symbol (“>”) and ending with a blank line.



```
>OL537710.1 Nothoscordum bivalve voucher BRIT:Gostel384 ribulose-1,5-
bisphosphate carboxylase/oxygenase large subunit (rbcL) gene, partial
cds; chloroplast
AAGTGTTGGATTTAAAGCTGGTGTTAAAGATTACAGATTGACTTATTATACTCCTGAGTACGAAACCAAA
GATACTGATATCTTAGCAGCATTCCGAGTAACTCCTCAACCCGGAGTTCCCCCTGAAGAAGCAGGGGCTG
CGGTAGCTGCCGAATCTTCTACTGGTACATGGACAACTGTGTGGACTGATGGACTTACCAGTCTTGATCG
TTACAAAGGACGATGCTACCACATTGAGGCCGTTGTTGGGGAAGAAAATCAATTTATTGCTTATGTAGCT
TATCCTTTAGACCTTTTTGAAGAAGGTTCTGTTACTAACATGTTTACTTCCATTGTGGGTAATGTATTTG
GTTTCAAAGCCCTACGAGCTCTACGTCTAGAGGATCTGCGAATTCCCCCCGCTTATTCCAAAACTTTCCA
AGGCCCGCCCCACGGCATCCAAGTTGAAAGAGATAAATTGAACAAGTATGGTCGTCCCCTATTGGGATGT
ACTATTAAACCAAAATTGGGATTATCCGCAAAAAACTACGGTAGAGCGTGTTATGAATGTCTG

>OL537649.1 Griffinia sp. Gostel 559 voucher BRIT:Gostel559
ribulose-1,5-bisphosphate carboxylase/oxygenase large subunit (rbcL)
gene, partial cds; chloroplast
AAGTGTTGGATTTAAAGCTGGTGTTAAAGATTACAGATTGACTTATTATACTCCTGATTACGAAACCAAA
...
```

Fig. 2. RNN, LSTM, and GRU architecture development

Table 2. List of species selected for test data

No.	Amaryllis	Lily
1	*Agapanthus campanulatus*	*Alstroemeria aurea*
2	*Allium altaicum*	*Calochortus apiculatus*
3	*Allium cepa*	*Calochortus lyallii*
4	*Allium chrysanthum*	*Cardiocrinum cathayanum*
5	*Allium chrysocephalum*	*Cardiocrinum cordatum*
6	*Allium fistulosum*	*Cardiocrinum giganteum*
7	*Allium monanthum*	*Erythronium albidum*
8	*Allium obliquum*	*Erythronium americanum*
9	*Allium porrum*	*Fritillaria unibracteata*
10	*Allium prattii*	*Gagea serotina*
11	*Allium pskemense*	*Lilium bulbiferum*
12	*Allium sativum*	*Lilium davidii*
13	*Allium tuberosum*	*Lilium distichum*
14	*Allium xichuanense*	*Lilium fargesii*
15	*Amaryllis minuta*	*Lilium lancifolium*
16	*Crinum asiaticum*	*Lilium longiflorum*
17	*Crinum macowanii*	*Lilium pardalinum*
18	*Hymenocallis caribaea*	*Lloydia oxycarpa*
19	*Hymenocallis henryae*	*Medeola virginiana*
20	*Hymenocallis tubiflora*	*Nomocharis aperta*
21	*Lycoris radiata*	*Scoliopus bigelovii*
22	*Narcissus poeticus*	*Tricyrtis macropoda*
23	*Pancratium arabicum*	*Tulipa gesneriana*
24	*Zephyranthes candida*	*Zigadenus glaberrimus*
25	*Zephyranthes simpsonii*	

Table 3. List of species included in test data

No.	Amaryllis	Lily
1	*Agapanthus campanulatus*	*Calochortus apiculatus*
2	*Allium altaicum*	*Calochortus lyallii*
3	*Allium ampeloprasum*	*Cardiocrinum cathayanum*
4	*Allium cepa*	*Cardiocrinum cordatum*
5	*Allium fistulosum*	*Cardiocrinum giganteum*
6	*Allium monanthum*	*Cardiocrinum giganteum var. giganteum*
7	*Allium prattii*	*Cardiocrinum giganteum var. yunnanense*
8	*Allium pskemense*	*Erythronium albidum*
9	*Allium sativum*	*Erythronium americanum*
10	*Allium tuberosum*	*Fritillaria unibracteata*
11	*Amaryllis minuta*	*Fritillaria unibracteata var. longinectarea*
12	*Crinum asiaticum*	*Gagea serotina*
13	*Crinum asiaticum var. japonicum*	*Lilium apertum*
14	*Crinum macowanii*	*Lilium bulbiferum*
15	*Hymenocallis caribaea*	*Lilium bulbiferum subsp. croceum*
16	*Hymenocallis henryae*	*Lilium davidii*
17	*Hymenocallis tubiflora*	*Lilium davidii var. willmottiae*
18	*Lycoris radiata*	*Lilium distichum*
19	*Narcissus poeticus*	*Lilium fargesii*
20	*Narcissus poeticus var. plenus*	*Lilium lancifolium*
21	*Pancratium arabicum*	*Lilium longiflorum*
22	*Zephyranthes candida*	*Lilium longiflorum var. scabrum*
23	*Zephyranthes simpsonii*	*Lilium pardalinum*
24		*Lilium pardalinum subsp. pardalinum*
25		*Lloydia oxycarpa*
26		*Medeola virginiana*
27		*Tricyrtis macropoda*
28		*Tulipa gesneriana*

B. *Computational Model*

The computational model used in this study is depicted in Figure 3. This study uses the R programming language R version 4.2.1, which is run on a computer with an eight-core CPU using an Intel Core i5-1135G7 processor with a frequency of 2.4 GHz, RAM with a capacity of 16GB and 512GB Solid-State Disk (SSD). Several stages use the package libraries available in the public repository CRAN and Bioconductor. However, preparatory steps are still being taken to use the package according to research needs. Furthermore, each stage in the computational model of this research will be explained as follows.

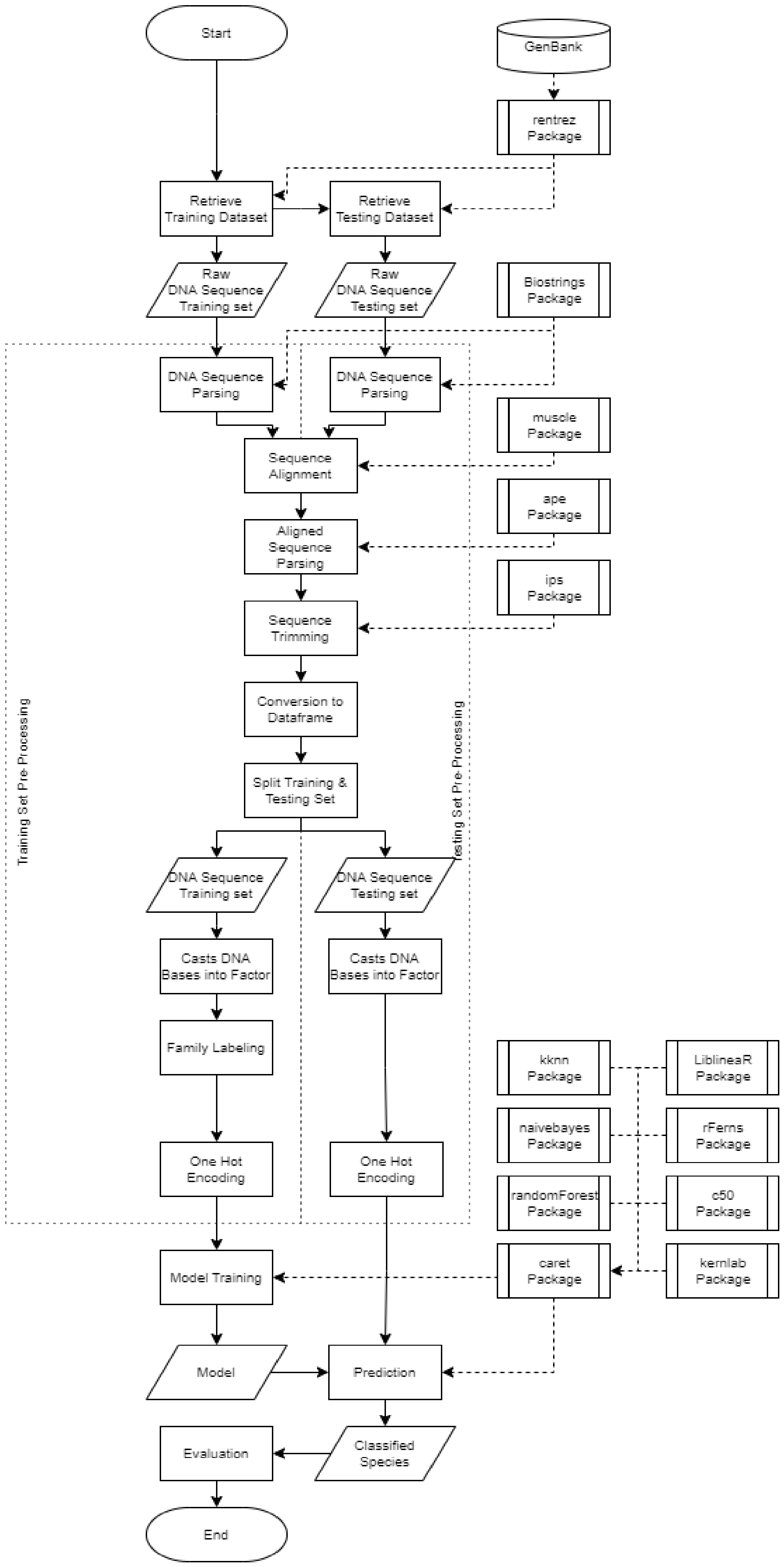


Fig. 3. Computational model of comparison of machine learning algorithms for species family classification using DNA barcode

The first is to retrieve the training/testing dataset. All data are downloaded using the program code with the help of the rentrez package [53]. First, a filter query was made to search for DNA sequences that matched the following criteria: (1) members of the Amaryllidaceae and Liliaceae families, (2) more than 450 bp and less than 10,000 bp in length, (3) excluding species excluded from training data or only species selected for data testing, and (4) is the rbcL gene. The search results are used to download the whole sequence in FASTA format. A series of pre-processed data stages are carried out to use DNA sequences in the classification model. The pre-processing stage starts from the DNA Sequence Parsing stage to Family Labeling.

The second is DNA Sequence Parsing. At this stage, sequences in FASTA format are converted to the DNAStringSet format with the help of the Biostrings package [54]. The results of the sequence conversion in this stage are exemplified in Figure 4.

```
Sequence in FASTA:
>Sequence 1
AAGTG

>Sequence 2
ATAGT

>Sequence 3
CAAGT
```

```
Sequence in DNAStringSet data type:
DNAStringSet object of length 5:
    width  seq    names
[1]     5  AAGTG  Sequence 1
[2]     5  ATAGT  Sequence 2
[3]     5  CAAGT  Sequence 3
```

Fig. 4. Conversion of DNA sequences from FASTA format to the DNAStringSet data type

The third is sequence alignment. Datasets are combined and processed so that the symbols in the sequences are arranged between each sequence to have the same length. Sequence Alignment is run using the Multiple Sequence Alignment (MUSCLE) algorithm with the help of the muscle package [55].

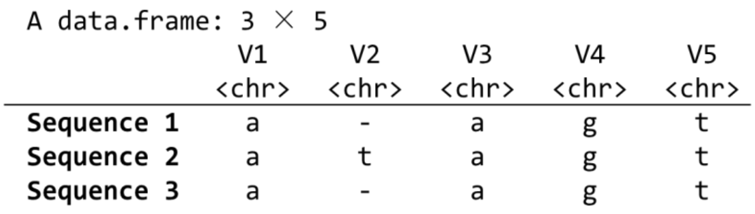
Fourth, aligned sequence parsing. The sequence alignment results are then converted to DNAbin format with the help of the ape (Analyses of Phylogenetics and Evolution) package [56] so that it can be read by the package used in the next stage.

Fifth is sequence trimming. The next step is to perform Sequence Trimming on the existing sequences so that there are no gap symbols in each sequence's upstream (left end) and downstream (right end). The sequences were trimmed with the help of the IPS (Interfaces to Phylogenetic Software) package [57] until 99% of the sequences had no gaps upstream downstream. Figure 5 shows an example of DNA sequence data before and after sequence alignment.

```
[1] AAGTG
[2] ATAGT     Initial DNA Sequence
[3] CAAGT
     ↓
[1] -A-AGTG
[2] -ATAGT-   Result of Sequence Alignment
[3] CA-AGT-
     ↓
[1] A-AGT
[2] ATAGT     Result of Sequence Trimming
[3] A-AGT
```

Fig. 5. DNA sequences before and after alignment and trimming

Sixth, conversion to the data frame. Furthermore, the sequence conversion from the Sequence Trimming results is carried out into the data frame structure. It is the fundamental format commonly used in the R programming language. Each symbol in the sequence is converted to a column with the character data type (character; chr). The DNA representation in the data frame is shown in Figure 6.



```
A data.frame: 3 × 5
                V1     V2     V3     V4     V5
              <chr>  <chr>  <chr>  <chr>  <chr>
Sequence 1      a      -      a      g      t
Sequence 2      a      t      a      g      t
Sequence 3      a      -      a      g      t
```

Fig. 6. DNA sequences in the data frame

Seventh is a split training and testing set. The data in the data frame are then separated back into the data frame for training and testing. All species sequences whose species names are listed in Table 3 are separated into a new data frame as a testing data frame.

Eight casts DNA bases into factor. Each column containing the DNA sequence symbol in the data frame is then cast to an unordered factor data type with five levels. Each of these levels represents a gap symbol and a nucleobase in the DNA sequence, "-", "a", "c", "g", and "t". Gaps replace other nucleobase symbols that have ambiguous properties.

Ninth is family labeling. The training data frame is then added to a new column filled with family labels according to the data from GenBank, while the data frame testing added a new column for the family but with empty data.

The next is one-hot encoding. The data is transformed into a numeric representation in this stage, facilitating its subsequent processing. Precisely, each character that represents nucleobases-namely "a", "c", "g", "t", or "-" derived from the alignment process, is mapped to a five-column matrix. Within this matrix, the column corresponding to the specific nucleobase character is assigned a value of 1, while the remaining columns are assigned a value of 0, as illustrated in Figure 7 [58].

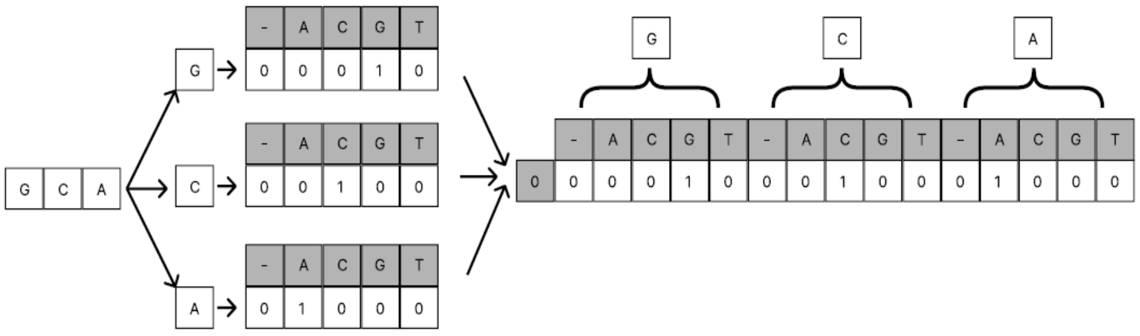

G C A
G → - A C G T / 0 0 0 1 0
C → - A C G T / 0 0 1 0 0
A → - A C G T / 0 1 0 0 0
G C A
- A C G T - A C G T - A C G T
0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0

Fig. 7. One hot encoding process

After that, models training. At this stage, a prediction model is made based on the training data frame that has been prepared. Packages used to build classification models are C5.0, kknn, LiblineaR, naivebayes, rFerns, randomForest, kernlab, and caret. At this stage, experiments were carried out on the dataset and the parameters of the Random Ferns algorithm, the number of ferns, and the depth. A validation process is also carried out to ensure the model is not overfitting or underfitting through a cross-validation process with the help of a caret package [59]. Parallel [60] and doParallel [61] packages speed up the cross-validation resampling. The foreach package [62] also turns off parallel compute mode. The model used in this experiment including: C5.0, Knn (k-Nearest Neighbors), lssvmPoly (Least Squares Support Vector Machine with a polynomial kernel), naive_bayes, regLogistic (Regularized Logistic Regression), rf: (Random Forest), rFerns (Random Ferns), svmLinear (Support Vector Machines with Linear Kernel), svmPoly (Support Vector Machines with

Polynomial Kernel), svmRadial (Support Vector Machines with Radial Basis Function Kernel), and svmRadialWeights (Support Vector Machines with Class Weights).

The next one is prediction. Class prediction is carried out on the data frame testing based on the model made in the previous stage.

The last is evaluation. The prediction results of the classification models are then evaluated based on the level of accuracy concerning the family label of each sequence in GenBank and the results of the sequence consensus made using the DECIPHER package [63]. Duration and memory used when training the model are measured using the profvis package [64].

III. Results and Discussions

This study used rbcL gene sequence data from species in the Amaryllidaceae and Liliaceae families obtained from GenBank. Each species in the dataset has more than one sample because each sequence comes from sequencing results in different locations. All dataset downloads are performed using program code. For example, downloading the Amaryllis training dataset starts by searching GenBank using the entrez_search function from the rentrez package in the following program code. The argument for the term parameter is a variable that contains the search query.

```
search_result <- rentrez::entrez_search(
  db = "nuccore",
  term = query,
  retmax = limit
  use_history = TRUE
)
```

The search results are then used to download the sequences using the entrez_fetch function from the rentrez package. Iterations are carried out with 50 steps until the DNA sequences from the search results are entirely downloaded.

```
Initialize sequences as an empty string
Set chunk_size to 50

Calculate num_iterations based on the total number of ids in search_result
divided by chunk_size, rounded up

For each iteration i from 1 to num_iterations:
    Calculate start_idx based on the current iteration and chunk_size
    Calculate end_idx as the smaller value between i times chunk_size and the
total number of ids in search_result

    Extract a subset of ids from search_result between start_idx and end_idx
and store in current_ids

    Use current_ids to fetch sequence data from nuccore database and store the
result in fetchRes

    If fetchRes is not found in sequences:
        Append fetchRes to sequences
```

The downloaded result is then exported into a file using the write function. Detailed information about the datasets successfully downloaded and used in this study has been presented in the Datasets section.

```
write(amaryllis_train, file = "amaryllis_train.fasta")
```

After all datasets have been downloaded, the next step is to carry out a series of data and experiments in the pre-processing stages. The configuration of this experimental scenario is shown in Table 4. All the data used went through the pre-processing stage to the exact sequence alignment. After that step, the difference is started by setting the sequence trimming threshold, which will affect the length of the resulting sequence after the sequence trimming stage. A resampling method was used for each configuration combination using ten iterations of 10-fold cross-validation.

Table 4. Experimental scenario and results

Scenario	Result
Sequence Trimming Threshold	99%
Amaryllis training samples	689
Lily training samples	713
Total sample training used	1402

One of the steps in the pre-processing stage of the data in this study is to perform sequence alignment. The MUSCLE algorithm is used with the help of the muscle library package. The DNA sequence data from the training and testing datasets combined and converted to the DNAStringSet data type are given as arguments to the muscle function. After sequence alignment, each sequence has a length of 3050bp and is stored in DNAMultipleAlignment format.

```
aligned_dataset <- muscle::muscle(parsed_dataset)
```

The aligned sequences are then trimmed using the trimEnds function from the IPS package. At this stage, you can set a minimum threshold for the number of columns that do not have gaps in the sequence_trimming_threshold variable. The DNA sequence converted to the DNAbin data type is given as the first argument of this function. After going through the sequence trimming process with a 99% threshold configuration, the sequence has a length of 1192 bp.

```
arsed_aligned_dataset <- ape::as.DNAbin(aligned_dataset)
trimmed_dataset <- ips::trimEnds(parsed_aligned_dataset, trim_at_least)
```

The DNA sequences are then converted to data frame data types and separated again into training and testing sets. The separation is done based on each dataset's sequence labels (row names) before being combined. After splitting, the test data frame contains 220 sequence lines. The training data frame contains 1,402 sequence rows. It can be noted that the threshold set in the sequence trimming process affects the resulting sequence data. The larger the set threshold, the shorter the sequence will be and cause the sequence data between one specimen and another to have the same content.

Furthermore, the conversion of base symbols in the data sequences in the data frame is carried out into a factor data type. The factor function encodes the vector data type to the factor data type and is used as the second argument of the lapply function. The following argument in the lapply function is the argument to the function specified in the second parameter. It is specified that five levels represent gaps and nucleotide base symbols in anonymous vectors for the levels parameter. After the conversion, the nucleotide base symbol other than the four symbols will be changed to NA and replaced with a gap symbol.

```
For each column in train_df:
    Convert the column to a factor with levels "-", "a", "c", "g", "t"
    Replace the original column with the converted column
```

In the training data frame, a new column is added for the family label based on the information obtained from GenBank. Labeling is done based on data sequence labels (row names) in each training dataset (Amaryllis and Lily) before being combined. After this step, the data is transformed into a numeric representation using one-hot encoding.

```
Function oneHotDNA(df, n = ["A", "C", "G", "T", "-"]):
    Convert all elements in df to uppercase

    Initialize seq_col as the number of columns in df
    Initialize seq_row as the number of rows in df

    Create an empty matrix seq_mat with dimensions (seq_row, seq_col *
length(n))
    Initialize an empty list column_names

    For each column i from 1 to seq_col:
        For each row j from 1 to seq_row:
            If the element at (j, i) in df is in n:
                Find the position of the element in n
                Set the corresponding position in seq_mat to 1

        For each element j in n:
            Append (j + "-" + i) to column_names

    Set the column names of seq_mat to column_names

    Return seq_mat
```

Furthermore, the classification models and the parameters obtained from the random search process are made. The first argument is the formula for the attribute, and the following is the data source used. The third argument is a model that is being used, and this is a model that has been made previously with the parameters obtained from the random search process. The last two arguments are the configurations for Caret's train function and the maximum number of tuning parameter combinations that will be generated. The resulting model is then stored in the model variable. The tilde operator (~) is used to define the model formulae. The left side of the operator is interpreted as the result data of the function, and the right side is the function input. In this case, the formulae Family ~. means that the Family column is modeled as a function of all data other than the Family column.

```
model <- train_profile(
    Family ~ .,
    data = data,
    method = custom_model,
    trControl = train_configuration,
    tuneLength = tune_length
)
```

Classification models can be created directly using the default parameters. However, it cannot be known whether the model is the best as it was created once. A resampling step with the cross-validation method was carried out using a caret package to verify the performance of the classification model. In this study, a 10-fold cross-validation method was used with ten repetitions. Caret supports creating classification models and cross-validation from other package containing machine learning models such as C5.0, kknn, LiblineaR, naivebayes, rFerns, randomForest, and kernlab. Below is an example of creating a random fern model.

```
custom_model <- caret::getModelInfo("rFerns")$rFerns

custom_model$parameters <- data.frame(
  parameter = c("depth", "ferns"),
  class = rep("numeric", 2),
  label = c("Fern Depth", "Number of Ferns")
)

custom_model$grid <- function(x, y, len = NULL, search = "grid") {
  if (search == "grid") {
    out <- expand.grid(
      depth = unique(floor(seq(1, 16, length = len))),
      ferns = floor((1:len) * 100)
    )
  } else {
    out <- data.frame(
      depth = sample(1:16, size = len, replace = TRUE),
      ferns = sample(1:1000, replace = TRUE, size = len)
    )
  }
  out
}

custom_model$fit <- function(x, y, wts, param, lev, last, classProbs, ...) {
  if (!is.data.frame(x) | inherits(x, "tbl_df")) {
    x <- as.data.frame(x, stringsAsFactors = TRUE)
  }
  rFerns::rFerns(x, y, depth = param$depth, ferns = param$ferns, ...)
}
```

After the model definition is completed, a cross-validation configuration is prepared using the trainControl function.

```
caret::trainControl(
  method = "cv",
  number = 10,
  search = "random",
  timingSamps = 1,
  verboseIter = TRUE
)
```

In practice, this resampling process takes a relatively long time. The resampling process is configured to be done in parallel by utilizing all CPU cores available to speed it up. By default, caret has configured the resampling process to run in parallel. However, configuring and initializing socket clusters is still necessary to perform parallel processing. The makePSOCKcluster function from the parallel package is used with the parameter number of cores to be used, in this case, using eight cores. It is then configured to run operations in parallel using the registerDoParallel function from the doParallel package. Validation of socket cluster creation can be done with the showConnections function.

```
# Enable Parallel Processing
cl <- parallel::makePSOCKcluster(8)
doParallel::registerDoParallel(cl)
# Check
showConnections()
# Disable Parallel Processing
parallel::stopCluster(cl)
foreach::registerDoSEQ()
# Check
showConnections()
```

The outcomes of the classification algorithms are delineated in Table 5, where optimal hyperparameters were ascertained through a 10-fold cross-validation technique. Notably, the C5.0, Regularized Logistic Regression, Random Forest, SVM linear, SVM poly, SVM radial, and SVM radial weights yielded exceptional classification accuracy at 99.85%. In contrast, the Naïve Bayes algorithm emerged as the least efficacious model, registering a mere 53.2% accuracy rate. A graphical representation of the accuracy metrics accrued during the training phase is furnished in Figure 8.

From Figure 8, we can see that the model's accuracy is relatively high, except for Naïve Bayes. The rest of the model obtained more than 97% accuracy. Whereas for the computing power, we monitor memory usage and the computational cost for each model. The result can be seen in Table 6.

In Table 6, we present a comparative analysis of computational efficiency, quantified in terms of time complexity and memory utilization across various machine learning models. The Random Ferns algorithm exhibits superior computational performance, requiring a mere 24.67 seconds for execution. Naïve Bayes, SVM follows this with a linear kernel, and Regularized Logistic Regression, which necessitate computational durations of 41.14 seconds, 45 seconds, and 77.76 seconds, respectively. Conversely, the Random Forest algorithm demonstrates the least computational efficiency, consuming a substantial 1 hour, 12 minutes, and 51.75 seconds for its computational tasks.

Table 5. Best parameters obtained from the training process, along with accuracy

Algorithm	Best Parameters	Accuracy
C5.0	– trials:51 – model:2 – winnow: FALSE	0.9985
knn	k:27	0.974320884
lssvmPoly	– degree:3 – scale:0.0438458726217134 – tau:27.3810618566797	0.995704
naive_bayes	– laplace:0 – use kernel: FALSE – adjust:1	0.532098
regLogistic	– cost:5.98885780307469 – loss:L2_dual – epsilon:1	0.998571356
rf	mtry:270	0.998571356
rFerns	– depth:14 – ferns:330	0.996428499
svmLinear	C:0.0594077128418059	0.998571356
svmPoly	– degree:1 – scale:0.00434829110517236 – C:1.73112678364247	0.998571356
svmRadial	– sigma:0.000353579812294785 – C:8.56785410178861	0.998571356
svmRadialWeights	– sigma:0.000353579812294785 – C:8.56785410178861 – Weight:12.9745684396476	0.998571

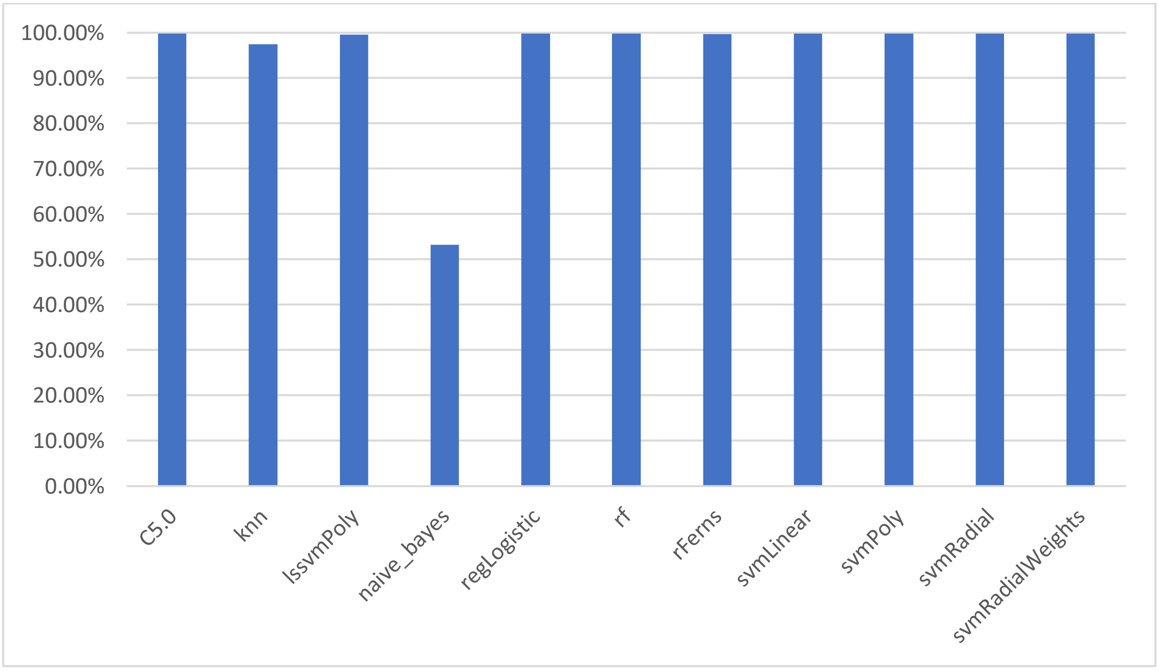


Fig. 8. Accuracy obtained from training process

In the context of memory utilization, the Naïve Bayes algorithm demonstrates the most efficient memory footprint, consuming a mere 863.9 megabytes (Mb), but in return, this model performs poorly in the experiment. This result is followed by k-Nearest Neighbors, Random Forest, Random Ferns, and Support Vector Machines with a linear kernel, which exhibits memory usage within the 1450 to 1650 Mb range. Conversely, the Least Squares Support Vector Machine (LSSVM) employing a polynomial kernel and Regularized Logistic Regression algorithms manifest significantly elevated memory consumption, requiring 6511.4 Mb and 4419.8 Mb, respectively. The visual representations of these memory utilization metrics are provided in Figure 9 and Figure 10.

Table 6. Computational time and memory used in the training process

Algorithm	Time (s)	Memory (Mb)
C5.0	326.17	2968.5
knn	308.69	1481.4
lssvmPoly	264.91	6511.4
naive_bayes	41.14	863.9
regLogistic	77.76	4419.8
rf	4371.75	1646.8
rFerns	24.67	1582.2
svmLinear	45	1599.9
svmPoly	88.83	2385.5
svmRadial	122.25	2763.1
svmRadialWeights	103.13	2959.2

After the classification model, predictions are made on the testing data using the model. The predict function is used with the training models as the first argument and the testing data frame as the second. The output of the prediction function is saved to the prediction variable.

```
prediction$results <- predict(model, PredictData)
```

Figure 11 illustrates the outcomes of various algorithms applied in predicting the family of species within the test dataset. This evaluation was conducted by executing each Machine Learning model to predict classifications of the ambiguous data, followed by a comparison with the labels provided by NCBI. A prediction was deemed accurate if it aligned with the NCBI labels. Remarkably, the consistency between these results and the accuracy observed during the training phase suggests the reliability of NCBI's classification of the contentious data.

For species in the Amaryllidaceae family (according to NCBI), almost all algorithms consistently predict Amaryllidaceae, aligning perfectly with the NCBI consensus. This suggests that these algorithms are agreed for classifying species into the Amaryllidaceae family. There are a few anomalies, such as *Lilium bulbiferum subsp. croceum,* where most algorithms predict Amaryllidaceae instead of Liliaceae, diverging from the NCBI consensus and most algorithmic predictions.

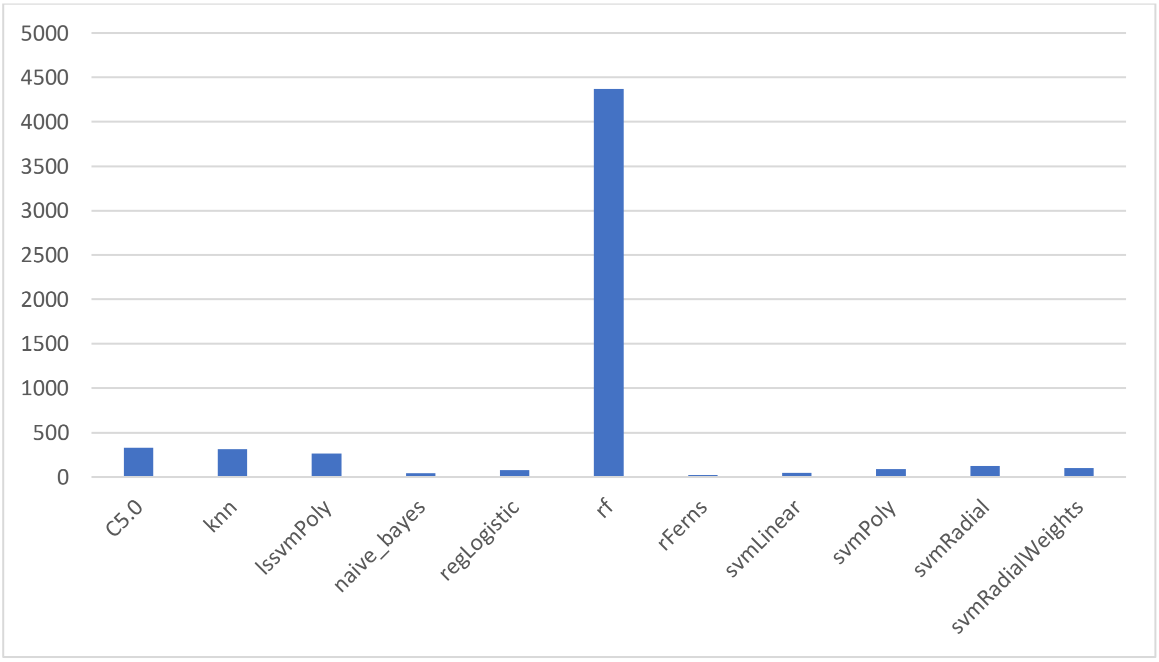


Fig. 9. Computational time for training process (s)

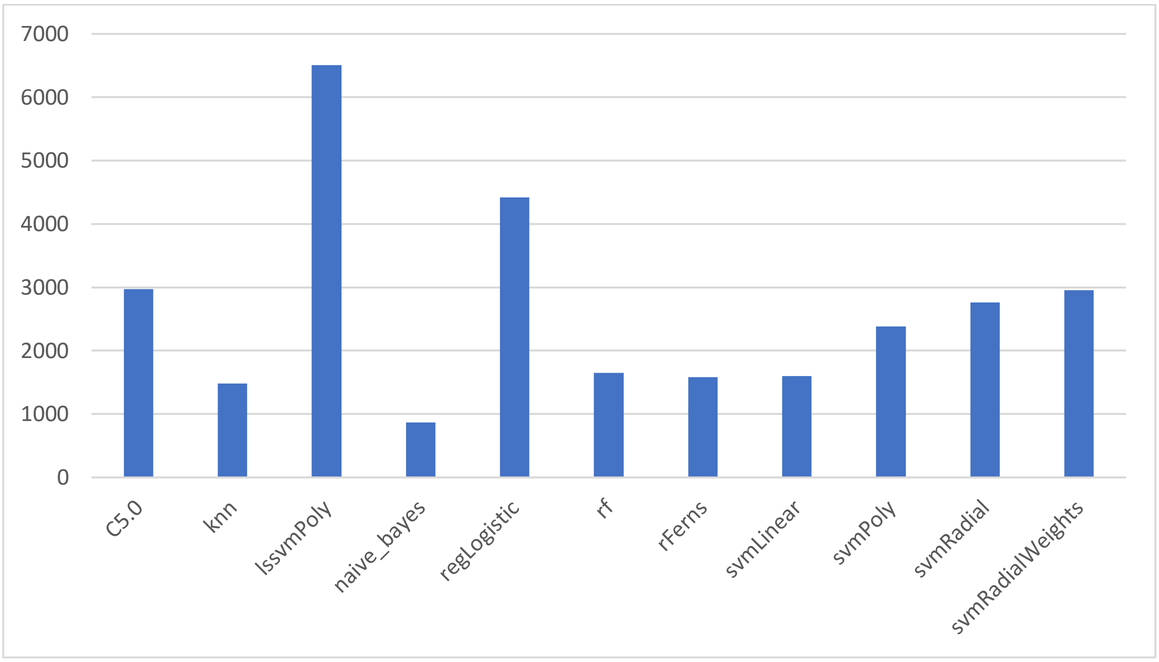


Fig. 10. Memory used for training process (Mb)

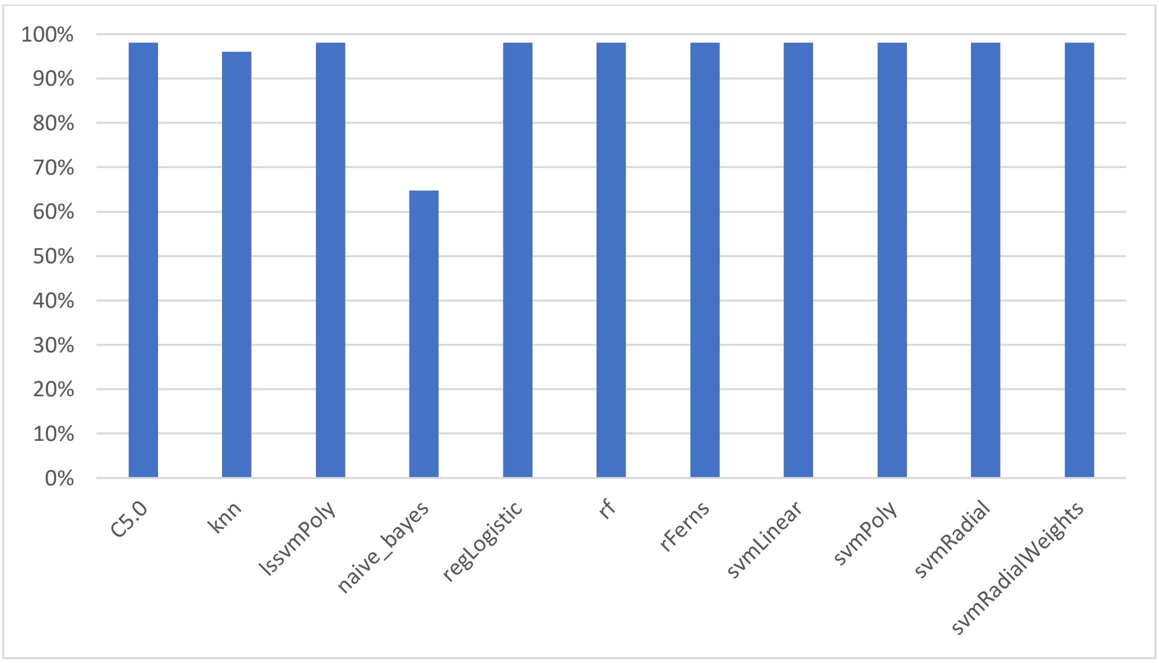


Fig. 11. How accurate model predicts the disputed data

However, there is some variability in the predictions. For example, the algorithm LSSVM poly and naïve bayes occasionally predict species in the Amaryllidaceae family as belonging to the Liliaceae family. This indicates that while these algorithms are generally accurate, there may be specific cases where they diverge from the consensus. The algorithms agree with the NCBI consensus regarding the Liliaceae family, predicting Liliaceae for all species listed under this family.

The machine learning algorithms largely agree with the NCBI consensus, demonstrating their effectiveness in classifying species into the Amaryllidaceae and Liliaceae families. Almost all machine learning algorithms gain an accuracy of 98%, except knn with 96% and naïve bayes with 65% accuracy. However, the algorithms diverge a few instances, suggesting areas for further research and model refinement. Overall, the table is a valuable resource for evaluating the performance and reliability of various machine-learning algorithms in plant taxonomy.

IV. Conclusions

Almost all of the models compared in this study were able to classify the DNA Barcode data using the rbcL gene with reasonable accuracy with more than 97% accuracy, except for the Naïve Bayes model with just 53% accuracy. From the results of the resampling process using ten iterations of 10-fold cross-validation, we get that the most accurate model, namely C5.0, Regularized Logistic Regression, Random Forest, SVM linear, SVM poly, SVM radial, and SVM radial weights yielded exceptional classification accuracy at 99.85%. Regarding computational time, the most exhaustive model is Random Forest and the least exhaustive model is Random Ferns, which only uses 24.67 seconds of computing time. In terms of memory used by the model, the LSSVM that uses a polynomial kernel model and Regularized Logistic Regression gain the highest memory usage at 6511.4 Mb and 4419.8 Mb, respectively. In contrast, Naïve Bayes gets the least computing power, but the model's accuracy is less significant than other models. While predicting the family using a test dataset, the machine learning models align highly with the NCBI's classifications, effectively categorizing species into the Amaryllidaceae and Liliaceae families. Nonetheless, some discrepancies exist, indicating the need for additional research and model improvement.

Acknowledgment

The authors would like to acknowledge the Ministry of Research and Technology, Research and Community Services Institutions of Universitas Pendidikan Indonesia on Indonesia Collaboration Research (RKI) for funding this work through the research grant of 1261/UN40.LP/PT.01.03/2022.

Declarations

Author contribution

All authors contributed equally as the main contributor of this paper. All authors read and approved the final paper.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Conflict of interest

The authors declare no known conflict of financial interest or personal relationships that could have appeared to influence the work reported in this paper.

Additional information

Reprints and permission information are available at http://journal2.um.ac.id/index.php/keds.

References

[1] A. Yang, W. Zhang, J. Wang, K. Yang, Y. Han, and L. Zhang, "Review on the Application of Machine Learning Algorithms in the Sequence Data Mining of DNA," Front. Bioeng. Biotechnol., vol. 8, p. 1032, Sep. 2020.

[2] S. Behjati and P. S. Tarpey, "What is next generation sequencing?," Arch. Dis. Child. - Educ. Pract. Ed., vol. 98, no. 6, Art. no. 6, Dec. 2013.

[3] J. Dabney et al., "Complete mitochondrial genome sequence of a Middle Pleistocene cave bear reconstructed from ultrashort DNA fragments," Proc. Natl. Acad. Sci., vol. 110, no. 39, Art. no. 39, Sep. 2013.

[4] L. Riza, M. Nurfathiya, J. Kusnendar, and K. Abu Samah, "DNA barcoding using particle swarm optimization on apache spark SQL case study: DNA of covid-19," Int. J. Nonlinear Anal. Appl., vol. 12, no. Special Issue, Art. no. Special Issue, Jan. 2021.

[5] P. D. N. Hebert, A. Cywinska, S. L. Ball, and J. R. deWaard, "Biological identifications through DNA barcodes," Proc. R. Soc. Lond. B Biol. Sci., vol. 270, no. 1512, pp. 313–321, Feb. 2003.

[6] C. Manwell and C. M. A. Baker, "A sibling species of sea cucumber discovered by starch gel electrophoresis," Comp. Biochem. Physiol., vol. 10, no. 1, Art. no. 1, Sep. 1963.

[7] P. D. N. Hebert, S. Ratnasingham, and J. R. De Waard, "Barcoding animal life: cytochrome c oxidase subunit 1 divergences among closely related species," Proc. R. Soc. Lond. B Biol. Sci., vol. 270, no. suppl_1, Aug. 2003.

[8] CBOL Plant Working Group1 et al., "A DNA barcode for land plants," Proc. Natl. Acad. Sci., vol. 106, no. 31, Art. no. 31, Aug. 2009.

[9] C. L. Schoch et al., "Nuclear ribosomal internal transcribed spacer (ITS) region as a universal DNA barcode marker for Fungi," Proc. Natl. Acad. Sci., vol. 109, no. 16, pp. 6241–6246, Apr. 2012.

[10] C.-H. Yang, K.-C. Wu, L.-Y. Chuang, and H.-W. Chang, "DeepBarcoding: Deep Learning for Species Classification Using DNA Barcoding," IEEE/ACM Trans. Comput. Biol. Bioinform., vol. 19, no. 4, pp. 2158–2165, Jul. 2022.

[11] J. Yang et al., "Development of Chloroplast and Nuclear DNA Markers for Chinese Oaks (Quercus Subgenus Quercus) and Assessment of Their Utility as DNA Barcodes," Front. Plant Sci., vol. 8, p. 816, May 2017.

[12] M. Emu and S. Sakib, "Species Identification using DNA Barcode Sequences through Supervised Learning Methods," in 2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), Cox'sBazar, Bangladesh: IEEE, Feb. 2019, pp. 1–6.

[13] T. He, L. Jiao, A. C. Wiedenhoeft, and Y. Yin, "Machine learning approaches outperform distance- and tree-based methods for DNA barcoding of Pterocarpus wood," Planta, vol. 249, no. 5, Art. no. 5, May 2019.

[14] L. Jin, J. Yu, X. Yuan, and X. Du, "Fish Classification Using DNA Barcode Sequences through Deep Learning Method," Symmetry, vol. 13, no. 9, Art. no. 9, Aug. 2021.

[15] P. K. Meher, T. K. Sahu, and A. R. Rao, "Identification of species based on DNA barcode using k-mer feature vector and Random forest classifier," Gene, vol. 592, no. 2, pp. 316–324, Nov. 2016.

[16] E. Weitschek, G. Fiscon, and G. Felici, "Supervised DNA Barcodes species classification: analysis, comparisons and results," BioData Min., vol. 7, no. 1, p. 4, Dec. 2014.

[17] D. Sobolewska, A. Galanty, K. Grabowska, J. Makowska-Wąs, D. Wróbel-Biedrawa, and I. Podolak, "Saponins as cytotoxic agents: an update (2010–2018). Part I—steroidal saponins," Phytochem. Rev., vol. 19, no. 1, pp. 139–189, Feb. 2020.

[18] P. Nagare and S. S. Shekokar, "A Literature Review Of Some Important Pharmacological Activities Of Few Plants Of Liliaceae Family," 2022.

[19] P. F. Stevens, "Angiosperm Phylogeny Website. Version 13.," Angiosperm Phylogeny Website Version 13, 2016.

[20] A. M. Takos and F. Rook, "Towards a molecular understanding of the biosynthesis of Amaryllidaceae alkaloids in support of their expanding medical use," Int. J. Mol. Sci., vol. 14, no. 6, pp. 11713–11741, 2013.

[21] L. Torras Claveria, L. R. Tallini, F. Viladomat Meya, and J. Bastida Armengol, "Research in natural products: Amaryllidaceae ornamental plants as sources of bioactive compounds," Recent Adv. Pharm. Sci. VII 2017 Res. Signpost Ed. Diego Muñoz-Torrero Montserrat Riu Carles Feliu ISBN 978-81-308-0573-3 Chapter 5 P 69-82, 2017.

[22] M. W. Chase, J. L. Reveal, and M. F. Fay, "A subfamilial classification for the expanded asparagalean families Amaryllidaceae, Asparagaceae and Xanthorrhoeaceae," Bot. J. Linn. Soc., vol. 161, no. 2, pp. 132–136, 2009.

[23] A. Kornienko and A. Evidente, "Chemistry, biology, and medicinal potential of narciclasine and its congeners," Chem. Rev., vol. 108, no. 6, pp. 1982–2014, 2008.

[24] R. M. Dahlgren and H. T. Clifford, The monocotyledons: a comparative study. Academic Press, 1982.

[25] G. Bentham and J. D. Hooker, Genera plantarum :ad exemplaria imprimis in Herberiis Kewensibus servata definita /auctoribus G. Bentham et J.D. Hooker. London, England: A. Black, 1862.

[26] A. Engler, K. Krause, R. Pilger, and K. Prantl, Die Natürlichen Pflanzenfamilien nebst ihren Gattungen und wichtigeren Arten, insbesondere den Nutzpflanzen, unter Mitwirkung zahlreicher hervorragender Fachgelehrten begründet. Leipzig: W. Engelmann, 1887.

[27] C. E. Bessey, "The Phylogenetic Taxonomy of Flowering Plants," Ann. Mo. Bot. Gard., vol. 2, no. 1/2, p. 109, Feb. 1915.

[28] A. B. Rendle, The classification of flowering plants, no. Vol. 2. Cambridge: Cambridge Univ. Press, 1925.

[29] J. Hutchinson, "Families of Flowering Plants. II. Monocotyledons," Oxf. Univ. Press, p. 243, 1934.

[30] A. Cronquist, An integrated system of classification of flowering plants. New York: Columbia University Press, 1981.

[31] A. L. Takhtajan, "Outline of the classification of flowering plants (magnoliophyta)," Bot. Rev., vol. 46, no. 3, pp. 225–359, Jul. 1980.

[32] H. Clifford, R. Dahlgren, and P. Yeo, The families of the monocotyledons: structure, evolution, and taxonomy. Springer, 1985.

[33] Y. Mimaki and Y. Sashida, "Steroidal Saponins from the Liliaceae Plants and Their Biological Activities," in Saponins Used in Traditional and Modern Medicine, G. R. Waller and K. Yamasaki, Eds., in Advances in Experimental Medicine and Biology, vol. 404. Boston, MA: Springer US, 1996, pp. 101–110.

[34] P. Korall and P. Kenrick, "Phylogenetic relationships in Selaginellaceae based on rbcL sequences," Am. J. Bot., vol. 89, no. 3, pp. 506–517, 2002.

[35] J. E. Richardson, M. F. Fay, Q. C. Cronk, D. Bowman, and M. W. Chase, "A phylogenetic analysis of Rhamnaceae using rbcL and trnL-F plastid DNA sequences," Am. J. Bot., vol. 87, no. 9, pp. 1309–1324, 2000.

[36] T. M. Mitchell, Machine Learning. in McGraw-Hill series in computer science. New York: McGraw-Hill, 1997.

[37] A. C. Müller and S. Guido, Introduction to machine learning with Python: a guide for data scientists, First edition. Sebastopol, CA: O'Reilly Media, Inc, 2016.

[38] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Comput., vol. 7, no. 1, Art. no. 1, Jan. 2003.

[39] K. Jacobson, V. Murali, E. Newett, B. Whitman, and R. Yon, "Music Personalization at Spotify," in Proceedings of the 10th ACM Conference on Recommender Systems, Boston Massachusetts USA: ACM, Sep. 2016, pp. 373–373.

[40] L. S. Riza, A. D. Pertiwi, E. F. Rahman, M. Munir, and C. U. Abdullah, "Question Generator System of Sentence Completion in TOEFL Using NLP and K-Nearest Neighbor," Indones. J. Sci. Technol., vol. 4, no. 2, Art. no. 2, Sep. 2019.

[41] L. S. Riza, F. S. Anwar, E. F. Rahman, C. U. Abdullah, and S. Nazir, "Natural Language Processing and Levenshtein Distance for Generating Error Identification Typed Questions on TOEFL," J. Comput. Soc., vol. 1, no. 1, Art. no. 1, Jun. 2020.

[42] L. S. Riza, R. A. Rosdiyana, A. R. Pérez, and A. Wahyudin, “The K-Means Algorithm for Generating Sets of Items in Educational Assessment,” Indones. J. Sci. Technol., vol. 6, no. 1, Art. no. 1, Jan. 2021.
[43] P. Larrañaga et al., “Machine learning in bioinformatics,” Brief. Bioinform., vol. 7, no. 1, Art. no. 1, Mar. 2006.
[44] L. S. Riza, F. D. Pratama, E. Piantari, and M. Fahsi, “Genomic repeats detection using Boyer-Moore algorithm on Apache Spark Streaming,” TELKOMNIKA Telecommun. Comput. Electron. Control, vol. 18, no. 2, Art. no. 2, Apr. 2020.
[45] L. S. Riza, A. B. Rachmat, Munir, T. Hidayat, and S. Nazir, “Genomic repeat detection using the Knuth-Morris-Pratt algorithm on R high-performance-computing package,” Int J Adv. Soft Compu Appl, vol. 11, no. 1, Art. no. 1, Mar. 2019.
[46] S. J. Russell, P. Norvig, and E. Davis, Artificial intelligence: a modern approach, 3rd ed. in Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010.
[47] S. Salzberg, “Locating Protein Coding Regions in Human DNA Using a Decision Tree Algorithm,” J. Comput. Biol., vol. 2, no. 3, Art. no. 3, Jan. 1995.
[48] C. Cortes and V. Vapnik, “Support-vector networks,” Mach. Learn., vol. 20, no. 3, Art. no. 3, Sep. 1995.
[49] L. Breiman, “Random Forests,” Mach. Learn., vol. 45, no. 1, Art. no. 1, 2001.
[50] L. Bao and Y. Cui, “Prediction of the phenotypic effects of non-synonymous single nucleotide polymorphisms using structural and evolutionary information,” Bioinformatics, vol. 21, no. 10, Art. no. 10, May 2005.
[51] A. Gupta, H. Wang, and M. Ganapathiraju, “Learning structure in gene expression data using deep architectures, with an application to gene clustering,” in 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Washington, DC, USA: IEEE, Nov. 2015, pp. 1328–1335.
[52] D. A. Benson et al., “GenBank,” Nucleic Acids Res., vol. 41, no. D1, pp. D36–D42, Nov. 2012.
[53] D. Winter J., “rentrez: An R package for the NCBI eUtils API,” R J., vol. 9, no. 2, p. 520, 2017.
[54] H. Pagès et al., “Biostrings: Efficient manipulation of biological strings.” Bioconductor version: Release (3.17), 2023.
[55] R. C. Edgar, “MUSCLE: multiple sequence alignment with high accuracy and high throughput,” Nucleic Acids Res., vol. 32, no. 5, Art. no. 5, Mar. 2004.
[56] E. Paradis and K. Schliep, “ape 5.0: an environment for modern phylogenetics and evolutionary analyses in R,” Bioinformatics, vol. 35, no. 3, Art. no. 3, Feb. 2019.
[57] C. Heibl, “PHYLOCH: R language tree plotting tools and interfaces to diverse phylogenetic software packages.” Jan. 2008.
[58] L. S. Riza, M. I. Zain, A. Izzuddin, Y. Prasetyo, T. Hidayat, and K. A. F. Abu Samah, “Implementation of Machine Learning in DNA Barcoding for Determining the Plant Family Taxonomy,” SSRN Electron. J., 2022.
[59] M. Kuhn et al., “caret: Classification and Regression Training.” Mar. 21, 2023. (Accessed: Jul. 10, 2023)
[60] R Core Team, “R: The R Project for Statistical Computing,” Jul. 10, 2023. (Accessed: Jul. 10, 2023)
[61] F. Daniel, M. Corporation, S. Weston, and D. Tenenbaum, “doParallel: Foreach Parallel Adaptor for the ‘parallel’ Package.” Feb. 07, 2022. (Accessed: Jul. 10, 2023)
[62] F. Daniel, H. Ooi, R. Calaway, Microsoft, and S. Weston, “foreach: Provides Foreach Looping Construct.” Feb. 02, 2022. (Accessed: Jul. 10, 2023)
[63] E. Wright S., “Using DECIPHER v2.0 to Analyze Big Biological Sequence Data in R,” R J., vol. 8, no. 1, Art. no. 1, 2016.
[64] W. Chang et al., “profvis: Interactive Visualizations for Profiling R Code.” May 02, 2023. Accessed: Jul. 10, 2023.