Exploiting the Sequential Nature of Genomic Data for Improved Analysis and Identification

M. Saqib Nawaz[1] · **M. Zohaib Nawaz**[1,2] · **Zhang Junyi**[1] · **Philippe Fournier-Viger**[1,*] · **Jun-Feng Qu**[3]

Date of receiving / date of acceptance

Abstract Genomic data is growing exponentially, posing new challenges for sequence analysis and classification, particularly for managing and understanding harmful new viruses that may later cause pandemics. Recent genome sequence classification models yield promising performance. However, the majority of them do not consider the sequential arrangement of nucleotides and amino acids, a critical aspect for uncovering their inherent structure and function. To overcome this, we introduce GenoAnaCla, a novel approach for analyzing and classifying genome sequences, based on sequential pattern mining (SPM). The proposed approach first constructs and preprocesses datasets comprising RNA virus genome sequences in three formats: *nucleotide*, *coding region*, and *protein*. Then, to capture sequential features for the analysis and classification of viruses, GenoAnaCla extracts frequent sequential patterns and rules in three forms and in codons. Eight classifiers are utilized, and their effectiveness is assessed by employing a variety of evaluation metrics. A performance comparison demonstrates that the suggested approach surpasses the current state-of-the-art genome sequence classification and detection techniques with a 3.18% performance increase in accuracy on average.

Keywords Genomes · Nucleotide bases · Codons · Amino acids · Frequent sequential patterns · Classification

1 Introduction

Thanks to advanced sequencing techniques, genomes can now be sequenced quickly and shared on accessible online databases such as GenBank [1], NGDC [2], and GI-

[1]College of Computer Science and Software Engineering, Shenzhen University, China
[2]Faculty of Computing and Information Technology, Department of Computer Science, University of Sargodha, Pakistan
[3]School of Computer Engineering, Hubei University of Arts and Science, Xiangyang, Hubei, China
E-mail: msaqibnawaz@szu.edu.cn, zohaib.nawaz@uos.edu.pk, zhangjunyi37@126.com, philfv@szu.edu.cn, qmxwt@163.com
* Corresponding author

SAID [3]. But those huge and complex biological datasets surpass the capabilities of traditional methods for processing, analyzing, and comparing genome sequences. The majority of genomic tools utilized to establish taxonomies of living organisms employ alignment-based methods. For instance, BLAST [4] and FASTA [5], along with their improved or extended versions are widely recognized and commonly serve as benchmarks for genomic sequence analysis. Nevertheless, they possess numerous drawbacks, including high computational costs when aligning big datasets, their inadequacy for scenarios with low sequence identity, and the influence of various prior assumptions and parameters on their outcomes [6,7]. Similarly, genome analysis and classification methodologies that utilize *k-mers* or minimizers [6,8] require extensive regions of high similarity, which can compromise precision and recall. Gene-based methods depend on the manual design of features, and selecting the most appropriate features for a specific task can be challenging [9].

The urgent need to manage and analyze massive genomic data arises from the constant emergence of dangerous viruses globally. The third United Nations Sustainable Development Goal[1], "Good Health and Well-being," underscores the importance of research and development in healthcare to combat these emerging diseases and enhance health outcomes. To achieve this, it is crucial to overcome the challenges of managing, storing, and processing genomic data, and to extract meaningful insights from it. Genome sequence analysis and classification play a pivotal role in understanding the genetic foundations of diseases, enabling the development of targeted therapies, personalized medicine, and early detection strategies. To facilitate early action, the availability of vast genomic data offers a unique opportunity to develop efficient prediction models for detecting viral genomes in human DNA [10]. Combining conventional machine learning (ML) and frequent pattern mining techniques can provide an innovative approach to mining this data, leading to the development of robust and explainable classification/prediction systems that can significantly contribute to improving global health outcomes.

Recent computational methods based on ML and deep learning (DL) [9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25] have focused on the classification/detection of genome sequences and metagenomic data. The goal of these studies is to identify significant features, such as *k-mers*-based features [10,11,13,19,21], CpG-based features [14,15,16], representative genomic sequences [20], intrinsic genomic signatures [25], intrinsic dinucleotide genomic signatures [17], discrete Fourier and Cosine transforms and moment invariants-based features [22], recoding system-based features [11], softmax and pooling-based features [9,12,18,20,26] and biomarkers [23]. Furthermore, one-hot encoding was utilized in [24,26,27], whereas decimal and integer number encoding were applied in [18] and [12] respectively, three feature encoding algorithms in [28], Seq2Vec encoding in [9], and bag-of-words encoding in [10]. Some studies [13,27,28,29,30] have examined a specific gene, such as Spike, instead of an entire genome sequence for classification purposes.

The aforementioned studies achieved good performance and the proposed genome sequence classification models showed promising results. However, they ignored the sequential nature of nucleotides, codons, or amino acids, which is essential to prop-

[1] undp.org/sustainable-development-goals/good-health

erly identify and understand their underlying structure and function. One-hot and bag-of-words encoding are not suitable for genome sequences as they encode nucleotides, codons, or amino acids independently and disregard their order. Furthermore, the feature extraction methods used in most of the aforementioned studies are computationally expensive, resulting in long model-learning times, and are difficult to interpret or comprehend. To our knowledge, the use of frequent sequential patterns for effective or improved classification/detection of genome sequences in different forms has not been extensively studied. In the literature, the studies [31] and [32] utilized frequent sequential patterns found in nucleotide bases and amino acids, respectively, to analyze and classify macromolecule genome sequences and heat shock protein sequences.

By taking the genome sequences in three forms (*nucleotide form* (NF), *coding region form* (CRF) and *protein form* (PF)), this paper introduces a general approach, called GenoAnaCla (Genome Analyzer and Classifier), which utilizes sequential pattern mining (SPM) [33], to analyze and classify genome sequences. The main contributions are:

- First, datasets are developed that comprise genome sequences in three formats associated with various types of RNA viruses. These genome sequences were sourced from the GenBank database and underwent preprocessing to prepare them for the application of SPM algorithms.
- Second, based on the nucleotides present in genome sequences in *NF* and *CRF*, codons in *NF*, and amino acids in *PF*, an analysis and classification model is designed that leverages the discovered frequent sequential patterns of nucleotides, codons, and amino acids to classify genome sequences. Eight classification models are employed and experiments are conducted using various evaluation metrics to thoroughly investigate the effectiveness and generalizability of the proposed approach.

An evaluation of the proposed approach on a developed corpus comprising genome sequences of 15 RNA viruses, in three forms, revealed that employing SPM to identify frequent nucleotide, codon, and amino acid patterns, and subsequently leveraging these sequential patterns, resulted in better classification performance compared to utilizing all the nucleotide bases, codons, and amino acids in the whole genome sequences. Furthermore, the performance of GenoAnaCla was also compared with the most recent approaches for genome detection/classification, and the obtained results demonstrated that it outperformed these methods.

The next four sections of this paper contain the following content: Section 2 discusses the relevant literature on the utilization of computational methods for classifying and detecting genome sequences. Section 3 details the dataset creation and the proposed GenoAnaCla approach. Section 4 discusses the results, and Section 5 concludes the paper.

2 Related Work

In recent years, computational methods based on DL and ML were used for analyzing, predicting, and classifying genome sequences. For instance, [14,15,16] leveraged CpG-based features to classify genome sequences. Naeem et al. [22] developed a system that employed a variety of moment invariants and discrete transforms-based features present in genome sequences for classification. Lopez-Rincon et al. [20] identified representative genome sequences through a combination of DL (convolutional neural network (CNN)) and explainable artificial intelligence methods. Randhawa et al. [25] found intrinsic signatures within genomes and then employed these, along with an alignment-free approach, to accurately classify whole genomes. Ahmad & Jeon [34] classified genomes of various viruses using a range of traditional ML classifiers. Singh et al. [23] first extracted biomarkers using digital signal processing (SDP) techniques, which were then fed into ML classifiers to distinguish between genome sequences of different viruses. El-Dosuky et al. [18] utilized a cockroach swarm-optimized CNN to classify Influenza and SARS-CoV-2 genomes. The classification method of [10] utilized *k-mers* and their occurrence frequencies to predict/-classify viral genomes in human DNA sequence samples. The SPM4GAC [31] and FSP4HSP [32] methods were recently introduced, which used sequential frequent patterns of nucleotides and amino acids, respectively, to classify viruses and heat shock proteins. VirusPredictor [11] employed XGBoost model on features extracted from genome sequences through the *k-mer* and recoding system that comprised three approaches: (1) fixed mapping, (2) physic-chemical property, and (3) DNA-graph based long-range correlation.

For virus classification, some studies have narrowed their focus to specific genes rather than entire genome sequences. For example, Ali et al. [13] utilized *k-mers* and kernel approximation to classify S protein sequences that belong to SARS-CoV-2 variants. Similarly, Kuzmin et al. [27] utilized one-hot encoding on S protein sequences for the classification of coronaviruses. PSAC-PDB [30] was developed for the analysis and classification of protein structures in the Protein Data Bank (PDB) database. It was found that patterns of amino acids can effectively classify protein structures, eliminating the need for full amino acid sequences. Three algorithms were used by Qiang et al. [28] to identify important features in S protein sequences, which were then utilized to train random forest models for classification. SPM algorithms were employed in studies [35,36] to identify hidden nucleotide and amino acid sequential patterns and rules in genome sequences.

DL methods were utilized by Gunasekaran et al. [19] for classifying DNA sequences encoded with *k-mers* and label encoding. An alternative classification method by Dlamini et al. [17] relied on the frequencies of inherent dinucleotide genomic signatures in the classification of pathogenic species. A DL-based classifier called PACIFIC [21] was designed for detecting RNA viruses. This classifier converted nucleotide sequences into numerical tokens using *k-mers*. Then, these tokens were transformed into dense representations by using a continuous vector space. autoBioSeqpy [24], a DL-based method for biological sequence classification, employed dictionary-based and one-hot encoding for nucleotides/amino acids. Dubey et al. [37] introduced two methodologies for genome analysis. The first one focused on determining nucleotide frequencies, while the second one was employed for mutation analysis. Tandan et al. [38] applied association rule mining techniques to find COVID-19 symptom patterns and rules in patients. Acer et al. [39] used seven ML classifiers to classify and detect pancreatic cancer, specifically PDAC (Pancreatic Ductal AdenoCarcinoma), using noninvasive urinary biomarkers and carbohydrate antigen 19-9.

The referenced studies have primarily explored the importance of diverse features for genome classification and detection, and some studies have only considered genome sequences comprising the four main nucleotides. Te present study concentrates on identifying frequent sequential patterns within genomes for enhancing the accuracy and reliability of genome classification and detection. Specifically, SPM is utilized on the processed dataset of genomes in various forms to identify frequent sequential patterns (subsequences of nucleotides, codons, and amino acids) that are subsequently used by various classifiers to perform classification.

To provide a clearer understanding of the methodologies employed in the literature on genome sequence classification/prediction, it is essential to differentiate between sequence alignment and classification methods. Table 1 provides a comparison by highlighting the respective purposes, outputs, methodologies, and applications of sequence alignment and classification.

Table 1: Comparison between sequence alignment and classification

Aspect	**Sequence Alignment**	**Sequence Classification**
Purpose	Compares two or more sequences to identify similarities and differences.	Assigns sequences to categories or classes based on features.
Output	Produces an alignment (e.g., a matrix) showing how sequences match.	Provides a predicted class label for each sequence.
Tools	BLAST, Kalign, Clustal Omega	Support Vector Machine, Random Forest, Neural Networks
Applications	Identifying homologous sequences, studying evolutionary relationships among organisms.	Classifying sequences (e.g., virus classification, functional annotation).
Methodology	Aligns sequences based on scoring systems (e.g., match/mismatch scores, gap penalties).	Uses machine learning or statistical methods to make predictions based on features.
Data Type	Typically works with raw sequence data.	Often requires feature extraction from sequences before classification.

3 The Proposed GenoAnaCla Approach

GenoAnaCla (depicted in Figure 1) is a comprehensive approach for genome detection and classification, which processes genome sequences in three forms. The pipeline of GenoAnaCla consists of three steps: (1) data collection, processing, and corpus development, (2) the extraction of sequential patterns and rules from genome sequences, and (3) training classifier(s) using the patterns and performance evaluation. GenoAnaCla is designed as a generic approach that can be configured with different SPM algorithms and classifier types. The next subsections describe the three steps of GenoAnaCla.

3.1 Data Collection, preprocessing, and corpus development

The first step consists of collecting the data and preparing it for further processing in the following steps. More precisely, the sequencing data for fifteen RNA viruses has been retrieved from GenBank [1]. The genomes are downloaded in the FASTA format

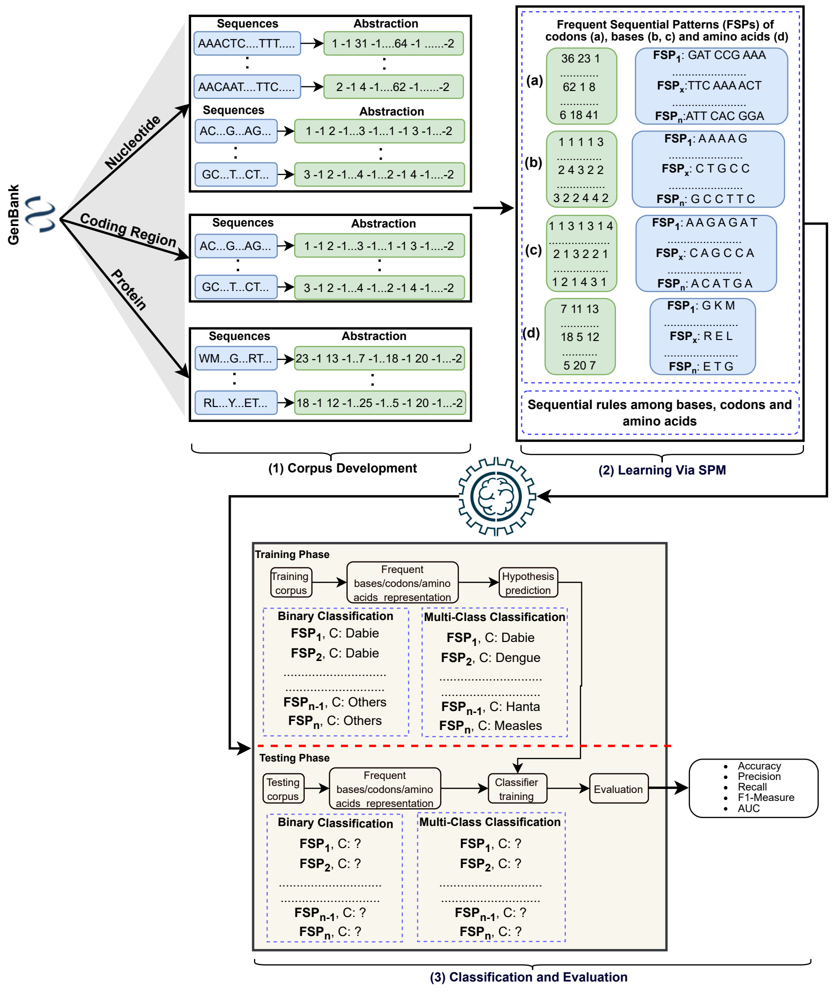


Fig. 1: The GenoAnaCla approach for the analysis and classification of genome sequences in three forms, consists of three steps: (1) data collection, preprocessing, and corpus development; (2) using SPM to find frequent bases, codons, amino acids, and their frequent sequential patterns and rules; and (3) using the identified patterns of nucleotides/codons/amino acids for classification by training various classifiers. C stands for class or category.

in three forms: (1) *Nucleotide form (NF)*, (2) *Coding region form (CRF)*, and (3) *Protein form (PF)*. Genome sequences in *NF* provide the foundational level of genetic information, essential to understanding the structure of the entire genome, variations, and the potential regulatory elements. Conversely, within *CRF*, genome sequences signify the genetic parts that are translated into proteins, playing an important role in identifying functional genes, elucidating gene expression patterns, and their direct correlation with phenotypic characteristics and disease associations. Furthermore, the genome sequences in *PF* are vital to understand the functional aspects of genes. They enable the examination of protein architecture, functionality, and interactions, offering valuable insights into post-translational modifications and the functional consequences of genetic mutations.

Only complete genomes with extensive coverage were selected. The FASTA format for genome sequences starts with a definition line, which contains the identifier and other information pertaining to the genome sequence. A genome sequence in *NF* contains a single definition line, whereas in the *CRF* and *PF*, a sequence may have multiple definition lines. We developed a Python script (accessible at: `github.com/keyboardman37/GenoAnaCla`) to preprocess the genome sequences. In the first form, the script outputs each genome sequence in one line, and for the other two forms, the script generates each coding region and protein sequence in individual lines. The total count of genome sequences in *NF*, *CRF*, and *PF* are listed in Table 2.

Table 2: Viruses genome sequences retrieved from NCBI GenBank in three forms.

Virus	**RNA Type**	**Sample in NF**	**Samples in CRF**	**Samples in PF**
Dengue	(+)ssRNA	4,841	4,826	4,882
Dabie Banda	(-)ssRNA	2,937	2,937	3,984
Hanta	(-)ssRNA	1,154	1,140	1,162
SARS-CoV-2	(+)ssRNA	9,156	135	1,611
Ebola	(-)ssRNA	605	605	5,406
MERS	(+)ssRNA	657	656	7,129
HIV	(-RT)ssRNA	7,068	6,496	54,847
Hepaci	(+)ssRNA	1,214	1,190	1,697
Rhino	(+)ssRNA	893	893	915
Influenza	(-)ssRNA	11,229	11,201	15,056
Noro	(+)ssRNA	1,565	1,565	4,776
Rota	dsRNA	2,711	2,704	2,831
Measles	(-)ssRNA	770	648	4,285
Rabies	(-)ssRNA	2,559	2,558	10,792
West Nile	(+)ssRNA	1,914	1,906	2,848
Total		49,273	39,460	122,211

NF: Nucleotide Form, CRF: Coding Region Form, PF: Protein Form, RNA: Ribonucleic acid, (+)ssRNA: Positive-sense single-stranded RNA, (-)ssRNA: Negative-sense single-stranded RNA, (-RTssRNA): Negative-sense reverse transcription single-stranded RNA, dsRNA: Double-stranded RNA.

After this initial preprocessing, genome sequences are converted into sequences of discrete items, leveraging the "*bases/codons/amino acids to integers*" abstraction for this transformation. In this scheme, each unique base, codon, or amino acid is assigned a distinct positive integer. The genome sequences are encoded using this representation. Moreover, a special code of -1 is added as a separator between bases, codons, and amino acids, and -2 is put at the end of each sequence to show that it has ended [40]. The reason for transforming sequences in this integer-based format is that it is required for applying SPM algorithms, which are generic tools for analyzing sequences that may or may not be from the biological field. It is to be noted that the transformation process is lossless and reversible. In other words, it is possible to transform the data back and forth between the raw sequence format and the integer-based format.

The following is a comprehensive description of the transformation process, which will be followed by an example. Let $NB = \{A,C,G,T,R,Y,S,W,K,M,B,D,H,V,N\}$ and $AA = \{A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y\}$ be the sets of distinct nucleotides and amino acids respectively. In genome sequences, nucleotides other than the four bases (*A, C, G* and *T*) can occur, representing different combinations of these four bases. These nucleotides are referred to as *redundant nucleotides* (denoted as RN) due to their infrequent occurrence. Similarly, some amino acids represent the different combinations of 20 amino acids, and they are called *redundant amino acids* (RAA). Redundant samples are those that contain RN or RAA. The percentage of redundant samples in collected genomes is: NF (24.7%), CRF (6.47%), and PF (4.7%). The transformation process converts each distinct or unique *RN* or *RAA* by assigning a distinct or unique positive integer.

In *NF* and *CRF*, a raw genome sequence is represented as a list of nucleotides, written as $GS = \langle NB_1, NB_2\ , ..., NB_n\rangle$, such that $NB_i \in NB$ $(1 \leq i \leq n)$. In *PF*, a raw genome sequence is represented as a list of amino acids, written as $GS = \langle AA_1, AA_2\ , ..., AA_n\rangle$, such that $AA_i \in AA$ $(1 \leq i \leq n)$. We also consider codons, which are groups of three nucleotides. In *NF*, a raw genome sequence can be represented as a list of codons, such as $GS = \langle NB_1NB_2NB_3, ..., NB_{n-2}NB_{n-1}NB_n\rangle$, where any list of three consecutive nucleotides $NB_{i-2}NB_{i-1}NB_i \in NB$ $(1 \leq i \leq n)$ constitutes a codon.

Let $GSC = \langle GS_1, GS_2, ..., GS_p\rangle$ denote a corpus of raw genome sequences. The transformation process involves converting each raw genome sequence from GSC to the integer-based format (abstraction).

In *NF*, a raw genome sequence denoted as $GS = \langle NB_1, NB_2\ , ..., NB_n\rangle$ is converted into $GS' = \langle f(NB_1)$, -1, $f(NB_2)$, -1, ..., $f(NB_n)$, -1, -2 $\rangle$ by applying a transformation function f, which will be defined after. Similarly, the transformation of a raw sequence of codons in NF (called *CNF*) yields a sequence $GS' = \langle f(NB_1NB_2NB_3)$, -1, $f(NB_4NB_5NB_6)$, -1, ..., $f(NB_{n-2}NB_{n-1}NB_n)$, -1, -2 $\rangle$. And in *PF*, a raw sequence $GS = \langle AA_1, AA_2\ , ..., AA_n\rangle$ is transformed as $GS' = \langle f(AA_1), -1,\ f(AA_2), -1,\ ...,\ f(AA_n), -1, -2\ \rangle$. In the transformation process, the function f (f:$NB \rightarrow \mathbb{N}$ (or f:$AA \rightarrow \mathbb{N}$ or f:$Codon \rightarrow \mathbb{N}$)) maps each nucleotide (or amino acid or codon) to a distinct positive integer. Note that codons can encode 20 different amino acids, and three codons are stop codons that halt the cell's protein synthesis. The collected datasets of genome sequence and their respective transformation are available at: `github.com/keyboardman37/GenoAnaCla`. The transformation process for sample genome sequences in NF, CRF, and PF is illustrated in Table 3.

For example, consider the first raw sequence $\langle\{$TATCAGTCG$\}\rangle$ in Table 2 (a), with ID 1. It is transformed into the sequence with ID 1 in Table 2 (b), which is: 4 -1 1 -1 4 -1 2 -1 1 -1 3 -1 4 -1 2 -1 3 -1 -2. The transformation process works as follows in this example: the base T is replaced by the integer 4, while the bases A, C and G have been replaced by the integers 1, 2, and 3, respectively. The separator -1 is added between bases and -2 indicates the end of the sequence.

To validate the integrity of transformation, we compared the frequency distributions of nucleotides, amino acids and codons before and after transformation. The consistency in these distributions supports the conclusion that the integer-based transformation preserves the original biological characteristics of the genomic data.

Table 3: Three sample genome sequence corpuses (GSC), containing five sequences in (a) NF/CRF, (c) PF, and (e) codons in NF, respectively. The corresponding transformed genome sequence corpuses of (b) bases, (d) amino acids, and (f) codons

.

(a) A GSC in NF/CRF

ID	Sequence
1	⟨{TATCAGTCG}⟩
2	⟨{GTACAGTAA}⟩
3	⟨{TTGTGGACT}⟩
4	⟨{CTAATCGCA}⟩
5	⟨{CCAAGTGTA}⟩

(b) The transformed GSC in NF/CRF

ID	Sequence
1	4 -1 1 -1 4 -1 2 -1 1 -1 3 -1 4 -1 2 -1 3 -1 -2
2	3 -1 4 -1 1 -1 2 -1 1 -1 3 -1 4 -1 1 -1 1 -1 -2
3	4 -1 4 -1 3 -1 4 -1 3 -1 3 -1 1 -1 2 -1 4 -1 -2
4	2 -1 4 -1 1 -1 1 -1 4 -1 2 -1 3 -1 2 -1 1 -1 -2
5	2 -1 2 -1 1 -1 1 -1 3 -1 4 -1 3 -1 4 -1 1 -1 -2

(c) A GSC in PF

ID	Sequence
1	⟨{NYYEELGV}⟩
2	⟨{RDYYEILN}⟩
3	⟨{DLYSVLGV}⟩
4	⟨{LGVYRFRE}⟩
5	⟨{NRDLVNVE}⟩

(d) The transformed GSC in PF

ID	Sequence
1	14 -1 25-1 25 -1 5 -1 5 -1 12 -1 7 -1 22 -1 -2
2	18 -1 4 -1 25 -1 25 -1 5 -1 9 -1 12 -1 14 -1 -2
3	4 -1 12 -1 25 -1 19 -1 22 -1 12 -1 7 -1 22 -1 -2
4	12 -1 7 -1 22 -1 25 -1 18 -1 6 -1 18 -1 5 -1 -2
5	14 -1 18 -1 4 -1 12 -1 22 -1 14 -1 22 -1 5 -1 -2

(e) A GSC with codons in NF

ID	Sequence
1	⟨{TAT CAG TCG}⟩
2	⟨{GTA CAG TAA}⟩
3	⟨{TTG TGG ACT}⟩
4	⟨{CTA ATC GCA}⟩
5	⟨{CCA AGT GTA}⟩

(f) The transformed GSC with codons in NF

ID	Sequence
1	52 -1 19 -1 55 -1 -2
2	45 -1 19 -1 49 -1 -2
3	63 -1 59 -1 8 -1 -2
4	29 -1 14 -1 37 -1 -2
5	21 -1 12 -1 45 -1 -2

GSC: Genome Sequence Corpus, NF: Nucleotide Form, CRF: Coding Region Form, PF: Protein Form

3.2 Using SPM for learning

The second step is the extraction of sequential patterns from each transformed genome sequence corpus. It is important to mention here that SPM is a special case of frequent itemset mining (FIM) [41]. The studies [38,42] used the Apriori algorithm [43], a rule-based algorithm for FPM, to find feature patterns and rules in clinical datasets. The study [44] proposed a constraint-based algorithm, based on CHARM [45], to find frequent closed itemsets of symptoms, diagnosis and medication in a clinical dataset. However, Apriori and CHARM do not take into account the sequential arrangement of features present in clinical datasets. SPM is used because it considers the sequential order of nucleotides, codons, or amino acids. This consideration is crucial to properly identify and understand their underlying structure and function. The aim is to identify sequential patterns (subsequences from genome sequences) that could be used to characterize the genome sequences. These sequential patterns will then be used in the third step of GenoAnaCla for the classification of genome sequences.

To uncover interesting and hidden patterns, it is necessary to use an appropriate measure for selecting these patterns. The *support* measure (occurrence frequency) [33, 41] is the most widely used measure in pattern mining to identify patterns. To explain it formally, it is first necessary to explain the concept of subsequence.

In *NF* and *CRF*, a sequence of nucleotides $GS_x = \langle NB'_1, NB'_2\ ,..., NB'_m\rangle$ is a *subsequence* of another sequence of nucleotides $GS = \langle NB_1, NB_2\ ,..., NB_n\rangle$, which is denoted as $GS_x \sqsubseteq GS$, if an only if there exists some integers i_1, i_2, $\ldots i_m$ such that $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ and $NB'_1 = NB_{i_1}$, $NB'_2 = NB_{i_2}, \ldots NB'_m = NB_{i_m}$. For example, the sequence $\langle\{\text{CAG}\}\rangle$ is a subsequence of $\langle\{\text{TATCAGTCG}\}\rangle$, that is $\langle\{\text{CAG}\}\rangle \sqsubseteq \langle\{\text{TATCAGTCG}\}\rangle$.

In *PF*, an amino acid sequence $GS_x = \langle AA'_1, AA'_2\ ,..., AA'_m\rangle$ is a *subsequence* of another amino acid sequence $GS = \langle AA_1, AA_2\ ,..., AA_n\rangle$, which is denoted as $GS_x \sqsubseteq GS$, if an only if there exists some integers i_1, i_2, $\ldots i_m$ such that $1 \leq i_1 < i_2 < \ldots < i_m \leq n$ and $AA'_1 = AA_{i_1}$, $AA'_2 = AA_{i_2}$, $\ldots AA'_m = AA_{i_m}$. For instance, the sequence $\langle\{\text{NYY}\}\rangle$ is a subsequence of $\langle\{\text{NYYEELGV}\}\rangle$, and hence $\langle\{\text{NYY}\}\rangle \sqsubseteq \langle\{\text{NYYEELGV}\}\rangle$.

In the case of *NF*, a codon sequence $GS_x = \langle NB'_1, NB'_2\ ,..., NB'_m\rangle$ is a *subsequence* of another codon sequence $GS = \langle NB_1, NB_2\ ,..., NB_n\rangle$, which is denoted as $GS_x \sqsubseteq GS$, if an only if there exists some integers i_1, i_2, $\ldots i_{\frac{m}{3}}$ such that $1 \leq i_1 < i_2 < \ldots < i_{\frac{m}{3}} \leq n$, $NB'_1 = NB_{i_1}$, $NB'_2 = NB_{i_1+1}$, $NB'_3 = NB_{i_1+2}$, $NB'_4 = NB_{i_2}$, $NB'_5 = NB_{i_2+1}$, $NB'_6 = NB_{i_2+2}$, $\ldots\ NB'_{m-2} = NB_{i_{\frac{m}{3}}}$, $NB'_{m-1} = NB_{i_{(\frac{m}{3}+1)}}$, $NB'_m = NB_{i_{(\frac{m}{3}+2)}}$. For instance, the sequence $\langle\{\text{CAG TCG}\}\rangle$ is a subsequence of $\langle\{\text{TAT CAG TCG}\}\rangle$, and hence $\langle\{\text{CAG TCG}\}\rangle \sqsubseteq \langle\{\text{TAT CAG TCG}\}\rangle$.

Based on the concept of subsequence, the support measure is defined as follows. Consider a transformed GSC and a subsequence GS_x. The *support* of the subsequence GS_x, denoted as $sup(GS_x)$, is the total number of genome sequences from GSC that contain GS_x. Formally, it is defined as:

$$sup(GS_x) = |\{GS | GS \in GSC \wedge GS_x \sqsubseteq GS\}| \tag{1}$$

For this study, this measure is relevant because it allows discovering subsequences of nucleotides, codons, and amino acids that frequently occur in genomes, thereby uncovering their similarities. The task of SPM with the *support* measure is called *frequent SPM*. This task involves enumerating all frequent (sub)sequences in a discrete sequence set [33]. For a given GSC and a user-defined minimum sup value ($minsup \in [0, 1]$), the objective of this task is to identify all subsequences that are frequent. A genome sub-sequence GS_x is *frequent* if and only if its support is greater than or equal to $minsup$, i.e. $sup(GS_x) \geq minsup$. The $minsup$ threshold can be viewed as a filter to eliminate patterns that rarely occur, and thus may be less significant than frequent ones.

Frequent SPM has been widely used to analyze various types of data in many fields, with several algorithms designed for this task. These algorithms generally have the same input and output but differ in terms of their inner workings, such as data structures, search strategies, and optimizations, which influence their runtimes. In this work, the CM-SPAM [46] algorithm is used to find frequent sequential patterns of nucleotides, codons, and amino acids due to its efficiency and availability in open-source software. However, it is important to note that GenoAnaCla can also accommodate other SPM algorithms. CM-SPAM [46] is an improved version of the SPAM [47] algorithm, with the key improvement being the use of a data structure called CMAP (Co-occurrence MAP), which stores information about the co-occurrences of items to speed up the identification of frequent sequential patterns. Additionally, CM-SPAM offers useful features to set constraints on the minimum and maximum length of patterns and the maximal gap between consecutive items in patterns. These constraints can be used to filter patterns that are deemed too short, too long, or where items are separated by too big gaps. Since the CM-SPAM algorithm is not a novelty of this work, interested readers are referred to the paper describing CM-SPAM for more details about its inner workings [46].

Frequent SPM algorithms identify sequential patterns based on the support measure. However, a potential drawback of using that measure is that it does not assess the likelihood of some items (bases, amino acids, or codons) following others in a pattern. To evaluate this aspect, an alternative approach is to extract another type of patterns called sequential rules, using an additional measure called the *confidence*. Sequential rules are used in various fields, especially for tasks requiring prediction, forecasting, and decision-making [48]. A sequential rule is a relationship between two sets of items that considers both the *confidence* (also called conditional probability) and *support* of items. A sequential rule can be denoted as $A \rightarrow B$ to represent a sequential relationship between two non-empty and disjoint sets of nucleotides (or codons or amino acids) A, B. The interpretation of a rule r:$A \rightarrow B$ is that if A's items are present in a sequence, B's items will appear after (within the same sequence). It is said that a sequence $S_a = \langle a_1, a_2, \ldots, a_n \rangle$ contains A, iff $A \subseteq \bigcup_{x=1}^{n} \{a_x\}$. Besides, a sequence S_a contains r (denoted as $r \sqsubseteq S_a$) in the case where there exists an integer k s.t. $1 \leq k < n$, $A \subseteq \bigcup_{x=1}^{k} \{a_x\}$, and $B \subseteq \cup_{x=k+1}^{n} \{a_x\}$. For a rule r in a GSC, its confidence and support are defined as:

$$conf_{GSC}(r) = \frac{|\{S|r \sqsubseteq S \wedge S \in GSC\}|}{|\{S|A \sqsubseteq S \wedge S \in GSC\}|} \tag{2}$$

$$sup_{GSC}(r) = \frac{|\{S|r \sqsubseteq S \wedge S \in GSC\}|}{|GSC|} \tag{3}$$

For instance, the rule $E \rightarrow L$ has a support of 0.4 in the GSC of Table 2 (c) because E is followed by L in 2 out of 5 sequences. The confidence of that rule is 2/4 = 0.5 because E appears in 4 sequences and L follows E in 2 sequences. Therefore, the confidence indicates that the likelihood that L follows E is 50%.

For a GSC, and $minsup$, $minconf \in [0, 1]$ thresholds set by the user, a rule r is a frequent sequential rule iff $sup_{GSC}(r) \geq minsup$ and r is a valid sequential rule iff it is frequent and $conf_{GSC}(r) \geq minconf$. The task of identifying sequential rules consists of finding all valid rules. As for SPM, multiple algorithms have been proposed to identify sequential rules in sequences. In this study, the ERMiner [48] algorithm is used to find the valid sequential rules among nucleotides, codons, and amino acids, because it is an efficient algorithm and it is open-source. However, it should be noted that other algorithms could be used as replacement in GenoAnaCla. ERMiner is an algorithm that performs a depth-first search to find rules and rely on the concept of equivalence classes and several optimizations to accelerate the discovery process. An equivalence class is a group of rules that has either identical antecedents or consequents. ERMiner gradually uncovers sequential rules using two types of rule merging operations, called left and right merging. As ERMiner is an existing algorithm, interested readers are referred to the original paper about ERMiner for more details about its inner workings [48].

In summary, SPM and sequential rule mining are two different tasks that vary in several aspects, including the types of patterns that are discovered and the measures used to identify these patterns. Multiple algorithms exist for each task, with the same input and output but varying in data structures, search strategies and optimizations to reduce runtimes or offer optional constraints. The proposed GenoAnaCla approach can be configured to use various algorithms.

3.3 Classification through discovered frequent patterns

The third and final step of GenoAnaCla involves utilizing the frequent sequential patterns and rules identified in the second step (Section 3.2) for the classification of genomes. This classification process contains two distinct phases: (1) the training phase, and (2) the testing phase. The purpose of the training phase is to build classification models (classifiers) using the training data, while the goal of the testing phase is to evaluate the models that have been built to see if they also perform well on unseen data.

The training phase consists of two steps, executed sequentially: (a) frequent nucleotides, codons, or amino acids representation, and (b) classifier training. The first step prepares the training data, while the second step aims to build the classification models using the prepared data.

The testing phase consists of three steps: (a) frequent nucleotides, codons, or amino acids representation, (b) hypothesis prediction, and (c) evaluation. The first step prepares the testing data, while the second step involves using the built models to make predictions on unseen data. The third step calculates measures to determine the overall performance of the models based on the predictions.

An important consideration for classifying genome sequences is how to handle repetitive occurrences. Genome sequences are generally long, and the majority of them in *NF* and *CRF* consist of repetitive occurrences of the same bases, often tens to hundreds of times. Similarly, the genome sequences in *PF* contain repeated instances of the same amino acids. For better classification, the repetitive sequences of bases or amino acids can be replaced with their frequent sequential patterns. More specifically, GenoAnaCla utilizes frequent sequential patterns of nucleotides, codons, and amino acids to classify various RNA virus families.

Binary as well as multi-class (MC) classification are conducted in this study. Binary classification is employed for training a classifier to distinctly classify each type of virus. In the MC classification method, each genome sequence is assigned a label corresponding to its respective type name, with 15 distinct viruses listed in Table 2. A classification model is trained to accurately label genome sequences. For both classification types, one DL and seven standard ML algorithms are used, which are: (1) Logistic Regression (LR), (2) Random Forest (RF), (3) k-Nearest Neighbors (kNN), (4) Gaussian Naive Bayes (GNB), (5) Support Vector Machine (SVM), (6) Multilayer Perceptron (MLP), (7) Decision Tree (DT), and (8) Multinomial Naive Bayes (MNB) . The reason for using conventional classifiers is that they provide a solid foundation for baseline comparison, making it easier to evaluate the impact of the frequent sequential patterns discovered in genome sequences as features in the classification process. The performance of the classifiers is assessed using five metrics: (1) Accuracy (ACC), (2) Recall (R), (3) Precision (P), (4) F1 score (F1), and (5) Area under curve (AUC). The five metrics are defined as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{5}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{6}$$

$$F - measure = 2 \times \frac{P \times R}{P + R} \tag{7}$$

$$\text{AUC} = \int_0^1 \text{R}(d\text{FPR}) \tag{8}$$

where TP = true positive, TN = true negative, FP = false positive, and FN = false positive. In the context of this work, TP represents the correctly identified sequential patterns of nucleotides, codons, or amino acids to a specific virus type. TN represents the correctly classified sequential patterns of nucleotides, codons, or amino acids as not part of a specific virus type. FP represents the incorrectly identified sequential patterns of nucleotides, codons, or amino acids to a specific virus type. FN represents the incorrectly identified sequential patterns of nucleotides, codons, or amino acids as not part of the specific virus type. In equation 8, *dFPR* is the derivative of the false positive rate (*FPR*), that is equal to $\frac{FP}{FP+TN}$.

Algorithm 1 provides the overall pseudocode of the proposed GenoAnaCla approach. The experimental evaluation of GenoAnaCla is presented in the next section.

4 Results

All the experiments were conducted on a computer system equipped with an 11th-generation Intel Core i5 processor and 8 GB of RAM. The open-source cross-platform SPMF library [40], developed in Java, was used to analyze and discover patterns in the transformed datasets of genome sequences. Python was used for the classification process. Several libraries were used, such as scikit-learn [49] to train and evaluate

Algorithm 1 GenoAnaCla

```
    Input: Genome sequences ($GSs$) of RNA viruses in $NF$, $CRF$, $PF$, and $CNF$
    Output: Classification results including ACC, P, R, F1, and AUC
 1: procedure GENOANACLA
    // Step 1: Convert sequences into the needed abstraction
 2:     Abstraction ← Convert $GS$s to integer-based representation

    // Step 2: Find frequent sequential patterns ($FSP$s)
 3:     $FSP$s ← Extract frequent sequential patterns and rules in abstracted $GS$s

    // Step 3: Train classifiers
 4:     for each classifier in Classifiers do
 5:         Train classifier with $FSP$s as features using default hyperparameters
 6:     end for

    // Step 4: Evaluate classifiers
 7:     for each classifier in Classifiers do
 8:         Evaluate classifier using 80:20 training:testing ratio
 9:         Store metrics: ACC, P, R, F1, and AUC
10:     end for

    // Output classification results
11:     Return ACC, P, R, F1, and AUC
12: end procedure
```

classifiers, Pandas[2] for data manipulation, and NumPy[3] for numerical calculations. To ensure reliable model evaluation, the datasets were split into training and testing subsets (80% training and 20% testing) using the "train_test_split" function of scikit-learn. The default hyperparameters for classifiers were used for both binary and multi-class classification (Table 4).

Table 4: Hyperparameters and their corresponding values for the eight classifiers.

Classifier	Parameters
LR	max iterations: 100, solver: lbfgs, C: 1
RF	max depth: none, estimators: 100, criterion: gini, min samples leaf: 1, min samples split: 2
kNN	weight scheme: uniform, neighbors: 2, algorithm: auto, leaf size: 30 distance metric: euclidean
GNB	default (no significant hyperparameters)
SVM	gamma: scale, kernel: rbf, C: 1, degree: 3
MLP	optimizer: Adam, activation: tanh, $\alpha = 0.0001$, size of hidden layers 600, learning rate init: 0.001, learning rate: invscaling,
DT	criterion: gini, max depth: none, splitter: best, min samples split: 2, min samples leaf: 1
MNB	$\alpha = 1$

LR: Logistic Regression, RF: Random Forest, kNN: k-Nearest Neighbors, GNB: Gaussian Naive Bayes, SVM: Support Vector Machine, MLP: Multilayer Perceptron, DT: Decision Tree, MNB: Multinomial Naive Bayes

[2] pandas.pydata.org/
[3] numpy.org/

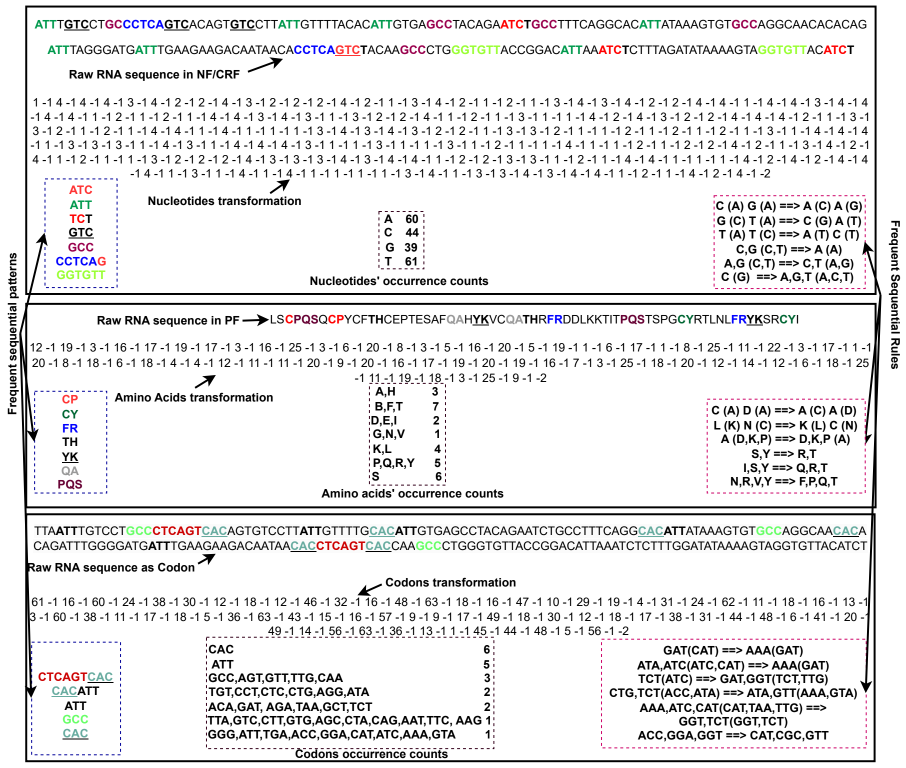
4.1 Frequent Patterns and Rules

To extract patterns in genome sequences, three algorithms were applied. The first two, CM-SPAM and ERMiner, which were presented in the previous section, are used to identify sequential patterns and rules of nucleotides, codons, or amino acids in genome sequences. In addition to these algorithms, the Apriori [43] algorithm was employed to count the frequency of individual nucleotides, codons, and amino acids in each dataset.

To give a visual overview of some patterns discovered through this process, a simple example is offered in Figure 2, where the obtained results with the three aforementioned algorithms are explained. The top box of the figure displays a raw genome sequence in *NF* and *CRF*, containing nucleotide bases, followed by their transformation. The middle box shows another raw RNA sequence in *PF*, containing amino acids, followed by their transformation. The bottom box contains another raw RNA sequence in *NF*, containing codons (three nucleotides together), followed by their transformation. In the bottom center of each box, bases (top box), amino acids (middle box), and codons (bottom box) compositions, obtained by Apriori are listed. In the bottom left of each box, the frequent sequential patterns discovered in nucleotides (top box), amino acids (middle box), and codons (bottom box) are shown with their respective colors. It is noticeable that some frequent sequential patterns of nucleotides (*ATT*, *CCTCAG*), amino acids (*CP*, *PQS*) and codons (*CTCAGT*, *GCC*) occur more than once at various positions within the sequence. The patterns identified within sequences can be interpreted as a characterization or description of those sequence. In the bottom right side of each box, the frequent sequential rules of nucleotides (top), amino acids (middle), and codons (bottom) are listed with varying numbers of antecedents and consequent.

With regards to the identification of patterns in genomes, we observed that the process was relatively efficient and quick. Nevertheless, when dealing with virus types such as SARS-CoV-2 (SC2), MERS, and Ebola, which have lengthy genome sequences, adjustments to certain algorithm parameters and fine-tuning are necessary to obtain the desired patterns.

The heatmap of Figure 3 provides a visual representation of codon frequencies, indicated by a color code, found in 15 virus families. The heatmap offers important insights into the codon usage patterns among different viruses. For instance, the dark blue color codons (*AAA, AAT, TTT*) are highly frequent in certain virus families, and light yellow color codons (such as *ACG, CGA, TCG*) are less common in other viruses. The codon *AAA* is the most frequent one in Dengue, Ebola, Hanta, HIV, and Rhino, the second most frequent one in Influenza, and third most frequent in SARS-CoV-2. Similarly, *AAT* is the second most frequent one in Rhino and Rota, and the third in Ebola and Influenza, Interestingly, the one-stop codon (*TAA*) is also present. Such analysis helps in the identification of conserved codons across multiple virus families. Moreover, the ANOVA statistical test [50] was performed, and the obtained *p-value* was below the standard significance level of 0.05, indicating that there is a statistically significant difference in codon frequencies across different viruses.



Raw RNA sequence in NF/CRF
Nucleotides transformation
Nucleotides' occurrence counts
A 60
C 44
G 39
T 61
ATC
ATT
TCT
GTC
GCC
CCTCAG
GGTGTT
C (A) G (A) ==> A (C) A (G)
G (C) T (A) ==> C (G) A (T)
T (A) T (C) ==> A (T) C (T)
C,G (C,T) ==> A (A)
A,G (C,T) ==> C,T (A,G)
C (G) ==> A,G,T (A,C,T)
Raw RNA sequence in PF
LSCPQSQCPYCFTHCEPTESAFQAHYKVCQATHRFRDDLKKTITPQSTSPGCYRTLNLFRYKSRCYI
Amino Acids transformation
Amino acids' occurrence counts
A,H 3
B,F,T 7
D,E,I 2
G,N,V 1
K,L 4
P,Q,R,Y 5
S 6
CP
CY
FR
TH
YK
QA
PQS
C (A) D (A) ==> A (C) A (D)
L (K) N (C) ==> K (L) C (N)
A (D,K,P) ==> D,K,P (A)
S,Y ==> R,T
I,S,Y ==> Q,R,T
N,R,V,Y ==> F,P,Q,T
Raw RNA sequence as Codon
Codons transformation
Codons occurrence counts
CAC 6
ATT 5
GCC,AGT,GTT,TTG,CAA 3
TGT,CCT,CTC,CTG,AGG,ATA 2
ACA,GAT, AGA,TAA,GCT,TCT 2
TTA,GTC,CTT,GTG,AGC,CTA,CAG,AAT,TTC, AAG 1
GGG,ATT,TGA,ACC,GGA,CAT,ATC,AAA,GTA 1
CTCAGTCAC
CACATT
ATT
GCC
CAC
GAT(CAT) ==> AAA(GAT)
ATA,ATC(ATC,CAT) ==> AAA(GAT)
TCT(ATC) ==> GAT,GGT(TCT,TTG)
CTG,TCT(ACC,ATA) ==> ATA,GTT(AAA,GTA)
AAA,ATC,CAT(CAT,TAA,TTG) ==>
GGT,TCT(GGT,TCT)
ACC,GGA,GGT ==> CAT,CGC,GTT
Frequent sequential patterns
Frequent Sequential Rules

Fig. 2: Nucleotides, amino acids, and codons compositions, their frequent sequential patterns and rules discovered in raw RNA sequences in NF/CRF (top), PF (middle) and codon (below). The frequent sequential patterns found in genome sequences in different forms can be interpreted as their descriptors or features. *NF* and *CRF* stand for Nucleotide Form and Coding Region From respectively.

4.2 Classification Results

For classification, the identified patterns undergo more preprocessing to make sure they contain three to four distinct frequent bases, codons, or amino acids. Table 5 provides the classification (both binary and MC) results for patterns that are identified using the CM-SPAM algorithm. The results for a classifier's metrics in each table are provided in the following format: $\frac{NF(CRF)}{PF(CNF)}$. For example, the first entry of $\frac{93.6(92)}{93.2(92.9)}$ in Table 5 indicates that MNB achieved an accuracy of 93.6%, 92%, 93.2% and 92.9% on frequent patterns found in Dengue for *NF*, *CRF*, *PF* and *CNF* respectively. This format is used for metrics throughout this section to reduce the total number of tables. It was observed that all classifiers, except GNB, performed better, on average, in binary classification.

For binary classification, DT, kNN, RF, and SVM achieved an average accuracy of approximately 94%, whereas MLP, LR, MNB, and GNB achieved an average ac-

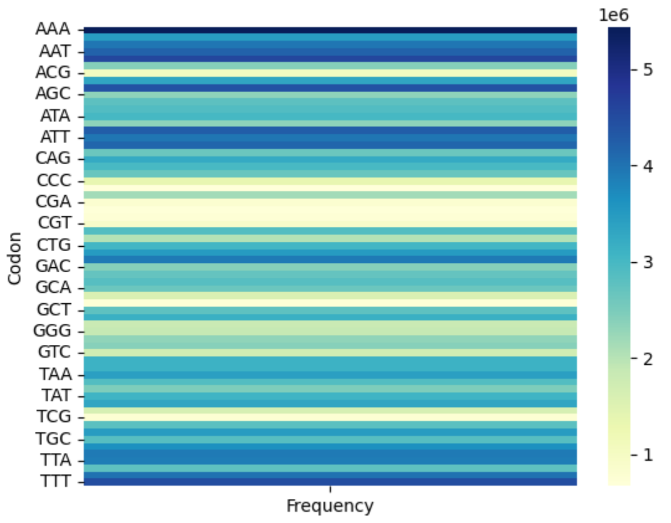


Fig. 3: Heatmap of codon frequency in 15 viruses. Dark blue codons are the most frequent ones and light yellow codons are the least frequent ones.

curacy of 93.77%, 93.75%, 93.32%, and 55.4% respectively. In terms of computational time, GNB and MNB performed the best, whereas MLP was slow compared to the other models. Interestingly, all classification models, except for GNB, exhibited improved performance on the discovered frequent patterns of nucleotides, amino acids, and codons. On NF, CRF, PF, and CNF, the classifiers, excluding GNB, achieved an average accuracy of 93.53%, 94.34%, 93.58% and 93.94%, respectively. SVM achieved the highest accuracy of 99.97% on the frequent sequential patterns of nucleotides and codons. One of the reasons for the relatively better performance of DT, kNN, RF and SVM, compared to others could be their ability to properly handle various patterns as features during the process of classification. The detailed results for SVM on genome sequences of viruses in three forms and in codons are listed in Table 6.

For MC classification, SVM performed better on average than other models, followed by MLP and kNN. The average accuracy of classification models for MC classification on frequent patterns found in NF, CRF, PF, and CNF is 31.1% 43.1%, 22%, and 42.1% respectively. The classifiers took more time for MC classification, and their results were not as encouraging, particularly for NF and PF. We believe that the main reasons for this are (1) The large number of virus classes (a total of 15), for MC classification. For fewer classes, the accuracy increased. For example, for 3 virus classes, SVM achieved more than 70% accuracy. (2) The presence of similar patterns in more than one virus class. As in MC classification, the goal is to assign a frequent pattern to a single virus class. It is important to point out here that the *GridSearchCV* of scikit-learn was also used for classifiers fine-tuning and to find the best hyperparameters. The obtained results for both types of classification with *Grid-*

Table 5: Classifiers accuracy (in %) via frequent patterns of bases, amino acids, and codons.

Virus	MNB	GNB	DT	RF	MLP	SVM	kNN	LR
Dengue	$\frac{93.6(92)}{93.2(92.9)}$	$\frac{12.4(61.6)}{85.1(53.2)}$	$\frac{93.6(95)}{93.1(93.7)}$	$\frac{93.6(95)}{93.1(93.7)}$	$\frac{93.6(95)}{92(92.4)}$	$\frac{93.6(95)}{93(93.7)}$	$\frac{93.4(93.3)}{93.8(92.8)}$	$\frac{93.6(95)}{93(93.7)}$
Dabie	$\frac{93(93)}{93(92.2)}$	$\frac{13(72.3)}{86.5(41.1)}$	$\frac{92.3(92)}{94.4(94.2)}$	$\frac{93(92.3)}{93.5(93.3)}$	$\frac{93(93)}{94.7(94.4)}$	$\frac{93(93.3)}{93(94)}$	$\frac{92.8(90)}{96(94.4)}$	$\frac{93(90.6)}{93.4(93.9)}$
Hanta	$\frac{92.7(93.3)}{92.8(92.6)}$	$\frac{13.3(64)}{85(41.9)}$	$\frac{92.7(92.6)}{92.9(92.7)}$	$\frac{92.7(93)}{92.8(92.6)}$	$\frac{91.7(93.3)}{92.2(92.8)}$	$\frac{92.7(93.3)}{92.7(92.6)}$	$\frac{92.8(92)}{93(92.4)}$	$\frac{92.7(93.3)}{92.7(92.9)}$
SC2	$\frac{97.9(92.6)}{93.1(93.2)}$	$\frac{97.9(91.6)}{65.4(43.8)}$	$\frac{97.9(98.3)}{93.1(93.9)}$	$\frac{97.9(98.6)}{93.1(93.3)}$	$\frac{95.4(96.6)}{92(93.7)}$	$\frac{\mathbf{99.9}(96.6)}{93.1(93.3)}$	$\frac{97.7(97.6)}{92.5(93.7)}$	$\frac{97.9(95.3)}{93.1(94)}$
Ebola	$\frac{93(92.6)}{93.7(92.7)}$	$\frac{13.2(88.6)}{81.3(43.7)}$	$\frac{95.1(96.3)}{94.9(93.2)}$	$\frac{94.1(96.6)}{96(92.3)}$	$\frac{94.9(94)}{96.2(93)}$	$\frac{95.7(92.6)}{95.8(92.3)}$	$\frac{95.3(97.6)}{\mathbf{98.8}(93)}$	$\frac{95.7(92.6)}{93.9(92.9)}$
MERS	$\frac{93.1(92)}{94.2(92.5)}$	$\frac{12.9(66)}{71.4(47.7)}$	$\frac{93.3(95)}{94.2(93.2)}$	$\frac{93.1(95)}{94.2(93.1)}$	$\frac{93.2(94)}{94.1(93.3)}$	$\frac{93.1(95)}{94.2(93.2)}$	$\frac{92.7(92.3)}{93.6(93.5)}$	$\frac{92.8(95)}{94.2(93.4)}$
HIV	$\frac{93.4(94.3)}{93.5(93.8)}$	$\frac{12.6(96.6)}{69.5(44)}$	$\frac{93.4(97.6)}{93.5(93.7)}$	$\frac{93.5(97.3)}{93.5(93.4)}$	$\frac{93.5(95.3)}{93.5(93.2)}$	$\frac{93.4(94.6)}{93.5(93.6)}$	$\frac{93(99)}{93.4(92.5)}$	$\frac{93.4(95.3)}{93.5(93.8)}$
Hepaci	$\frac{92.6(90.6)}{92.6(97.1)}$	$\frac{13.4(65)}{76.5(88.6)}$	$\frac{92.6(91.6)}{92.7(95.9)}$	$\frac{92.6(92.3)}{92.7(94.1)}$	$\frac{92.6(92.3)}{93(96.9)}$	$\frac{92.6(95.3)}{92.7(97.8)}$	$\frac{91.8(93.3)}{93.3(97.3)}$	$\frac{92.6(91.3)}{92.7(97.6)}$
Rhino	$\frac{93.2(92.6)}{93.2(92.6)}$	$\frac{12.8(69.3)}{87.7(47.1)}$	$\frac{93.2(94)}{93.6(92.9)}$	$\frac{93.2(94)}{93.4(93.1)}$	$\frac{93.2(93)}{93.8(92.6)}$	$\frac{93.2(92.6)}{93.2(93.1)}$	$\frac{93.3(93.3)}{94.8(91.9)}$	$\frac{93.2(92.6)}{93.2(92.9)}$
Influenza	$\frac{93.7(92.3)}{93(93.6)}$	$\frac{12.3(96.6)}{59.8(52.2)}$	$\frac{93.7(97.6)}{93(93.8)}$	$\frac{93.3(97.3)}{93(93.8)}$	$\frac{93.7(94.3)}{93(93.5)}$	$\frac{93.7(94.6)}{93(93.8)}$	$\frac{93.5(99)}{92.6(92.6)}$	$\frac{93.7(95.3)}{93(93.6)}$
Noro	$\frac{93(92.3)}{92.8(93.1)}$	$\frac{13(69.3)}{86.9(35.1)}$	$\frac{93(92.6)}{93.5(94.1)}$	$\frac{93(94)}{92.8(93.4)}$	$\frac{93(93)}{94(94.5)}$	$\frac{93(93.3)}{92.8(93.7)}$	$\frac{92.7(92.6)}{94.2(93.9)}$	$\frac{93(92.6)}{92.8(94)}$
Rota	$\frac{94.2(92)}{94.2(98.6)}$	$\frac{11.8(76.3)}{73.9(56.8)}$	$\frac{94.2(92.3)}{94.8(\mathbf{99.9})}$	$\frac{94.2(92.3)}{94.3(98.4)}$	$\frac{92.2(91.6)}{93.5(\mathbf{99.9})}$	$\frac{94.2(92)}{94(\mathbf{99.9})}$	$\frac{94.1(91.6)}{95.3(99.6)}$	$\frac{94.2(92)}{94.3(\mathbf{99.9})}$
Measles	$\frac{93.4(95)}{93.5(94.4)}$	$\frac{12.6(87.6)}{67.3(30.4)}$	$\frac{93.4(99.6)}{93.5(94.1)}$	$\frac{93.4(\mathbf{99.9})}{93.5(93.9)}$	$\frac{93.4(96.6)}{93.2(94.4)}$	$\frac{93.4(97.3)}{93.5(93.9)}$	$\frac{90.3(\mathbf{99.9})}{94.2(94)}$	$\frac{93.4(97.6)}{93.5(94)}$
Rabies	$\frac{93.3(92.6)}{93(93.2)}$	$\frac{12.7(69.3)}{87.7(32.2)}$	$\frac{93.3(94)}{93.6(93.4)}$	$\frac{93.3(94)}{93.2(93.5)}$	$\frac{93.3(94)}{94.3(93.4)}$	$\frac{93.3(92.6)}{93.1(93.5)}$	$\frac{91.5(93.3)}{95.7(93.2)}$	$\frac{93.3(92.6)}{93(93.6)}$
West Nile	$\frac{93.5(96.3)}{93.7(92.9)}$	$\frac{12.6.4(96)}{86.8(50.8)}$	$\frac{93.5(97)}{93.9(94.1)}$	$\frac{93.5(99)}{93.8(93.7)}$	$\frac{93.5(96.3)}{94.7(93.5)}$	$\frac{93.5(98)}{93.6(93.7)}$	$\frac{93.3(99.6)}{95.1(93.3)}$	$\frac{93.5(97.3)}{93.6(93.8)}$
Average	$\frac{93.5(92.9)}{93.3(93.6)}$	$\frac{18.4(78)}{78(47.2)}$	$\frac{93.6(95)}{93.6(94.1)}$	$\frac{93.62(95.3)}{93.52(93.7)}$	$\frac{93.3(94.1)}{93.6(94.1)}$	$\frac{93.8(94.4)}{93.41(94.1)}$	$\frac{93.2(94.9)}{94.4(93.8)}$	$\frac{93.7(93.8)}{93.3(94.2)}$
MC	$\frac{20.1(25)}{17.9(49.6)}$	$\frac{22.7(40.3)}{18.8(31.2)}$	$\frac{19.4(36.6)}{15.4(24)}$	$\frac{23.8(44)}{24(39.8)}$	$\frac{25.8(48)}{26.2(\mathbf{51.5})}$	$\frac{\mathbf{26.3}(\mathbf{53})}{22(51)}$	$\frac{22.1(51)}{\mathbf{30.7}(39.8)}$	$\frac{24.7(47.6)}{21(50.1)}$

MNB: Multinomial Naive Bayes, GNB: Gaussian Naive Bayes, DT: Decision Tree, RF: Random Forest, MLP: Multilayer Perceptron, SVM: Support Vector Machine, kNN: k-Nearest Neighbors, LR: Logistic Regression, MC: Multi-class

Table 6: Classification results for SVM.

	ACC	P	R	F1	AUC
Dengue	$\frac{93.6(95)}{93(93.7)}$	$\frac{87.6(90.2)}{93.2(87.8)}$	$\frac{93.3(95)}{93(93.7)}$	$\frac{90.5(92.5)}{90.3(90.6)}$	$\frac{0.50(0.50)}{0.51(0.50)}$
Dabie	$\frac{93(93.3)}{93(94)}$	$\frac{86.5(87.1)}{93.1(93.6)}$	$\frac{93.0(93.3)}{93(94)}$	$\frac{89.6(95.1)}{91.5(92.9)}$	$\frac{0.50(0.50)}{0.53(0.58)}$
Hanta	$\frac{92.7(93.3)}{92.7(92.6)}$	$\frac{85.5(87.1)}{93.5(91.5)}$	$\frac{92.3(93.3)}{93.1(92.6)}$	$\frac{88.9(90.1)}{90.1(89.5)}$	$\frac{0.50(0.50)}{0.53(0.51)}$
SC2	$\frac{99.9(96.6)}{93.1(93.3)}$	$\frac{99.8(96.6)}{86.8(92.7)}$	$\frac{99.9(96.6)}{93.1(93.3)}$	$\frac{99.9(96.6)}{89.9(91.1)}$	$\frac{0.99(0.87)}{0.50(0.52)}$
Ebola	$\frac{95.7(92.6)}{95.8(92.3)}$	$\frac{95.3(91.8)}{95.9(85.95)}$	$\frac{95.7(92.6)}{95.8(92.3)}$	$\frac{95.2(90.9)}{94.8(98.1)}$	$\frac{0.75(0.61)}{0.66(0.50)}$
MERS	$\frac{93.1(95)}{94.2(93.2)}$	$\frac{86.8(90.2)}{88.8(92.1)}$	$\frac{93.1(95)}{94.2(93.2)}$	$\frac{89.9(92.5)}{91.5(90.9)}$	$\frac{0.50(0.50)}{0.50(0.51)}$
HIV	$\frac{93.4(94.6)}{93.5(93.6)}$	$\frac{87.3(96.5)}{87.5(93.2)}$	$\frac{93.4(94.6)}{93.5(93.6)}$	$\frac{90.3(94.5)}{90.4(91.1)}$	$\frac{0.51(0.58)}{0.50(0.53)}$
Hepaci	$\frac{92.6(95.3)}{92.7(97.8)}$	$\frac{85.8(86.9)}{93.2(97.74)}$	$\frac{92.6(95.3)}{92.7(97.80)}$	$\frac{89.1(90.9)}{89.6(97.67)}$	$\frac{0.50(0.50)}{0.51(0.90)}$
Rhino	$\frac{93.2(92.6)}{93.2(93.1)}$	$\frac{86.9(85.2)}{93.9(91.3)}$	$\frac{93.2(92.6)}{93.2(93.1)}$	$\frac{89.9(88.7)}{91.7(89.9)}$	$\frac{0.50(0.52)}{0.54(0.50)}$
Influenza	$\frac{93.7(94.6)}{93(93.8)}$	$\frac{87.9(89.9)}{93.5(91.8)}$	$\frac{93.7(94.6)}{93(93.8)}$	$\frac{90.7(92.1)}{89.8(91.2)}$	$\frac{0.50(0.50)}{0.51(0,52)}$
Noro	$\frac{93(93.3)}{92.8(93.7)}$	$\frac{86.6(87.1)}{93.1(92.8)}$	$\frac{93(93.3)}{92.8(93.7)}$	$\frac{89.7(90.1)}{90.7(92.1)}$	$\frac{0.50(0.50)}{0.52(0.55)}$
Rota	$\frac{94.2(92)}{94(99.9)}$	$\frac{88.7(84.65)}{95.1(99.4)}$	$\frac{94.2(92)}{94(98.9)}$	$\frac{91.3(88.1)}{92.9(98.5)}$	$\frac{0.50(0.51)}{0.58(0.97)}$
Measles	$\frac{93.4(97.3)}{93.5(93.9)}$	$\frac{87.2(97.3)}{93.7(92.9)}$	$\frac{93.4(97.3)}{93.7(93.9)}$	$\frac{90.2(97.3)}{91.4(91.5)}$	$\frac{0.59(0.93)}{0.52(0.52)}$
Rabies	$\frac{93.3(92.6)}{93.1(93.5)}$	$\frac{87.1(93)}{93.5(91.5)}$	$\frac{93.3(92.6)}{93.1(93.5)}$	$\frac{90.1(90.1)}{92.2(91.2)}$	$\frac{0.50(0.52)}{0.58(0.54)}$
West Nile	$\frac{93.5(98)}{93.6(93.7)}$	$\frac{87.4(98)}{93.5(91.7)}$	$\frac{93.5(98)}{93.6(93.7)}$	$\frac{90.3(97.9)}{92.4(91.5)}$	$\frac{0.50(0.82)}{0.530(0.53)}$
MC	$\frac{26.3(53)}{26.2(51)}$	$\frac{28.9(49.2)}{41.3(53.1)}$	$\frac{26.3(45.3)}{35(52.4)}$	$\frac{23.4(44.4)}{36.6(52.3)}$	$\frac{0.15(0.21)}{0.17(0.31)}$

ACC: Accuracy, P: Precision, R: Recall, F1: F-measure, AUC: Area Under Curve, MC: Multi-class

SearchCV were similar, with a negligible difference, to the one where classifiers were used with hyperparameters mentioned at the start of this section.

t-SNE [51] was utilized to visualize the frequent sequential patterns discovered in the genome sequences of five different viruses. Figure 4 illustrates the distribution of frequent sequential patterns across the coding regions of the selected viruses. Notably, while the visualization shows distinct clusters for each virus, it also highlights areas of overlap between certain viruses, such as SC2 and Ebola, as well as Hanta and MERS. These overlaps suggest that some frequent sequential patterns are shared among these viruses, which could indicate evolutionary relationships or common functional motifs within specific genomic regions. Investigating and understanding these overlaps is important as they offer insights into the genetic similarities and differences that might not be immediately apparent through traditional classification methods.

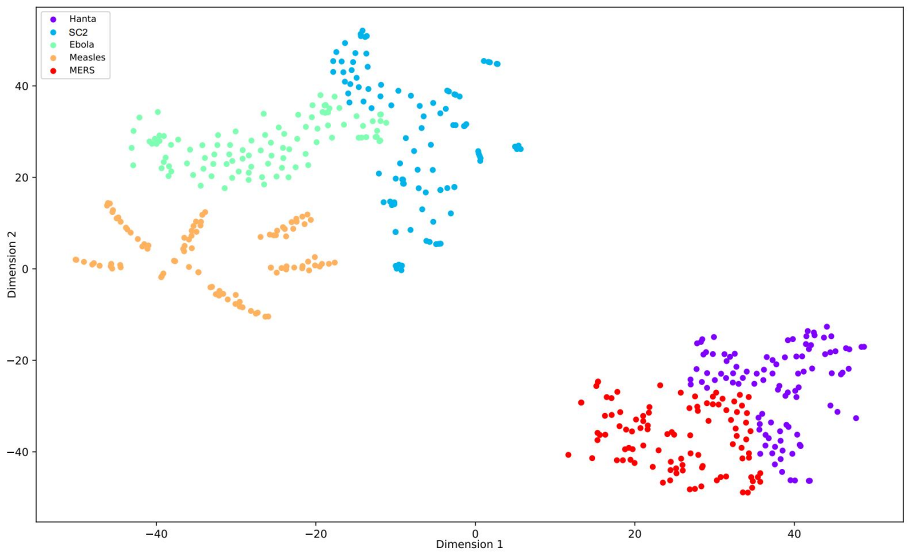


Fig. 4: Visualization of the frequent sequential pattern distribution discovered in the coding region form of five viruses. The five clusters are separable from each other with a minor amount of similarity among a few viruses. SC2 stands for SARS-CoV-2.

From the perspective of classification, the discovery of overlapping patterns may have several implications. Firstly, the existence of shared patterns can enhance classifier robustness by acting as redundant yet informative features. However, it is crucial that these features possess the discriminatory power to effectively distinguish closely related viruses. Secondly, the presence of overlaps underscores the need for classifiers that are highly sensitive to minute differences in genomic sequences, potentially necessitating the development of more intricate models that incorporate not only the existence of shared patterns but also their contextual occurrences. Thirdly, the overlapping patterns may influence the curation of the training data. By recognizing which patterns are shared among viruses, one can focus on enhancing the diversity of training samples, ensuring that classifiers are exposed to a wide range of genomic variations. In summary, the overlaps observed do not diminish the effectiveness of the

GenoAnaCla approach but rather enrich our understanding of viral relationships and highlight the need to further investigate the importance of similar patterns discovered in different viruses in the classification process and using advanced classification techniques that can navigate the complexities of genomic data.

GenoAnaCla has many practical and clinical relevances. For example, it can facilitate the early detection of harmful viruses, which is critical for infectious disease management and for public health preparedness and response in infectious disease management. Furthermore, the ability to analyze viral genomes more effectively can enhance epidemiological studies, aiding researchers in tracking virus mutations and spread patterns.

4.3 Comparison

Table 7 compares the performance of GenoAnaCla with some of the state-of-the-art approaches (SOTA) for the classification/detection of genomic data. The studies [14,15,16,18,19,20,23,34], [9,10,11,12,26] and [13,27,28,29,30] evaluated their respective approaches using datasets comprising genome sequences, metagenomic sequences and S protein sequences, respectively. The SVM classifier in the proposed GenoAnaCla performed similarly to kNN [14] (99.8% ACC) and performed better than XGBoost [11](97.8%), CNN [18] (99% ACC), SVM [34] (97% ACC), kNN [10] (98.6% ACC), RF [15] (93% ACC), kNN [16] (98.4%), CNN [19] (93.16% ACC), CNN [20] (98.73% ACC), CNN [26] (92% ACC) and RF [23] (97.47% ACC). The studies [22,25] obtained 100% accuracy; however, because their dataset comprised less genome sequences, their results are not included in Table 7. The highest SVM results obtained in GenoAnaCla are included in Table 7 as it demonstrated better overall performance in both types of classification when compared to other models. Note that the average results of XGBoost [11] in Table 7 is for the three-class (infectious virus, endogenous retrovirus (ERV) and non-ERV human) prediction.

GenoAnaCla (SVM) also performed better than the classifiers used in [13,27,28,29,30]. Their results are omitted from Table 7 as they considered only the S protein. Compared to the majority of the previous approaches presented in Table 7, the feature extraction process in the proposed approach is much simpler and does not require very high computational power. GenoAnaCla (SVM) achieved an average accuracy improvement (AAI) of 3.18%. AAI is defined as:

$$AAI = \frac{\sum_{i=1}^{n} \frac{\mathrm{NA}-\mathrm{OA}_i}{\mathrm{OA}_i}}{n} \tag{9}$$

where NA, and OA represent the new and old accuracy, respectively, and n is the total number of previous works. According to the data presented in Table 7, a total of 11 studies have provided accuracy results, therefore, $n = 11$.

The proposed approach is not specific to a particular viral family and can be used to classify/predict DNA viruses and metagenomic contigs. Previous approaches used various training:testing ratios or k-fold cross-validation and the ratios that produced the best results for the classifiers were selected. Similarly, we listed the best results of the classifiers that we obtained using a training:testing ratio of 80:20.

Table 7: Comparison of the GenoAnaCla with SOTA approaches.

Reference	**Model**	**Features**	**Results**				
			ACC	P	R	F1	AUC
[9]	Attn.-based LSTM	Seq2Vec+max pooling	–	91.14%	92.6%	91.86%	–
[10]	kNN	*k-mer*+bag of words	98.6%	98.5%	98.6%	–	–
[11]	XGBoost	*k-mer*+recoding system	97.8%	96.7%	96.8%	96.7%	–
[12]	LSTM based RNN-VirSeeker	softmax	–	92.11%	86.40%	89.16%	0.917
[14]	kNN	CpG+Similarity	99.8%	99.7%	99.9% R	99.8%	0.997
[15]	RF	CpG based	93%	93%	93%	93%	–
[16]	kNN	CpG+kNN with L_1 type distance metric	98.4%	98.4%	99.2%	98.8%	–
[18]	CNN	max pooling	99%	–	99%	–	–
[19]	CNN	*k-mer*	93.16%	90%	98%	94%	–
[20]	CNN	max pooling	98.73%	–	–	–	–
[23]	RF	biomarkers, obtained with three-base periodicity property	97.47%	–	96.29%	–	–
[26]	CNN	max and average pooling	92%	–	32%	–	0.923
[34]	SVM	nucleotides	97%	96%	77%	96%	–
GenoAnaCla	SVM	Frequent sequential patterns of nucleotides/codons/amino acids	99.9%	99.4%	98.9%	98.5%	0.97

LSTM: Long Short-Term Memory, kNN: k-nearest neighbors, XGBoost: Extreme Gradient Boosting, RNN: Recurrent Neural Network, RF: Random Forest, CNN: Convolutional Neural Network, SVM: Support Vector Machine, ACC: Accuracy, P: Precision, R: Recall, F1: F-measure, AUC: Area Under Curve.

5 Conclusion

A generic approach was proposed for the analysis and classification of genome sequences from diverse RNA viruses downloaded from NCBI's GenBank in three forms: *NF*, *CRF*, and *PF*. Genome sequences were initially transformed into a suitable format, and subsequently, algorithms for SPM were applied to discover frequent bases, codons, and amino acids, their frequent sequential patterns, and rules. Obtained frequent patterns of nucleotides, codons, and amino acids were then used for classification. Eight classifiers (LR, RF, kNN, GNB, SVM, MLP, DT, MNB) were employed, and their performance was assessed using five evaluation metrics, including accuracy, precision, recall, F1-score and AUC. The obtained findings suggest that SVM outperformed the other seven classifiers in binary classification. Moreover, it achieved better results than existing genome sequence classification approaches, achieving an average accuracy improvement of more than 3%. Compared to binary classification, classifiers performance was low for multi-class classification. The key insight from the findings is that utilizing limited frequent nucleotide, codon, and amino acid patterns can lead to improved prediction and classification, instead of providing the complete nucleotides, codons, or amino acids present in the sequences. The proposed approach is not limited to RNA viruses and can be used on DNA viruses, metagenomic data, and even human DNA.

Some future research directions are: (1) Exploring more advanced or specialized classifiers to investigate if they can leverage the sequential data and the shared frequent patterns among many viruses more effectively than conventional classifiers. (2) Extending the proposed approach to extract frequent sequential *k-mers* and using them as features in the classification process. (3) Using emerging, contrast pattern mining [52] or negative pattern mining [53,54] on the genome sequences to find contrasting or negative frequent patterns of bases, codons, and amino acids and employing them as features in the classification process. (4) Examining the potential for discriminative frequent patterns to be utilized more effectively, compared to similar frequent patterns, in the classification process, particularly in MC classification. (5) Using optimization and meta-heuristic algorithms [55] for the selection of optimal hyperparameters of classifiers.

CRediT author statement

M. Saqib Nawaz: Data Curation, Methodology, Validation, Visualization, Writing - Original Draft, Writing - Review & Editing. **M. Zohaib Nawaz**: Data Curation, Validation, Investigation, Visualization, Writing - Review & Editing. **Zhang Junyi**: Data Curation, Conceptualization, Formal analysis. **Philippe Fournier-Viger**: Supervision, Methodology, Validation, Writing - Review & Editing. **Jun-Feng Qu**: Investigation, Visualization, Writing - Review & Editing

Conflict of Interest: Authors declare no conflict on interest.

Funding: Authors did not receive funding for this work

References

1. E. W. Sayers, M. Cavanaugh, K. Clark, J. Ostell, K. D. Pruitt, I. Karsch-Mizrachi, Genbank, Nucleic Acids Research 48 (2020) D84–D86. doi:10.1093/nar/gkz956.
2. C.-N. Members, Partners, Database resources of the national genomics data center, China national center for bioinformation in 2023, Nucleic Acids Research 51 (D1) (2023) D18–D28. doi:10.1093/nar/gkac1073.
3. K. Kalia, G. Saberwal, G. Sharma, The lag in SARS-CoV-2 genome submissions to GISAID, Nature Biotechnology 39 (2021) 1058–1060. doi:10.1038/s41587-021-01040-0.
4. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, D. J. Lipman, Basic local alignment search tool, Journal of Molecular Biology 215 (3) (1990) 403–410. doi:10.1016/S0022-2836(05)80360-2.
5. W. R. Pearson, Using the FASTA program to search protein and DNA sequence databases, Methods in Molecular Biology 24 (1994) 307–331. doi:10.1385/0-89603-246-9:307.
6. A. Zielezinski, S. Vinga, J. Almeida, W. M. Karlowski, Alignment-free sequence comparison: Benefits, applications, and tools, Genome Biology 18 (2017) 1–17. doi:10.1186/s13059-017-1319-7.
7. S. Vinga, Information theory applications for biological sequence analysis, Briefings in Bioinformatics 15 (3) (2013) 376–389. doi:10.1093/bib/bbt068.
8. S. H. Ye, K. J. Siddle, D. J. Park, P. C. Sabeti, Benchmarking metagenomics tools for taxonomic classification, Cell 178 (2019) 779–794. doi:10.1016/j.cell.2019.07.010.
9. Y. Miao, F. Liu, T. Hou, Y. Liu, Virtifier: a deep learning-based identifier for viral sequences from metagenomes, Bioinformatics 38 (5) (2021) 1216–1222. doi:10.1093/bioinformatics/btab845.
10. M. H. Alshayeji, S. C. Sindhu, S. Abed, Viral genome prediction from raw human DNA sequence samples by combining natural language processing and machine learning techniques, Expert Systems with Applications 218 (2023) 119641. doi:10.1016/j.eswa.2023.119641.
11. G. Liu, X. Chen, Y. Luan, D. Li, VirusPredictor: XGBoost-based software to predict virus-related sequences in human data, Bioinformatics 40 (4) (2024) btae192. doi:10.1093/bioinformatics/btae192.
12. F. Liu, Y. Miao, Y. Liu, T. Hou, Rnn-virseeker: A deep learning method for identification of short viral sequences from metagenomes, IEEE/ACM Transactions on Computational Biology and Bioinformatics 19 (3) (2022) 1840–1849. doi:10.1109/TCBB.2020.3044575.

13. S. Ali, B. Sahoo, N. Ullah, A. Zelikovskiy, M. Patterson, I. Khan, A k-mer based approach for SARS-COV-2 variant identification, in: International Symposium on Bioinformatics Research and Applications (ISBRA), 2021, pp. 153–164. doi:10.1007/978-3-030-91415-8_14.
14. H. Arslan, COVID-19 prediction based on genome similarity of human sars-cov-2 and bat sars-cov-like coronavirus, Computers & Industrial Engineering 161 (2021) 107666. doi:10.1016/j.cie.2021.107666.
15. H. Arslan, Machine learning methods for COVID-19 prediction using human genomic data, Proceedings 74 (1) (2021) 20. doi:10.3390/proceedings2021074020.
16. H. Arslan, H. Arslan, A new COVID-19 detection method from human genome sequences using cpg island features and knn classifier, Engineering Science and Technology, an International Journal 24 (4) (2021) 839–847. doi:10.1016/j.jestch.2020.12.026.
17. G. S. Dlamini, S. J. Muller, R. L. Meraba, R. A. Young, J. Mashiyane, T. Chiwewe, D. S. Mapiye, Classification of COVID-19 and other pathogenic sequences: A dinucleotide frequency and machine learning approach, IEEE Access 8 (2021) 195263–195273. doi:10.1109/ACCESS.2020.3031387.
18. M. A. El-Dosuky, M. Soliman, A. E. Hassanien, COVID-19 vs Influenza viruses: A cockroach optimized deep neural network classification approach, International Journal of Imaging Systems and Technology 31 (2021) 471–482. doi:10.1002/ima.22562.
19. H. Gunasekaran, Ramalakshmi, R. M. Arokiaraj, D. Kanmani, C. Venkatesan, C. S. G. Dhas, Analysis of DNA sequence classification using CNN and hybrid models, Computational and Mathematical Methods in Medicine 1835056 (2021). doi:10.1155/2021/1835056.
20. A. Lopez-Rincon, A. Tonda, L. Mendoza-Maldonado, D. G. J. C. Mulders, R. Molenkamp, C. A. Perez-Romero, E. Claassen, J. Garssen, A. D. Kraneveld, Classification and specific primer design for accurate detection of SARS-CoV-2 using deep learning, Scientific Reports 11 (2021) 947. doi:10.1038/s41598-020-80363-5.
21. P. A. Mateos, R. F. Balboa, S. Easteal, E. Eyras, H. R. Patel, PACIFIC: A lightweight deep-learning classifier of SARS-CoV-2 and co-infecting RNA viruses, Scientific Reports 11 (2021) 3209. doi:10.1038/s41598-021-82043-4.
22. S. M. Naeem, M. S. Mabrouk, S. Y. Marzouk, M. A. Eldosoky, A diagnostic genomic signal processing (GSP)-based system for automatic feature analysis and detection of COVID-19, Briefings in Bioinformatics 2 (2) (2021) 1197–1205. doi:10.1093/bib/bbaa170.
23. O. P. Singh, M. Vallejo, I. M. El-Badawy, A. Aysha, J. Madhanagopal, A. A. M. Faudzi, Classification of SARS-CoV-2 and non-SARS-CoV-2 using machine learning algorithms, Computers in Biology and Medicine 136 (2021) 104650. doi:10.1016/j.compbiomed.2021.104650.
24. R. Jing, Y. Li, L. Xue, F. Liu, M. Li, J. Luo, autoBioSeqpy: A deep learning tool for the classification of biological sequences, Journal of Chemical Information and Modeling 60 (8) (2020) 3755–3764. doi:10.1021/acs.jcim.0c00409.
25. G. S. Randhawa, M. P. M. Soltysiak, H. E. Roz, C. P. E. de Souza, K. A. Hill, L. Kari, Machine learning using intrinsic genomic signatures for rapid classification of novel pathogens: COVID-19 case study, PLoS One 15 (4) (2020) e0232391. doi:10.1371/journal.pone.0232391.
26. A. Tampuu, Z. Bzhalava, J. Dillner, R. Vicente1, Viraminer: Deep learning on raw dna sequences for identifying viral genomes in human samples, PLoS ONE 14 (9) (2019) e0222271. doi:10.1371/journal.pone.0222271.
27. K. Kuzmin, A. E. Adeniyi, A. K. D. Jr., D. Lim, H. Nguyen, N. R. Molina, L. Xiong, I. T. Weber, R. W. Harrison, Machine learning methods accurately predict host specificity of coronaviruses based on spike sequences alone, Biochemical and Biophysical Research Communications 533 (3) (2020) 553–558. doi:10.1016/j.bbrc.2020.09.010.
28. X.-L. Qiang, P. Xu, G. Fang, W.-B. Liu, Z. Kou, Using the spike protein feature to predict infection risk and monitor the evolutionary dynamic of coronavirus, Infectious Diseases of Poverty 9 (2020) 33. doi:10.1186/s40249-020-00649-8.
29. M. S. Nawaz, P. Fournier-Viger, Y. He, S-PDB: Analysis and classification of SARS-CoV-2 spike protein structures., in: Proceedings of International Conference on Bioinformatics and Biomedicine (BIBM), 2022, pp. 2259–2265. doi:10.1109/BIBM55620.2022.9995562.
30. M. S. Nawaz, P. Fournier-Viger, Y. He, Q. Zhang, PSAC-PDB: Analysis and classification of protein structures, Computers in Biology and Medicine 158 (2023) 106814. doi:10.1016/j.compbiomed.2023.106814.
31. M. S. Nawaz, P. Fournier-Viger, S. Nawaz, H. Zhu, U. Yun, SPM4GAC: SPM based approach for genome analysis and classification of macromolecules, International Journal of Biological Macromolecules 130984 (2024). doi:10.1016/j.ijbiomac.2024.130984.

32. M. S. Nawaz, P. Fournier-Viger, S. Nawaz, W. Gan, Y. He, FSP4HSP: Frequent sequential patterns for the improved classification of heat shock proteins, their families, and sub-types, International Journal of Biological Macromolecules 277 (2024) 134147. doi:10.1016/j.ijbiomac.2024.134147.
33. P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, R. Thomas, A survey of sequential pattern mining, Data Science and Pattern Recognition 1 (2017) 54–77.
34. I. Ahmed, G. Jeon, Enabling artificial intelligence for genome sequence analysis of COVID-19 and alike viruses, Interdisciplinary Sciences: Computational Life Sciences 14 (2022) 504–519. doi:10.1007/s12539-021-00465-0.
35. M. S. Nawaz, P. Fournier-Viger, A. Shojaee, H. Fujita, Using artificial intelligence techniques for covid-19 genome analysis, Applied Intelligence 53 (2021) 3086–3103. doi:10.1007/s10489-021-02193-w.
36. M. S. Nawaz, P. Fournier-Viger, M. Aslam, W. Li, Y. He, X. Niu, Using alignment-free and pattern mining methods for SARS-CoV-2 genome analysis, Applied Intelligence 53 (2023) 21920–21943. doi:10.1007/s10489-023-04618-0.
37. S. Dubey, D. K. Verma, M. Kumar, Severe acute respiratory syndrome coronavirus-2 genoanalyzer and mutagenic anomaly detector using FCFMI and ncse, International Journal of Biological Macromolecules 258 (2024) 129051. doi:10.1016/j.ijbiomac.2023.129051.
38. M. Tandan, Y. Acharya, S. Pokharel, M. Timilsina, Discovering symptom patterns of covid-19 patients using association rule mining, Computers in Biology and Medicine 131 (2021) 104249. doi:10.1016/j.compbiomed.2021.104249.
39. I. Acer, F. Bulucu, S. Icer, F. Latifogul, Early diagnosis of pancreatic cancer by machine learning methods using urine biomarker combinations, Turkish Journal of Electrical Engineering and Computer Sciences 31 (1) (2023) 112–125. doi:10.55730/1300-0632.3974.
40. P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, H. T. Lam, The SPMF open-source data mining library version 2, in: Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), 2016, pp. 36–40. doi:10.1007/978-3-319-46131-1_8.
41. C. C. Aggarwal, J. Han, *Frequent Pattern Mining*, *Springer* (2014). doi:10.1007/978-3-319-07821-2.
42. A. Alaiad, H. Najadat, B. Mohsen, K. Balhaf, Classification and association rule mining technique for predicting chronic kidney disease, Journal of Information & Knowledge Management 19 (01) (2020) 2040015. doi:10.1142/S0219649220400158.
43. R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: Proceedings of Very Large Databases (VLDB), 1994, pp. 487–499.
44. J. Yu, L. Zhang, N. Xu, L. Fa, K. Yang, Application of constraint-based frequent closed itemsets mining in tcm clinical data analysis, in: Proceedings of International Conference on Bioinformatics and Biomedicine (BIBM), pp. 4689–4696. doi:10.1109/BIBM58861.2023.10385654.
45. M. J. Zaki, C. Hsiao, CHARM: An efficient algorithm for closed itemset mining, in: Proceedings of the SIAM International Conference on Data Mining (SDM), 2002, pp. 457–473. doi:10.1137/1.9781611972726.27.
46. P. Fournier-Viger, A. Gomariz, M. Campos, R. Thomas, Fast vertical mining of sequential patterns using co-occurrence information, in: Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), 2014, pp. 40–52. doi:10.1007/978-3-319-06608-0_4.
47. J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, in: Proceedings of Knowledge Discovery and Delivery (KDD), 2002, pp. 429–435. doi:10.1145/775047.77510.
48. P. Fournier-Viger, T. Gueniche, S. Zida, V. Tseng, Erminer: sequential rule mining using equivalence classes, in: Proceedings of International Symposium on Intelligent Data Analysis (IDA), 2014, pp. 108–119. doi:10.1007/978-3-319-12571-8_10.
49. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, Édouard Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (85) (2011) 2825–2830.
50. M. L. McHugh, Multiple comparison analysis testing in anova, Biochemia Medica 21 (2011) 21920–21943. doi:10.1007/s10489-023-04618-0.
51. L. van der Maaten, G. Hinton, Visualizing data using t-SNE, Journal of Machine Learning Research 9 (2008) 2657–2605.

52. S. Ventura, J. Luna, *Supervised Descriptive Pattern Mining*, *Springer* (2018). doi:10.1007/978-3-319-98140-6.
53. F. Cui, X. Ren, X. Dong, Mining interesting negative sequential patterns based on influence, IEEE Access 11 (2023) 12925–12936. doi:10.1109/ACCESS.2023.3242327.
54. C. Sun, Y. Gong, Y. Guo, L. Zhao, H. Guan, X. Liu, X. Dong, SN-RNSP: Mining self-adaptive nonoverlapping repetitive negative sequential patterns in transaction sequences, Knowl. Based Syst. 287 (2024) 111449. doi:10.1016/j.knosys.2024.111449.
55. V. Kumar, R. Naresh, V. Sharma, V. Kumar, State-of-the-Art Optimization and Metaheuristic Algorithms, John Wiley & Sons, Ltd, 2022, Ch. 25, pp. 509–536. doi:doi.org/10.1002/9781119792642.ch25.