Preprint of:
Nawaz, M. S., Fournier-Viger, P., Nawaz, S., Gan, W., He, Y. (2024). FSP4HSP: Frequent sequential patterns for the improved classification of heat shock proteins, their families, and sub-types. International Journal of Biological Macromolecules (BIOMAC). Elsevier, Volume 277, Part 1, October 2024, article 134147.

FSP4HSP: Frequent sequential patterns for the improved classification of heat shock proteins, their families, and sub-types

M. Saqib Nawaz[1] · **Philippe Fournier-Viger**[1,*] · **Shoaib Nawaz**[2] · **Wensheng Gan**[3] · **Yulin He**[4]

Date of receiving / date of acceptance

Abstract Heat shock proteins (HSPs) from different families and sub-types play a vital role in the folding and unfolding of proteins, in maintaining cellular health, and in preventing serious disorders. Previous computational methods for HSP classification have yielded promising performance. However, most of the existing methods rely on amino acid composition features and still face challenges related to interpretability and accuracy. To overcome these issues, we introduce a novel frequent sequential pattern-based analysis and classification method, called FSP4HSP (Frequent Sequential Patterns for HSPs), for the classification of HSPs, their families, and sub-types. The proposed model finds FSPs of amino acids (FSPAAs) and uses them for analysis and classification. Besides FSPAAs, sequential rules among amino acids are also discovered. Both binary and multi-class classification scenarios are considered, with the utilization of eight integer-based and four string-based classifiers. The use of FSPAAs in the classification/prediction task enhances the interpretability of FSP4HSPC and a performance comparison using various evaluation measures demonstrates that it surpasses existing methods for the classification/recognition of HSPs.

Keywords Heat shock protein · Protein sequence · Frequent sequential patterns · Sequential pattern mining · Classification

1 Introduction

Heat shock proteins (HSPs), which are present in almost all living organisms, are molecular chaperones that are produced within cells in reaction to a range of envi-

[1]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China E-mail: *msaqibnawaz@szu.edu.cn, philfv@szu.edu.cn*
[2]Department of Pharmacy, The University of Lahore, Sargodha Campus, Pakistan E-mail: *shoaib.nawaz@pharm.uol.edu.pk*
[3] College of Cyber Security, Jinan University, Guangzhou 510632, China E-mail: wsgan001@gmail.com
[4]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), Shenzhen China E-mail: *yulinhe@gml.ac.cn*
[*] *Corresponding author*

ronmental (chemical, biological, and physical) factors [1]. They are also referred to as "stress proteins" since they can be produced by heavy metals, toxins, oxidants, viruses, and free radicals. HSPs play a critical role in maintaining protein homeostasis within each cell. They help in the proper folding of partially folded or denatured proteins, ensuring that these molecules attain their correct three-dimensional conformation. Moreover, HSPs actively prevent the irreversible aggregation of damaged proteins, thereby safeguarding a cell from the deleterious consequences of protein misfolding and aggregation [2]. Moreover, they participate in a myriad of cellular functions, including modulation of their own synthesis [3], Ribonucleic acid (RNA) processing [4], and participation in signal transduction pathways [5]. The wide range of tasks carried out by HSPs underscores their importance in maintaining cellular health and preventing serious disorders. Notably, the accumulation of incorrectly folded proteins, a process that HSPs strive to inhibit, is implicated in various neurodegenerative diseases, such as Parkinson's and Alzheimer's disease [6–9]. Additionally, HSP dysfunction has been linked to cancer [10] and cardiovascular diseases [11].

HSPs are categorized into six primary families according to their core functions molecular weight [12, 13]: (1) HSP_{100}, (2) HSP_{90}, (3) HSP_{70}, (4) HSP_{60}, (5) HSP_{40}, and HSP_{20}. HSP_{20} are low-molecular-weight HSPs that are essential for inhibiting protein aggregation and promoting protein folding, particularly under stress conditions. HSP_{40} also known as J-domain proteins, acts as co-chaperones, regulating the activity of other HSP families, such as HSP70. J-proteins are further divided into four categories or types: $Type_1$ (T_1), $Type_2$ (T_2), $Type_3$ (T_3), and $Type_4$ (T_4). HSP_{60} are complexes of barrel-shape that help properly fold newly synthesised or misfolded proteins by offering an isolated environment for protein folding. The highly conserved HSP_{70} family is essential for protein folding, assembly, transport, and degradation, ensuring proteostasis throughout a cell. HSP_{90} mediates the maturation and activation of various client proteins, including kinases, transcription factors, and steroid receptors, making them essential for cellular signaling and regulation. HSP_{100} are large HSPs that are specialized in disaggregating and solubilizing protein aggregates, providing a crucial line of defense against the toxic effects of misfolded protein accumulation [14]. Due to the large number of cellular functions that HSPs perform, identifying and classifying them into distinct families and sub-types is an important and challenging task that has attracted considerable attention.

A reliable and robust classification of HSPs contributes to a thorough comprehension of their biological roles and their impacts on both health and disease. Conventional methods for HSP identification depend on experimental methods such as nuclear magnetic resonance (NMR) spectroscopy. However, this identification process is not only time-consuming and resource-intensive but also has high costs. To overcome these limitations, various computational methods were proposed [15–23] for the recognition/classification of HSP families and their sub-types. However, the majority of them depends on various sequence composition non-latent features vectors (such as pseudo amino acid composition (PAAC), split amino acid composition (SAAC), reduced amino acid alphabet (RAAA), dipeptide composition (DPC), spaced-DPC (SDPC), coupled amino acid (CAA), composition of k spaced amino acid pair (CKSAAP), G-spaced amino acid pair composition (GPC), etc. Extracting non-latent features requires expertise. Latent features, on the other hand, can be

generated automatically by using deep learning (DL) techniques, as done in [16, 23] for HSPs. These features are generally difficult to comprehend. Additionally, certain feature extraction techniques used in previous studies are expensive, and the model-learning time, particularly for DL-based models, is long. To our knowledge, the utilization of frequent sequential patterns (FSPs) as features for the identification/classification of HSPs has not been studied.

Sequential patterns [24] can reveal new and significant insights in sequential data and have been used for the analysis of genomics and proteomics data [25–30]. In this paper, we extract sequential patterns from HSP sequences in a novel general model, called FSP4HSP (Frequent Sequential Patterns for HSPs), for the improved classification of HSPs, their families, and sub-types. Two feature extraction methods are used in FSP4HSP. First, the whole protein sequences of HSPs are taken as features. Second, FSPs of amino acids (FSPAAs) are extracted from the protein sequences for the classification of HSPs. After feature extraction, eight integer-based and four string-based classifiers are trained, and experiments are carried out with a range of evaluation metrics to investigate the effectiveness and generalizability of the proposed HSP classification model. An evaluation of the proposed model on two datasets indicates that finding FSPAAs and subsequently utilizing them produces better classification performance than using all AAs in the whole HSP sequences. The performance of FSP4HSP is also compared with previous methods for the identification/classification of HSPs, and results show that it performs better than them.

This paper's remaining sections are arranged as follows: Section 2 discusses the previous computational methods for the classification/detection of HSPs. Section 3 provides the details for the datasets used in this study and the proposed FSP4HSP method. Section 4 discusses the obtained results and compares FSP4HSP with previous methods for HSP classification/detection. Lastly, Section 5 provides a conclusion with some remarks.

2 Related Work

In the last decade, various ML and DL-based methods have been developed to classify/identify the HSPs vs non-HSPs, HSP families, and their sub-types (Table 1). iHSP-PseRAAAC [17] used a support vector machine (SVM) on features obtained in HSP sequences by including the RAAA into the well-known PAAC [31]. The JPRED model [18] also used the tripeptide composition of RAAA to predict the four types of HSP_{40} (J-Protein). Amad et al. [19] used four classification algorithms ((1) K-nearest neighbor (kNN), (2) Probabilistic neural network (PNN), (3) SVM and (4) Multi-layer perceptron (MLP)) on features extracted in protein sequences by using three discrete methods ((1) SAAC, (2) PAAC, and (3) DPC). The obtained results showed that SVM performed better than the other three classifiers on the DPC feature space. The JPPRED method [21] identified J-protein types by discovering hybrid feature vectors obtained with the SAAC, PAAC, and position-specific scoring matrix (PSSM) that were fed to an ensemble learner (EL). The ES-PredHSP model [15] used the CKSAAP in HSPs as features and used six ML classifiers for the prediction of HSPs.

The PredHSP model [20] used the CAA compositions as feature vectors that were fed to two SVMs for the classification of HSPs vs Non-HSPs and HSP families. The ir-HSP method [22] used the GPC as feature vectors that were fed to SVM. That framework was used to predict HSPs, their families, as well as their sub-types (J-protein types). Recently, Min et al. [23] developed DeepHSP and DeeperHSP, based on convolutional neural networks (CNNs), for the classification of non-HSPs and HSP families. DeepHSP used one hot encoding (1HE) and convolutional layers, whereas DeeperHSP used a pre-trained protein language model (LM) along with a projection layer for improved performance. The MulCNN-HSP method [16] used multi-scale CNNs on protein sequences encoded with 1HE to classify HSPs. The early methods in [17–19] only focused on the classification of HSP families or their sub-types. The methods in [15, 16, 20, 22, 23] also take into account the non-HSP sequences as well.

Table 1: Summary of developed methods for the classification of HSPs

Method	Encdoing	Model(s)	Non-HSPs	HSP Families	J-Protein
iHSP-PseRAAAC [17]	RAAA+PAAC	SVM	No	Yes	No
JPRED [18]	RAAA	SVM	No	No	Yes
[19]	SAAC, PAAC, DPC	kNN, PNN, SVM, MLP	No	Yes	Yes
JPPRED [21]	SAAC, PAAC, PSSM	EL	No	No	Yes
PredHSP [20]	CAA	SVM	Yes	Yes	No
ir-HSP [22]	GPC	SVM	Yes	Yes	Yes
DeepHSP, DeeperHSP [23]	1HE, PTL	CNN	Yes	Yes	No
MulCNN-HSP [16]	1HE	CNNs	Yes	Yes	No
ES-PredHSP [15]	CKSAAP	LR, LDA, kNN, DT, SVM, NB	Yes	Yes	No

Five-fold cross-validation was used to evaluate MulCNN-HSP [16], DeepHSP, DeeperHSP, PredHSP, and irHSP. Ten-fold cross-validation was employed to evaluate JPPRED and ES-PredHSP. Whereas leave-one-out cross-validation was used to evaluate iHSP-PseRAAAC, JPRED and the model of [19]. The aforementioned methods performed well for the identification of HSPs, their families, and sub-types. However, they rely heavily on the AA sequence composition features. One-hot encoding can encode AAs independently by ignoring their location (order) but cannot capture high-level information. DL-based methods performed better than traditional ML-based methods as they use latent features. However, latent features are difficult to comprehend. The majority of the methods lack interpretability that is important to understand the underlying biological mechanisms and to use the methods in real-world applications. In contrast to previous research, this study focuses on finding sequential frequent patterns in HSP sequences and using them for reliable classification/detection tasks.

3 Materials and Methods

3.1 Datasets

We considered two datasets: (1) The first dataset HSP contains the HSP and Non-HSP sequences [16, 20, 22, 23]. (2) The second dataset HSP_{40} contains the J-protein types [18, 19, 21]. In HSP, the sequences of non-HSPs from SwissProt [32] were selected at random without reference to homologous proteins. HSP sequences were derived from HSPIR [12], which served as the source of sequences for HSPs. Moreover, CD-HIT [33] was used to remove the protein sequences in the same family with pairwise sequence similarity $\geq 40\%$. The final dataset, formulated by equation 1 was obtained by filtering out those HSP and non-HSP sequences that contained non-standard AAs.

$$HSP = HSP_{20} + HSP_{40} + HSP_{60} + HSP_{70} + HSP_{90} + HSP_{100} \quad (1)$$

where $+$ represents the union or combination of six HSP families.

The HSP_{40} dataset, formulated in equation 2, contains J-protein sequences originally sourced from HSPIR [12]. The original dataset was preprocessed by removing sequences that (1) have non-standard amino acids and (2) that have $\geq 40\%$ pairwise sequence similarity. The final dataset contains 1,199 sequences, of which 63 belong to Type-1 (T_1) J-protein, 55 belong to Type-2 (T_2), 1061 to Type-3 (T_3), and 20 to Type-4 (T_4). Table 2 provides the statistics for the two datasets. A small Python script was used to count the total amino acids (AA), minimum length (MinLen), maximum length (MaxLen) and average length (AvgLen) of protein sequences in each class.

$$HSP_{40} = T_1 + T_2 + T_3 + T_4 \quad (2)$$

Table 2: Summary of the two datasets: HSP and HSP_{40}

HSP						
Class	**Description**	**Samples**	**Total AAs**	**MinLen**	**MaxLen**	**AvgLen**
HSP_{20}	sHSP/Small HSP	354	75,114	83	1,214	212.1
HSP_{40}	DnaJ-proteins	1,257	582,725	62	3,321	463.5
HSP_{60}	GroEL/ES or chaperonins	159	132,302	80	2,740	832
HSP_{70}	DnaK/chaperones	278	178,970	130	2,725	643.7
HSP_{90}	Chaperonines	52	32,833	112	978	631.4
HSP_{100}	High Molecular Weight HSP	81	60,548	156	1,412	747.5
NonHSP		9,965	4,592,664	51	2,974	473.5
HSP_{40}						
T_1	$Type_1$	63	30,384	342	1,552	482.2
T_2	$Type_2$	55	20,836	205	950	378.8
T_3	$Type_3$	1,061	505,403	99	4,524	476.3
T_4	$Type_4$	20	5,529	66	1,049	276.4
Total		13,345	6,229,345	126	2,130.8	510.6

The two datasets are imbalanced. In HSP, the number of AA sequences belonging to non-HSP significantly exceeds the number of AA sequences found in the six HSP families. Similarly, in HSP_{40}, T_3 contains many sequences (1,061) while other

types contain very few sequences. Meher et al. [22] handled the class imbalance by creating a balanced dataset that contains an equal number of randomly selected sequences (2,180) for HSPs and non-HSPs from the total HSP and non-HSP sequences. Similarly, Akbar et al. [15] created a balanced dataset of HSP containing the same number of HSP and non-HSP sequences. Additionally, [15, 21] used oversampling (with the SMOTE algorithm [34]) to overcome the class imbalance in HSP_{40} and in HSP respectively.

3.2 FSP4HSP

The proposed FSP4HSP approach, to classify/identify protein sequences into non-HSP, HSP families and sub-types as well as multiple classes is illustrated in Figure 1. It consists of four parts: (1) encoding protein sequences, (2) feature extraction using pattern mining, (3) using the extracted features in classification by training various classification models, and (4) benchmark evaluation.

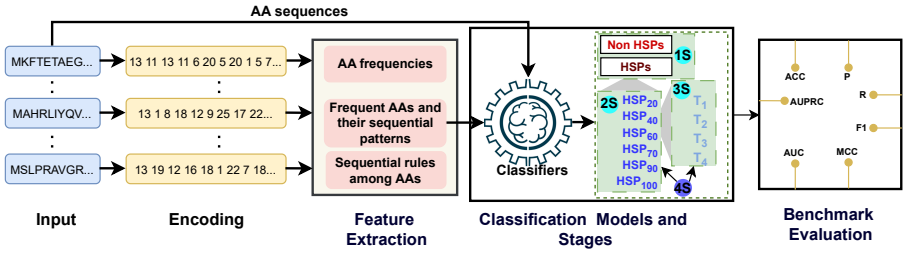

Fig. 1: Flow chart of FSP4HSP. AA stands for amino acid. The three stages (1S, 2S and 3S) are for binary classification and the fourth stage is for multi-class classification in HSP and HSP_{40} datasets.

3.2.1 Encoding

In this first part, protein sequences are converted into sequences of items (symbols) of discrete type. The "*AAs to integers*" abstraction is utilized for transformation. The main aim of that transformation, changing every distinct AA into a unique positive integer, is to represent the AA information in a protein sequence using integer codes. This transformation is lossless, reversible, and is required for applying the pattern mining algorithms discussed in this paper, which are generic tools for analyzing discrete sequences of symbols.

The following is a detailed explanation of the transformation process. Let $AA = \{$Ala (A), Cys (C), Asp (D), Glu (E), Phe (F), Gly (G), His (H), Ile (I), Lys (K), Leu (L), Met (M), Asn (N), Pro (P), Gln (Q),Arg (R), Ser (S), Thr (T), Val (V), Trp (W), Tyr (Y)$\}$ represents a set of 20 distinct amino acids. A protein sequence is a list of amino acids, written as $PS = \langle AA_1, AA_2, ..., AA_n \rangle$, such that $AA_i \in$

AA ($1 \leq i \leq n$). Let $PSC = \langle PS_1, PS_2, ..., PS_p \rangle$ denote a corpus of protein sequences. The process of transformation involves converting every protein sequence in PSC. A $PS = \langle AA_1, AA_2, ..., AA_n \rangle$ is transformed as $PS' = \langle f(AA_1), -1, f(AA_2), -1, ..., f(AA_n), -1, -2 \rangle$ by a transformation function f (f:$AA \rightarrow \mathbb{N}$)) that maps each amino acid to a distinct positive integer. Take the sample PSC of four protein sequences of AAs from Table 3(a) as an example. It is converted into positive integers, and the result is shown in Table 3(b). Take the protein sequence with ID 1 as an example. The amino acids N, Y, E, L, G, V are encoded as 14, 25, 5, 12, 7, and 22, respectively. In this example, -1 is used as a separator between amino acids, and the code -2 denotes the end of a protein sequence. The two datasets of HSP and HSP_{40} and their transformation is available at: `github.com/saqibdola/FSP4HSP`.

Table 3: The transformation of (a) a *PSC* corpus into (b) protein sequences where AAs are encoded as integers.

(a)

ID	Protein sequences
1	⟨{...NYYEELGV...}⟩
2	⟨{...RDYYEILN...}⟩
3	⟨{...DLYSVLGV...}⟩
4	⟨{...LGVYRFRE...}⟩

(b)

ID	Protein sequences
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 -1 25 -1 18 -1 6 -1 18 -1 5 -1 -2

3.2.2 Feature Extraction

In the second part, FSP4HSP extracts features from PSs that will be later used for building the classification models. Two different approaches are used.

1) Using Entire AA sequences. The first approach, is the most straightforward one for analysis and classification. It is to use the entire AA sequences (EAASs) of HSP and non-HSPs as features. This is formulated by the following equation for a sequence PS:

$$PS = AA_1AA_2AA_3AA_4.....AA_n$$

where AA_1 represents the first AA of PS, AA_2 represents the second AA, and so on.

2) Extracting frequent patterns as features. The second approach is more sophisticated. It consists of extracting frequent patterns from the HSP and non-HSP sequences from a protein sequence corpus PSC, to capture important sequential relationships in these sequences. Pattern mining, particularly SPM, is used to find sequential patterns (subsequences of AAs) that occur frequently in protein sequences. Formally, let there be two protein sequences $PS_x = \langle x_1, x_2, ..., x_n \rangle$ and $PS_y = \langle y_1, y_2, ..., y_m \rangle$. The protein sequence ($PS_x$) is contained (or present) in another protein sequence (PS_y) (denoted as $PS_x \sqsubseteq PS_y$), if and only if some integer numbers can be found $1 \leq i_1 < i_2 < ... < i_n \leq m$, such that $x_1 = y_{i1}, x_2 = y_{i2}, ..., x_n = y_{im}$. In other words, PS_x is a *subsequence* of PS_y if PS_y contains PS_x. For a protein sequence PS_x, its *support* in PSC, denoted as $sup(PS_x)$, is the total number of subsequences in PSC containing PS_x: $sup(PS_x) = |\{PS | PS_x \sqsubseteq PS \wedge PS \in PSC\}|$.

Finding all the frequent sub-sequences of AA for a PSC with a minimum support value ($minsup > 0$) is the aim of frequent SPM. A sub-sequence PS_x is frequent if its support is greater than or equal to $minsup$ ($sup(PS_x) \geq minsup$). For example, the sub-sequence $\langle LGV \rangle$ is found in three different PSs of the PSC of Table 3 (a). Thus, its support is 3, meaning that it occurs in 75% of the sequences. Note that for protein sequences, the support measure is relevant as it allows identifying frequent subsequences of AAs and thus uncovers their resemblances.

CM-SPAM [35], an improved version of the SPAM algorithm, is used to find FSPAAs. An efficient data structure is used by CM-SPAM to reduce the overall search space of patterns to find the frequent patterns. CM-SPAM is utilized because it is efficient and it offers multiple input parameters that allows users to obtain more tailored results such as by specifying the desired number of frequent sequential patterns of AAs to be extracted from protein sequences.

Besides sequential patterns, some SPM algorithms can be used to find sequential rules [36]. For protein sequences, a sequential rule (SR) shows a relationship among two sets of AAs by considering not only the support but also the confidence of the relationship between AAs represented by the rule. Between two sets of AAs P and Q, where $P, Q \subseteq AA$, s.t. $P \cap Q = \emptyset$ and $P, Q \neq \emptyset$, a SR is written as $P \rightarrow Q$. This rule means that if AAs of P are present in a sequence, then in the same sequence, AAs of Q will appear after. Formally, a protein sequence $PS = \langle AA_1, AA_2, ..., AA_n \rangle$ contains P iff $P \subseteq \bigcup_{x=1}^{n} \{AA_x\}$. Moreover, PS contains the rule r (denoted as $r \sqsubseteq PS$) in the case where an integer d exists such that $1 \leq d < n$, $P \subseteq \bigcup_{x=1}^{d} \{AA_x\}$ and $Q \subseteq \cup_{x=d+1}^{n} \{AA_x\}$. The support and confidence of a rule r in a protein sequence corpus PSC are defined as:

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

$$conf_{PSC}(r) = \frac{|\{PS | r \sqsubseteq PS \wedge PS \in PSC\}|}{|\{PS | P \sqsubseteq PS \wedge PS \in PSC\}|} \tag{4}$$

In this work, the ERMiner [36] algorithm is used to discover frequent sequential rules among AAs in protein sequences. ERMiner is utilized because it is one of the fastest algorithms for identifying sequential rules, as it uses various data structures and merging operations for search space reduction and to perform fast calculations.

3.2.3 Classification

In the third part of the proposed FSP4HSP approach, the EAASs and FSPAAs are used for the classification of HSPs. HSP sequences are generally long. In Table 2, the HSP and HSP_{40} datasets on average contain 473 and 510 AAs respectively. The majority of the PSs in both datasets consist of repetitive occurrences of the same AAs, ranging from few to tens of repetitions. This AA repetition can be replaced with FSPAAs for improved classification. More precisely, FSP4SPM makes use of FSPAAs to classify HSPs vs non-HSPs, HSP families, and sub-types.

Some previous models such as HSP-PseRAAAC [17], JPRED [18], the model of Ahmad et al. [19], JPPRED [21], DeepHSP and DeeperHP [23] take the classification

task as binary classification, where the identification/classification of HSP/HSP families HSP_{40} types was performed simultaneously. Pred-HSP [20] employed a two-stage approach where SVM was used for binary classification of HSPs vs non-HSPs in the first-stage. In the second stage, SVM was used for HSP families classification. ir-HSP proposed a three-stage approach. The first stage used binary-class SVM to classify HSPs vs non-HSPs, the second and third stages used SVM for the classification of HSP families and HSP_{40} types respectively. MulCNN-HSP [16] also used a two-stage approach where a CNN was first used for HSP vs NonHSP classification and then for HSP families classification. HSP families classification is a multi-class problem, but in Pred-HSP, ir-HSP and MulCNN-HSP, a series of classifiers were developed.

In this work, both binary and multi-class (MC) classification is performed. A four-stage classification approach is employed: The first, second, and third stages (1S, 2S, 3S) deal with the classification of HSPs vs non-HSPs, HSP families and HSP_{40} types. The fourth stage (4S) is for the MC classification: HSP_{20} vs HSP_{40} vs HSP_{60} vs HSP_{70} vs HSP_{90} vs HSP_{100} in the first dataset and T_1 vs T_2 vs T_3 vs T_4 in the second dataset. The first classification method (binary classification) is performed to train a classifier for the separate classification of each HSP or non-HSP, HSP families, or subtypes. In the MC classification method, each protein sequence is assigned a label corresponding to its respective type name. There is one HSP and non-HSP type, 6 HSP families, and 4 sub-types of HSP_{40}. Thus there are a total of 12 classes. In the two datasets, MC classification deals with the classification of HSP families and HSP_{40} types.

Most of the previous studies used one classifier, such as SVM [17, 18, 20, 22], CNN [16, 23] and EL [21]. Ahmad et al. [19] and Akbar et al. [15] used four and six classifiers respectively. In this work, various string-based and integer-based classifiers are used. which are: (1) Naive Bayes Multinomial Text (NBMT), (2) Recurrent Neural Network Sequence Classifier (RNNSC), (3) Stochastic Gradient Descent Text (SGDT), (4) ZeroR, (5) Naive Bayes (NB), (6) Logistic Regression (LR), (7) Support Vector Machine (SVM), (8) k-Nearest Neighbors (kNN), (9) K Star (K^*), (10) Decision Tree (J48), (11) Random Forest (RF), and (12) Multi-Layer Perceptron (MLP). The first four classifers (NBMT, RNNSC, ZeroR, and SGDT) are string-based classifiers and the next eighth (NB, LR, J48, kNN, MLP, SVM, RF, and K^*) are integer-based classifiers.

The performance of classifiers is assessed using seven metrics, which are. (1) Accuracy (ACC), (2) Recall (R), (3) Precision (P), (4) F1 score (F1), (5) MCC (Matthews correlation coefficient), (6) Area under the curve (AUC) and (7) Area Under the Precision-Recall Curve (AUPRC). The seven metrics are defined as:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Recall(R) = \frac{TP}{TP + FN} \tag{6}$$

$$Precision(P) = \frac{TP}{TP + FP} \tag{7}$$

$$F - measure = 2 \times \frac{P \times R}{P + R} \tag{8}$$

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{9}$$

$$\text{AUC} = \int_0^1 \text{R}(d\text{FPR}) \tag{10}$$

$$AUPRC = \sum_{i=1}^{n} \frac{(R_i - R_{i-1}) \times (P_i + P_{i-1})}{2} \tag{11}$$

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 EAASs or FSPAAs to a specific HSP type. TN represents the correctly classified EAASs or FSPAAs as not part of a specific HSP type. FP represents the incorrectly identified EAASs or FSPAAs to a specific HSP type. FN represents the incorrectly identified EAASs or FSPAAs as not part of a specific HSP type. In equation 10, *dFPR* is for the derivative of *FPR* = $\frac{FP}{FP+TN}$. P_i and R_i in equation 11 represent the values for precision and recall, respectively, at the i-th decision threshold. Each classification model performance is assessed through five-fold cross-validation in this study. The main reason to use five-fold is that it is used in majority of the previous studies [16, 20, 22, 23] for HSP identification.

4 Results

The experiments performed in this section utilized a computing machine running Windows 11 and equipped with a RAM of 8 GB and a Core i5-11530H processor. The open-source library SPMF [37], developed in JAVA, was used to analyze and discover FSPs and sequential rules in the two datasets (HSP and HSP_{40}). Moreover, the open-source WEKA [38], a JAVA-built software, was utilised for testing and training of classification models. In all four classification stages, the default hyperparameters of the models available in WEKA were employed.

4.1 Frequent AAs, FSPAAs and Sequential Rules

In a preliminary experiment, the distribution of AAs in the two datasets was first analyzed to compare their occurrence or frequency in different HSP families and sub-types. For this purpose, the Apriori algorithm [39] was used. It can be used to effectively count the instances of specific values or groups of values in data. Table 4 provides the details for the occurrence frequencies of AAs in two datasets.

The AA Glutamic acid (E) occurred as the top five most frequent AAs in HSP families and sub-types, as well as in NonHSP protein sequences. The two AAs Alanine (A) and Leucine (L) occurred as the top five most frequent AAs in all, except HSP_{90} and T_1, respectively. The top five AAs, based on occurrence frequency, are L, A, Serine (S), E, and Lysine (K) (Figure 2). Overall, L occurred more with 8.64%, followed by A (7.76%), S (7.75%), E (7.52%) and K (7.27%), respectively. We were able to examine the frequencies of AAs in two datasets through the Apriori analysis and to find the most frequent AAs on the basis of occurrence frequency. However,

Table 4: Total occurrence counts of AAs in two datasets. The values outside brackets are the total occurrence counts of AAs and the values inside brackets are the % of AAs. For each type, $*$ is used to denote the most frequent AA. Similarly $\star$, $\circ$, $\bullet$ and $\diamond$ are used to indicate the second third, fourth and fifth most frequent AA, respectively.

	HSP						
AA	**HSP20**	**HSP40**	**HSP60**	**HSP70**	**HSP90**	**HSP1000**	**NonHSP**
A	$\bullet$5, 008(6.66)	$*$51, 396(8.81)	$\diamond$8, 630(6.52)	$\star$14, 478(8.08)	2,179 (6.63)	$\star$5, 362(8.85)	$\circ$323, 358(7.04)
C	643 (0.85)	8,038 (1.37)	2,384 (1.80)	2,028 (1.13)	294 (0.89)	505 (0.83)	73,084 (1.59)
D	4,739 (6.3)	34,737 (5.96)	7,844 (5.92)	10,857 (6.06)	$\diamond$2, 252(6.85)	3,540 (5.84)	243,939 (5.31)
E	$*$6, 023(8.01)	$\bullet$44, 357(7.61)	$\circ$8, 821(6.66)	$\circ$13, 742(7.67)	$\star$3, 000(9.13)	$\circ$4, 747(7.84)	$\bullet$307, 686(6.69)
F	2,868 (3.81)	22,236 (3.81)	4,751 (3.59)	6,828 (3.81)	1,388 (4.22)	2,012 (3.32)	178,033 (3.87)
G	4,829 (6.42)	37,166 (6.37)	7,660 (5.78)	11,295 (6.31)	1,660 (5.05)	$\bullet$4, 164(6.87)	$\diamond$281, 850(6.13)
H	1,867 (2.48)	13,521 (2.32)	3,039 (2.29)	3,102 (1.73)	621 (1.89)	1,178 (1.94)	107,365 (2.33)
I	3,742 (4.98)	25,049 (4.29)	7,938 (5.99)	11,504 (6.42)	1,921 (5.85)	3,727 (6.15)	246,947 (5.37)
K	4,735 (6.30)	$\diamond$41, 823(7.17)	$\bullet$8, 813(6.66)	12,177 (6.80)	$\circ$2, 661(8.10)	3,758 (6.20)	280,050 (6.09)
L	$\circ$5, 561(7.40)	$\star$45, 965(7.88)	$\star$12, 067(9.12)	$*$15, 108(8.44)	$*$3, 105(9.45)	$*$6, 181(10.20)	$*$435, 401(9.48)
M	1,730 (2.30)	11,705 (2)	2,936 (2.21)	3,433 (1.91)	772 (2.35)	1,162 (1.91)	100,201 (2.18)
N	3,314 (4.41)	25,858 (4.43)	6,850 (5.17)	8,048 (4.49)	1,536 (4.67)	2,554 (4.21)	224,744 (4.89)
P	4,875 (6.49)	28,045 (4.81)	5,813 (4.39)	8,102 (4.52)	1,214 (3.69)	2,499 (4.12)	243,416 (5.30)
Q	3,111 (4.14)	25,155 (4.31)	5,352 (4.04)	6,979 (3.89)	1,148 (3.49)	2,248 (3.71)	201,561 (4.38)
R	$\diamond$4, 930(6.56)	38,245 (6.56)	6,823 (5.15)	9,373 (5.23)	1,548 (4.71)	4,083 (6.74)	244,140 (5.31)
S	$\star$5, 740(7.64)	$\circ$45, 485(7.80)	$*$12, 485(9.43)	$\diamond$12, 528(7)	$\bullet$2, 492(7.58)	4,007 (6.61)	$\star$385, 695(8.39)
T	3,871 (5.15)	28,186 (4.83)	7,051 (5.34)	10,477 (5.85)	1,698 (5.17)	3,065 (5.06)	251,868 (5.48)
V	5,200 (6.92)	30,987 (5.31)	8,591 (6.49)	$\bullet$12, 681(7.08)	1,989 (6.05)	$\diamond$4, 160(6.87)	276,273 (6.01)
W	694 (0.92)	6,117 (1.04)	968 (0.73)	1,344 (0.75)	320 (0.97)	262 (0.43)	52,202 (1.13)
Y	1,634 (2.17)	18,654 (3.2)	3,486 (2.63)	4,886 (2.72)	1,035 (3.15)	1,334 (2.20)	134,851 (2.93)

	HSP_{40}			
AA	**T-I**	**T-II**	**T-III**	**T-IV**
A	$\diamond$2, 018(6.64)	$\star$1, 656(7.94)	$*$45, 326(8.96)	$\circ$495(8.12)
C	707 (2.32)	209 (1)	6,841 (1.35)	68 (1.22)
D	1,880 (6.18)	1,291 (6.19)	30,241 (5.98)	263 (4.75)
E	$\circ$2, 084(6.85)	$\diamond$1, 362(6.53)	$\bullet$38, 815(7.68)	$\star$450(8.13)
F	1,258 (4.14)	973 (4.66)	19,020 (3.76)	214 (3.87)
G	$*$3, 107(10.22)	$*$2, 289(10.98)	30,450 (6.02)	360 (6.51)
H	812 (2.67)	416 (1.99)	11,686 (2.31)	110 (1.98)
I	1,427 (4.69)	971 (4.66)	21,574 (4.26)	273 (4.93)
K	$\star$2, 314(7.61)	$\bullet$1, 401(6.72)	$\diamond$36, 109(7.14)	$\bullet$416(7.52)
L	1,938 (6.37)	$\circ$1, 467(7.04)	$\star$40, 289(7.97)	$*$525(9.49)
M	589 (1.93)	362 (1.73)	10,118 (2)	141 (2.55)
N	1,257 (4.13)	972 (4.66)	22,391 (4.43)	231 (4.17)
P	1,553 (5.11)	1,185 (5.68)	24,159 (4.78)	225 (4.06)
Q	1,365 (4.49)	781 (3.74)	21,704 (4.29)	285 (5.15)
R	1,754 (5.77)	1,286 (6.17)	33,562 (6.64)	294 (5.31)
S	$\bullet$2, 047(6.73)	1,280 (6.14)	$\circ$40, 082(7.93)	$\diamond$401(7.25)
T	1,612 (5.30)	1,055 (5.06)	24,526 (4.85)	237 (4.28)
V	1,727 (5.68)	1,158 (5.55)	26,714 (5.28)	346 (6.25)
W	139 (0.45)	104 (0.49)	5,544 (1.09)	49 (0.88)
Y	876 (2.88)	618 (2.96)	16,252 (3.21)	146 (2.64)

this initial analysis is restricted because the Apriori algorithm does not take into account the consecutive arrangement of AAs and does not guarantee that AAs occur consecutively in a PS. Among the top 5 most frequent AAs, it was observed that L, A and K, E have the highest occurrence percentages in HSP_{100} and HSP_{90}, respectively. S has the highest occurrence frequency in HSP_{70}. In HSP_{40}, L, K, E, A, and T have the highest occurrence frequency in T_1, T_4 and T_1, T_3, respectively. The occurrence frequency of L increases gradually from T_1 to T_4. The same behavior is also observed for L, with little difference, in HSP families.

Efficient SPM algorithms were developed in last two decades that overcomes the constraints of Apriori. These algorithms can be used to find and discover more interesting and important sequential patterns, rules and information in the data. One

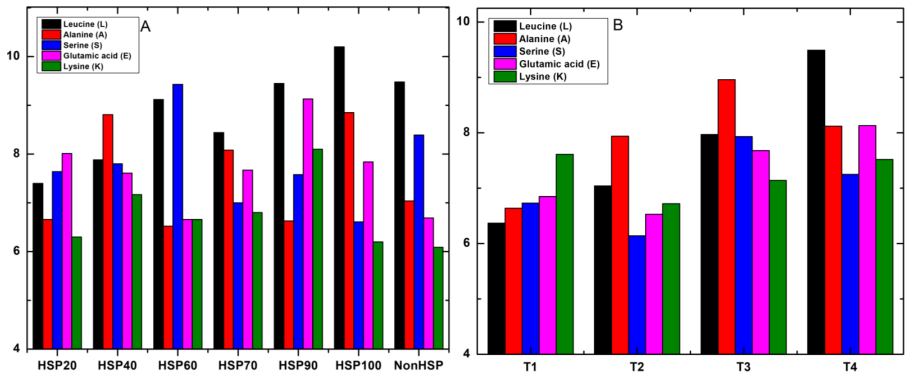

Fig. 2: Percentages of the top five AAs in the (a) HSP dataset, and (b) HSP_{40} dataset

simple example (Figure 3) is provided where the obtained results with the CM-SPAM and ERMiner are explained. Within the figure's top box, five raw PSs, containing 30 AAs, are shown. In the figure's bottom left corner, the FSPAAs discovered in raw PSs are listed. In the figure's bottom right corner, the sequential rules of AAs discovered in raw PSs are listed with varying numbers of antecedents and consequent. For a sequential rule, the values before and after the arrow are for the support and confidence respectively. The collection of AA patterns identified within a PS can be interpreted as a description or characterization of that sequence. These FSPAAs serve as features in the classification procedure of models. Overall, using the SPM algorithms on two datasets proved to be relatively efficient and quick. Note that SPM algorithm parameters are adjusted and fine-tuned to obtain the desired patterns and rules.

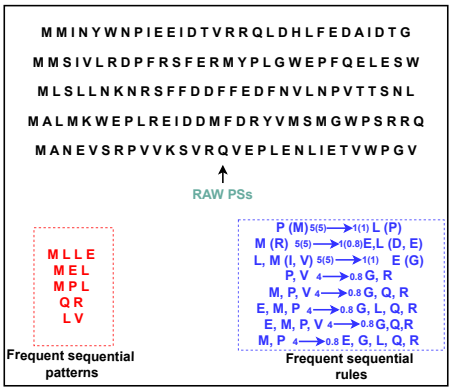

Fig. 3: AA frequent sequential patterns and rules discovered in raw PSs. FSPAAs discovered in PSs of HSPs can be interpreted as their description or characterization, that are then used in the classification process.

4.2 Classification Results

The results of twelve classification models on two protein sequence representation methods (EAASs and FSPAAs) for two datasets i.e. HSP and HSP_{40} is discussed in this section.

Before classification, discovered FSPAAs are preprocessed to ensure that each pattern length is at least 9. The binary and MC classification results for models when trained and tested on EAASs are listed in Table 5. We find that NBMT, ZeroR, and SGDT provided the same results for all classes. WordTokenizer is used in both NBMT and SGDT. SGDT was slow compared to the other three string-based classifiers. In the HSP and HSP_{40} datasets, the highest accuracy was achieved with NBMT, ZeroR and SGDT on binary classification of HSP_{90} (97.6%) and T_4 (98.3%) respectively. RNNSC performed worst on both datasets. However, its performance was better for MC classification in the HSP_{40} and its type T_3. Interestingly, NBMT, ZeroR and SGDT overall results for binary classification are better than the JPPRED model [21] that combined three feature extraction methods (SAAC, PAAC and PSSM) and used ensemble classifiers. From the two datasets, we removed the most frequently occurring AA, that is E, to check whether it has any effect on the overall accuracy of classifiers. Interestingly, the four classifiers generated the same accuracy when run on PSs of HSPs without the AA E. The same behavior of classifiers were found when the second most frequent AA, that is A, is removed from the sequences.

Table 5: Classifiers accuracy on EAASs

Dataset	**Stage**	**NBMT**	**RNNSC**	**ZeroR**	**SGDT**
HSP	1S NonHSP	0.82	0.179	0.82	0.82
	2S HSP_{20}	0.837	0.162	0.837	0.837
	2S HSP_{40}	0.576	0.576	0.576	0.576
	2S HSP_{60}	0.927	0.072	0.927	0.927
	2S HSP_{70}	0.872	0.127	0.872	0.872
	2S HSP_{90}	**0.976**	0.023	**0.976**	**0.976**
	2S HSP_{100}	0.962	0.037	0.962	0.962
	Average	0.852	0.168	0.852	0.852
	4S MC	**0.82**	0.013	**0.82**	–
Dataset	**Stage**	**NBMT**	**RNNSC**	**ZeroR**	**SGDT**
HSP40	3S T_1	0.947	0.052	0.947	0.947
	3S T_2	0.954	0.045	0.954	0.954
	3S T_3	0.884	0.884	0.884	0.884
	3S T_4	**0.983**	0.016	**0.983**	**0.983**
	Average	0.931	0.249	0.931	0.931
	4S MC	0.884	**0.885**	0.884	–

The binary and MC classification results for models when trained and tested on FSPAAs are listed in Table 6. Eight integer-based classifiers performance on FSPAA was better than the four string-based classifiers performance on EAAS. It was observed that K^*, RF and DT performance was better than others for binary classification in both datasets. K^* also performed better on MC classification in the HSP dataset, whereas DT performed slightly better than K^* and RF on MC classification

Table 6: Classifiers accuracy on FSPAAs

Dataset	Stage	NB	LR	SVM	kNN	K*	DT	RF	MLP
HSP	1S NonHSP	0.726	0.987	0.98	0.981	**0.99**	0.984	0.985	0.978
	2S HSP_{20}	0.981	1	1	1	**1**	1	1	1
	2S HSP_{40}	0.916	0.995	0.931	0.985	**0.998**	0.991	0.996	0.996
	2S HSP_{60}	0.835	0.996	0.983	0.991	**1**	0.996	0.996	0.996
	2S HSP_{70}	0.995	0.995	0.985	0.988	0.995	0.993	**0.996**	0.996
	2S HSP_{90}	0.991	0.898	0.906	0.995	**1**	1	1	0.993
	2S HSP_{100}	1	0.998	1	1	**1**	0.998	1	1
	Avg.	0.92	0.981	0.969	0.991	**0.997**	0.994	0.996	0.993
	4S MC	0.965	0.977	0.961	0.968	**0.988**	0.982	0.984	0.981

Dataset	Stage	NB	LR	SVM	kNN	K*	DT	RF	MLP
HSP	3S T_1	0.735	0.872	0.845	0.905	**0.952**	0.925	0.952	0.917
	3S T_2	0.867	0.91	0.902	0.895	**0.95**	0.937	0.937	0.932
	3S T_3	0.912	0.907	0.892	0.95	0.962	**0.97**	0.967	0.97
	3S T_4	0.667	0.975	0.965	0.955	0.965	**0.985**	0.977	0.985
	Avg.	0.795	0.916	0.901	0.926	0.957	0.945	**0.958**	0.951
	4S MC	0.837	0.867	**0.961**	0.86	0.895	0.90	0.892	0.862

in the HSP_{40} dataset. All the classifiers were fast and terminated within seconds. For the HSP dataset, based on average accuracy in binary classification, the models are ranked as follows: (1) K^* (99.7%), (2) RF (99.6%), (3) DT (99.4%), (4) MLP (99.3%), (5) kNN (99.1%), (6) LR (98.1%), (7) SVM (96.9%) and (8) NB (92%). Models are ranked as follows based on MC classification is in the order: (1) K^* (98.8%), (2) RF (98.4%), (3) DT (98.2%), (4) MLP (98.1%), (5) LR (97.7%), (6) kNN (96.8%), (7) NB (96.5%) and (8) SVM (96.1%). For the HSP_{40} dataset, based on average accuracy in binary classification, the models are ranked as follows: (1) RF (95.8%), (2) K^* (95.7%), (3) MLP (95.1%), (4) DT (94.5%), (5) kNN (92.6%), (6) LR (91.6%), (7) SVM (90.1%) and (8) NB (79.5%). Classifiers are ranked as follows based on MC classification: (1) SVM (96.1%), (2) DT (90%), (3) K^* (89.5%), (4) RF (89.2%), (5) LR (86.7%), (6) MLP (86.2%), (7) kNN (86%) and (8) NB (83.7%). The complete results of K^* on both datasets is provided in Table 7. One of the reasons for the relatively better performance of K^*, RF and DT is that that they are non-parametric models and they appear to have good ability at properly handling various patterns as features for classification.

HSP Dataset	HSP_{40} Dataset
HSP_{20} NB	T_1 MLP, kNN, LR, SVM, NB
HSP_{40} kNN, SVM, NB	T_2 kNN, LR, SVM, NB
HSP_{60} SVM, NB	T_3 LR, SVM, NB
HSP_{70} kNN, SVM	T_4 NB, DT, MLP
HSP_{90} LR, SVM	MC kNN, SVM, NB
HSP_{100}	
MC kNN, SVM, NB	

Fig. 4: t-test ACC results for seven classifiers compared to K*. Red-colored classifiers performed worst and blue-colored classifiers performed significantly better than K*

Table 7: K* results on FSPAAs

Dataset	Stage	ACC	Stage	R	F1	MCC	AUC	AUPRC
HSP	1S 1S NonHSP	0.99	0.99	0.99	0.99	0.959	0.999	0.999
	2S HSP_{20}	1	1	1	1	1	1	1
	2S HSP_{40}	0.998	0.998	0.998	0.998	0.994	0.999	0.999
	2S HSP_{60}	1	1	1	1	1	1	1
	2S HSP_{70}	0.995	0.995	0.995	0.995	0.982	1	1
	2S HSP_{90}	1	1	1	1	1	1	1
	2S HSP_{100}	1	1	1	1	1	1	1
	MC	0.988	0.989	0.989	0.989	0.987	0.999	0.997
Dataset	**Stage**	**ACC**	**P**	**R**	**F1**	**MCC**	**AUC**	**AUPRC**
HSP40	3S T_1	0.952	0.952	0.953	0.952	0.873	0.988	0.990
	3S T_2	0.95	0.95	0.95	0.949	0.864	0.981	0.983
	3S T_3	0.962	0.963	0.963	0.963	0.902	0.984	0.978
	3S T_4	0.965	0.966	0.965	0.964	0.906	0.987	0.990
	MC	0.895	0.899	0.895	0.895	0.862	0.984	0.957

In WEKA, the paired t-test was run to determine if K* significantly outperformed the others on FSPAAs. The comparative results based on the accuracy of models are shown in Figure 4. Classifiers in red indicate those that considerably underperformed K*. On the other hand, classifiers in blue are those that performed significantly better than K^*. NB did not perform well compared to K^* in almost all cases, except for the HSP_{100}. DT and RF did not appear as red in any case, that shows that both of them performed similarly to K^*. MLP almost performed well with only one appearance as red in the T_1 of HSP_{40}. For binary classification of HSP_{40} T_4, DT and MLP performed significantly better than K*.

t-SNE (t-distributed Stochastic Neighbor Embedding) [40] was used for visualization of EAASs and FSPAAs. This analysis provides important insights into the structure and relationships of AAs within EAASs and FSPAAs. Moreover, the visualization offers a more detailed explanation of classifier learning process and also investigates the ability of FSP4HSP to distinguish AA sequences of HSPs. Figure 5 provides the 2D visualization of EAASs and FSPAAs in HSP and HSP_{40}. In both datasets, EAASs of HSPs are scattered (Figure 5A, Figure 5C) and it is hard to distinguish between them. There is no clear separation among HSP families, sub-types or Non-HSPs and they show the same degrees of aggregation or clustering with a high level of scattering. On the other hand, FSPAAs of HSPs (Figure 5B and Figure 5D) in both datasets are more separable, but their distinction was not distinct or prominent. The six HSP families showed different levels of clustering, but there is still some variation among them. Note that HSP_{70} and HSP_{90} form two distinct clusters. Similarly, all types of HSP_{40} form multiple distinct clusters. That is the reason that integer-based classifiers performed less well on them as compared to FSPAAs of HSP families.

In summary, the two main findings of this work are:

1. Classifiers achieved better performance, both for binary and MC classification, on FSPAAs as compared to EAASs. For binary classification, K* on FSPAAs achieved an improvement of approximately 17.01% and 2.79% in accuracy compared to the NBMT, ZeroR and SGDT on EAASs in the HSP and HSP_{40}

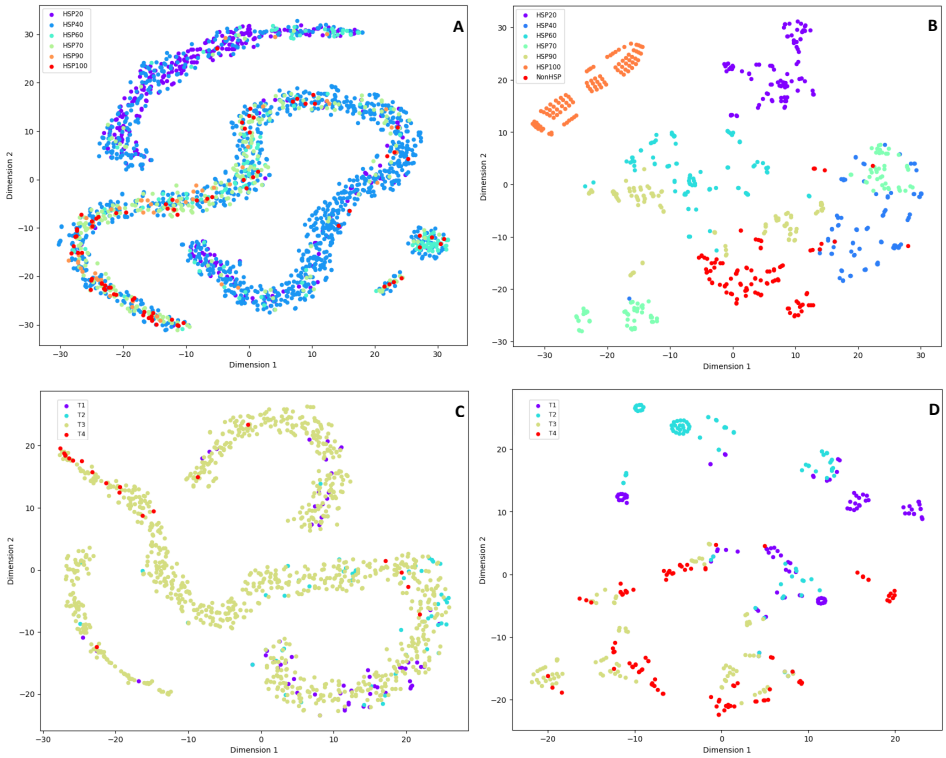

Fig. 5: Feature space distribution visualization in differentiating HSPs, their families and sub-types: (A) EAASs in HSP dataset, (B) FSPAAs in the HSP dataset, (C) EAASs in the HSP_{40} dataset, (B) FSPAAs in the HSP_{40} dataset

datasets, respectively. For MC classification, K* on FSPAAs achieved an improvement of approximately 20.48% and 8.13% in accuracy compared to NBMT, Zero, SGDT and RNNSC on EAASs in the two datasets, respectively.

2. Instead of using the whole PS or amino acid composition features, FSPAAs can be used for improved and interpretable identification of HSPs.

4.3 Comparison

FSP4HSP is compared in this section with previous methods for HSPs classification/prediction, that include MulCNN-HSP [16], DeepHSP and DeeperHSP [23], PredHSP [20], ir-HSP [22], iHSP-PseRAAAC [17], JPRED [18], JPPRED [21], ES-PredHSP [15] and the prediction method (called PM here) of [19].

MulCNN-HSP [16] used one-hot encoding and multi-scale convolutional neural networks for latent features. PM [19] used three different feature extraction methods (SAAC, PAAC and DPC), in which the best classifier SVM performed better on features extracted with DPC. JPPRED [21] used three feature extraction methods

Table 8: Comparison of FSP4HSP with previous classification/detection approaches for HSPs

Model	Stage	ACC	P	R	F1	MCC	AUC	AUPRC
MulCNN-HSP [16]	$\overline{1S+2S}$	0.968	0.894	0.846	0.85	0.84.4	0.984	0.936
DeepHSP [23]	2S	0.968	0.961	0.798	0.861	0.855	0.977	0.893
DeeperHSP [23]	2S	0.992	0.974	0.966	0.969	0.966	0.994	0.966
PredHSP [20]	$\overline{1S+2S}$	0.864	–	0.8	–	0.593	0.919	–
ir-HSP [22]	$\overline{1S+2S+3S}$	0.921	0.87	0.801	–	0.683	0.889	0.567
iHSP-PseRAAAC [17]	2S	0.878	–	0.750	–	0.695	–	–
JPRED [18]	3S	0.94	–	0.635	0.642	–	–	–
JPPRED [21]	3S	0.852	–	0.875	–	–	–	–
PM [19]	$\overline{2S+3S}$	0.938	–	0.89	0.765	0.775	–	–
ES-PredHSP [15]	$\overline{1S+2S}$	–	0.865	0.861	–	–	–	–
FSP4HSP(K^*)	$\overline{1S+2S+3S}$	0.982	0.983	0.983	0.974	0.952	0.994	0.994
FSP4HSP(K^*)	$\overline{4S_{HSP}+4S_{HSP40}}$	0.941	0.944	0.942	0.942	0.924	0.991	0.9977

1S First Stage; 2S: Second Stage; 3S: Third Stage, 4S: Fourth Stage. the bar over stages represents the average. Binary classification in 1S, 2S and 3S, MC classification in 4S.

(SAAC, PAAC, and PSSM). SU-IFS is then employed to find the optimal feature set. ES-PredHSP [15] used the feature extraction method of CKSAAP and used six classifiers, in which SVM performed better. Etchebest et al. [41] determined a new RAAA type on the basis of Protein Blocks [42], where the 20 AAs can be grouped into five different cluster profiles (CP): CP(13), CP(11), CP(9), CP(8), and CP(5). SVM in iHSP-PseRAAAC and JPRED performed best on CP11 and CP8, respectively.

Table 8 compares the performance of FSA4HSP with previous approaches. One previous method, ir-HSP [22], performed three stages of classification (1S, 2S and 3S). For the two stages (1S, 2S), K^* in the proposed method achieved 99.7% accuracy, on average, and achieved an improvement of approximately 3% compared to the most recent model MulCNN-HSP (average accuracy of 96.8% for 1S and 2S). Compared to ir-HSP (average accuracy of 92.1%), K^* (average accuracy of 98.2%), achieved an improvement of approximately 6.62%. The average of MC classification results in both datasets is listed for K^* in Table 8. Not that K^* results are added because it performed better on overall in all the four stages of classification. DeeperHSP [23] achieved high performance as it utilized a pre-trained protein LM to extract features. The obtained results of FSA4HSP are for five-fold cross-validation. MulCNN-HSP, DeepHSP, DeeperHSP, PredHSP and ir-HSP also used five-fold cross-validation. Some models such as iHSP-PseRAAAC, JPRED and JPPRED, ES-PredHSP employed leave-one-out cross-validation (LOOCV) and ten-fold cross-validation respectively.

Independent test dataset was also used in MulCNN-HSP, ir-HSP, DeepHSP, and DeeperHSP along with cross-validation for the identification of HSPs. We did not evaluate the performance of FSA4HSP on the independent test dataset since the classification models in FSP4HSP undergo evaluation through 5-fold cross-validation. Jackknife of k-fold cross-validation produces results by combining many different independent dataset tests. Therefore, it is not necessary to partition a benchmark dataset into training and testing datasets or utilize an independent test dataset.

5 Conclusion

An analysis and classification method (called FSA4HSP) was proposed for the identification/classification of HSPs. Two datasets, containing amino acid sequences of HSPs, were first formatted appropriately. Then, SPM algorithms were used on the datasets to extract FSPAAs and sequential rules among AAs. Extracted FSPAAs were then used in classification. Four string-based classifiers were used on EAASs of PS and eight integer-based classifiers were used on FSPAAs to perform four stages of classification including both binary and MC classification. Twelve classifiers were assessed by using seven performance evaluation measures. Both binary and MC classification results indicate that classifiers performed much better on FSPAAs as compared to EAASs. The classifier K^* performed the best followed by DT and RF. K^* in FSA4HSP outperformed previous methods for HSP identification/classification and the method allows more interpretability compared to previous methods. The proposed model is not limited to HSPs and can be used on protein sequences of RNA and DNA. Some future research directions are:

- Extending the FSP4HSP method to extract frequent sequential *k-mers* in AA sequence of HSPs and using them for analysis and classification.
- Investigating whether discriminative frequent patterns [43] can be used more effectively, compared to similar frequent patterns, in the classification process.

CRediT author statement

M. Saqib Nawaz: Data Curation, Methodology, Validation, Visualization, Writing - Original Draft, Writing - Review & Editing. **Philippe Fournier-Viger**: Supervision, Formal analysis, Methodology, Validation, Writing - Review & Editing. **Shoaib Nawaz**: Visualization, Investigation. **Wensheng Gan**: Investigation, Writing - Review & Editing. **Yulin He**: Writing - Review & Editing

Conflict of Interest: Authors declare no conflict on interest.

Funding: Authors did not receive funding for this work.

References

1. P. Jacob, H. Hirt, and A. Bendahmane, "The heat-shock protein/chaperone network and multiple stress resistance," *Plant Biotechnology*, vol. 15, pp. 405–415, 2017.
2. P. Poulain, J. C. Gelly, and D. Flatters, "Detection and architecture of small heat shock protein monomers," *PLOS ONE*, vol. 5, p. e9990, 2010.
3. A. Blaszczak, C. Georgopoulos, and K. Liberek, "On the mechanism of ftsh-dependent degradation of the sigma 32 transcriptional regulator of escherichia coli and the role of the dnak chaperone machine," *Molecular Microbiology*, vol. 31, pp. 157–166, 1999.
4. D. Ruggero, A. Ciammaruconi, and P. Londei, "The chaperonin of the archaeon sulfolobus solfataricus is an rna-binding protein that participates in ribosomal rna processing," *EMBO Journal*, vol. 17, pp. 3471–3477, 1998.
5. J. F. Louvion, T. Abbas-Terki, and D. Picard, "HSP90 is required for pheromone signalling in yeast," *Molecular Biology of the Cell*, vol. 9, pp. 3071–3083, 1998.

6. H. Adachi, M. Katsuno, M. Waza, M. Minamiyama, F. Tanaka, , and G. Sobue, "Heat shock proteins in neurodegenerative diseases: pathogenic roles and therapeutic implications," *International Journal of Hyperthermia*, vol. 25, pp. 647–654, 2009.
7. J. E. Hamos, B. Oblas, D. Pulaski-Salo, W. J. Welch, D. G. Bole, and D. A. Drachman, "Expression of heat shock proteins in alzheimer's disease," *Neurology*, vol. 41, pp. 345–350, 1991.
8. Y. R. Wu, C. K. Wang, C. M. Chen, Y. Hsu, S. J. Lin, Y. Y. Lin, H. C. Fung, K. H. Chang, and G. J. Lee-Chen, "Analysis of heat-shock protein 70 gene polymorphisms and the risk of parkinson's disease," *Human Genetics*, vol. 114, pp. 236–241, 2004.
9. R. E. Lackie, A. Maciejewski, V. G. Ostapchenko, J. Marques-Lopes, W. Y. Choy, M. L. Duennwald, V. F. Prado, and M. A. M. Prado, "The HSP70/HSP90 chaperone machinery in neurodegenerative diseases," *Frontier in Neuroscience*, vol. 11, p. 254, 2017.
10. M. G. Goldstein and Z. Li, "Heat-shock proteins in infectionmediated inflammation-induced tumorigenesis," *Journal of Hematology & Oncology volume*, vol. 2, p. 5, 2009.
11. A. G. Pockley, "Heat shock proteins, inflammation, and cardiovascular disease," *Circulation*, vol. 105, pp. 1012–1017, 2002.
12. R. K. Rateesh, N. S. Nagarajan, S. P. Arunraj, D. Sinha, V. B. Veedin Rajan, V. K. Esthaki, and P. D'Silva, "HSPIR: a manually annotated heat shock protein information resource," *Bioinformatics*, vol. 28, no. 21, pp. 2853–2855, 2012.
13. W. Chen, P. Feng, T. Liu, and D. Jin, "Recent advances in machine learning methods for predicting heat shock proteins," *Current Drug Metabolism*, vol. 20, no. 3, pp. 224–228, 2019.
14. R. A. Stetler, Y. Gan, W. Zhang, A. K. Liou, Y. Gao, G. Cao, and J. Chen, "Heat shock proteins: Cellular and molecular mechanisms in the central nervous system," *Progress in Neurobiology*, vol. 92, no. 2, pp. 184–211, 2010.
15. M. Y. Akbar, H. Azad, M. Rashid, W. Ajmal, A. A. Ansari, M. M. Salem, R. K. Sahu, M. M. Salem-Bekhit, and S. Ghazanfar1, "ES-PredHSP: Improved prediction of heat shock proteins using machine learning by enhanced sampling technique," *Journal of Biological Regulators and Homeostatic Agents*, vol. 38, no. 1, pp. 665–673, 2024.
16. G. Zhang, M. Li, Q. Tang, F. Meng, P. Feng, and W. Chen, "MulCNN-HSP: A multi-scale convolutional neural networks-based deep learning method for classification of heat shock proteins," *International Journal of Biological Macromolecules*, vol. 257, no. 128802, 2024.
17. P.-M. Feng, W. Chen, H. Lin, and K.-C. Chou, "iHSP-PseRAAAC: Identifying the heat shock protein families using pseudo reduced amino acid alphabet composition," *Analytical Biochemistry*, vol. 442, pp. 118–125, 2013.
18. P. Feng, H. Lin, W. Chen, and Y. Zuo, "Predicting the Types of J-Proteins Using Clustered Amino Acids," *BioMed Research International*, vol. 935719, 2014.
19. S. Ahmad, M. Kabir, and M. Hayat, "Identification of Heat Shock Protein families andJ-protein types by incorporating DipeptideComposition into Chouâs general PseAACS," *Computer Methods and Programs in Biomedicine*, vol. 122, no. 2, pp. 165–174, 2015.
20. R. Kumar, B. Kumari, and M. Kumar, "PredHSP: Sequence based proteome-wide heat shock protein prediction and classification tool to unlock the stress biology," *PLOS ONE*, vol. 11, p. e0155872, 2016.
21. L. Zhang, C. Zhang, R. Gao, and R. Yang, "JPPRED: Prediction of types of j-proteins from imbalanced data using an ensemble learning method," *BioMed Research International*, vol. 705156, 2015.
22. P. Meher, T. Sahu, S. Gahoi, and A. Rao, "ir-HSP: improved recognition of heat shock proteins, their families and sub-types based on g-spaced di-peptide features and support vector machine," *Frontiers in Genetics*, vol. 8, no. 235, 2018.
23. S. Min, H. Kim, B. Lee, and S. Yoon, "Protein transfer learning improves identification of heat shock protein families," *PLOS ONE*, vol. 16, no. e0251865, 2021.
24. P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, pp. 54–77, 2017.
25. M. S. Nawaz, P. Fournier-Viger, A. Shojaee, and H. Fujita, "Using artificial intelligence techniques for covid-19 genome analysis," *Applied Intelligence*, vol. 53, pp. 3086–3103, 2021.
26. M. S. Nawaz, P. Fournier-Viger, Y. He, and Q. Zhang, "PSAC-PDB: Analysis and classification of protein structures," *Computers in Biology and Medicine*, vol. 158, p. 106814, 2023.
27. T. P. Exarchos, C. Papaloukas, C. Lampros, and D. I. Fotiadis, "Mining sequential patterns for protein fold recognition," *Journal of Biomedical Informatics*, vol. 41, no. 1, pp. 165–179, 2008.
28. P. Cellier, T. Charnois, M. Plantevit, C. Rigotti, B. Cremilleux, O. Gandrillon, J. KlÃ©ma, and J.-L. Manguin, "Sequential pattern mining for discovering gene interactions and their contextual information from biomedical texts," *Journal of Biomedical Semantics*, vol. 6, p. 27, 2015.

29. S. Dubey, D. K. Verma, and M. Kumar, "Severe acute respiratory syndrome coronavirus-2 genoanalyzer and mutagenic anomaly detector using FCMFI and NSCE," *International Journal of Biological Macromolecules*, vol. 258, p. 129051, 2024.
30. M. S. Nawaz, P. Fournier-Viger, S. Nawaz, H. Zhu, and U. Yun, "SPM4GAC: SPM based approach for genome analysis and classification of macromolecules," *International Journal of Biological Macromolecules*, vol. 130984, 2024.
31. K.-C. Chou, "Prediction of protein cellular attributes using pseudo-amino acid composition," *Proteins: Structure, Function, and Bioinformatics*, vol. 43, no. 3, pp. 246–255, 2001.
32. B. Boeckmann, A. Bairoch, R. Apweiler, M. C. Blatter, A. Estreicher, E. Gasteiger, M. J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider, "The swiss-prot protein knowledgebase and its supplement trembl in 2003," *Nucleic acids research*, vol. 31, no. 1, pp. 365–370, 2003.
33. W. Li and A. Godzik, "Cd-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences," *Bioinformatics*, vol. 22, no. 13, pp. 1658–1659, 2006.
34. N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority oversampling technique," *Journal Of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.
35. P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," in *Proceedings of 18th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pp. 40–52, 2014.
36. P. Fournier-Viger, T. Gueniche, S. Zida, and V. Tseng, "ERMiner: Sequential rule mining using equivalence classes," in *Proceedings of 13th International Symposium on Intelligent Data Analysis (IDA)*, pp. 108–119, 2014.
37. P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and 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)*, pp. 36–40, 2016.
38. E. Frank, M. A. Hall, and I. H. Witten, "Mining: Practical Machine Learning Tools and Techniques, fourth edition," Morgan Kaufmann, 2016.
39. R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of Very Large Databases (VLDB)*, pp. 487–499, 1994.
40. L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2657–2605, 2008.
41. C. Etchebest, C. Benros, A. Bornot, A. Camproux, and A. de Brevern, "A reduced amino acid alphabet for understanding and designing protein adaptation to mutation," *European Biophysics Journal*, vol. 36, pp. 1059–1069, 2007.
42. A. de Brevern, C. Etchebest, and S. Hazout, "Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks," *Proteins: Structure, Function, and Bioinformatics*, vol. 41, pp. 271–287, 2007.
43. S. Ventura and J. Luna, "*Supervised Descriptive Pattern Mining*," Springer, 2018.

Comment 17: In what ways do Sequential Pattern Mining (SPM) algorithms vary in their approaches, such as database structure, search strategy, and support measure, and how do these differences impact the analysis of genome sequences?

SPM algorithms vary in terms of (1) whether a horizontal or vertical database is used and specific data structures; (2) the algorithm follows a depth first or breadth first search approach, and (3) the support measure that is used to count patterns in datasets. These aspects effects the algorithm efficiency in-terms of computatinal time and the standard of frequent ptterns discovered by them. In this work, the two most advanced algorithms (CM-SPAM and ERMiner) are used to find sequential patterns and rules in genome sequences.

Classification through discovered frequent patterns Comment 18: How are frequent sequential patterns, discovered through Sequential Pattern Mining (SPM) algorithms, utilized in the classification of genomes?

The frequent sequential patterns obtained with CM-SPAM algorithm are fed to various classifiers for the classification process. Figure 2 provides how some raw

genome sequences in various forms are transofmered and then frequent sequential patterns and rules were obtained. The frequent sequential patterns found in genome sequence indifferent forms can be interpreted as their descriptors or features.

Comment 19: Explain the phases involved in the classification process, including both the training and testing phases?

Two phases are carried out for classification: (1) First is the training phase, which is in turn performed in two steps: (a) frequent nucleotides, codons, or amino acids representation, and (b) classifier training, which is performed sequentially. (2) Then the testing phase has three steps: (a) frequent nucleotides, codons or amino acids representation, (b) hypothesis prediction, and (c) evaluation. This is discussed in the first paragaph of section 3.3. on page 9.

Comment 20: What are the two steps involved in the training phase, and how do they contribute to the overall classification process? Training phase contains two steps: (a) frequent nucleotides, codons, or amino acids representation, and (b) classifier training. Both the steps are performed sequentially to train the classifeirs. This is discussed in the first paragaph of section 3.3. on page 9.

Comment 21: Within the training phase, how are frequent nucleotides, codons, or amino acids represented, and how does this representation aid in classifier training?

In both training and testing phase, frequent nucleotides, codons or amino acids are represented as a sequence off integers. With this encoding, the classifier can capture patterns and relationships in the data that may not be apparent in the raw sequence data.

Comment 22: What are the steps involved in the testing phase, and how do they differ from those in the training phase? Three steps are involved in training phase: (a) frequent nucleotides, codons or amino acids representation, (b) hypothesis prediction, and (c) evaluation. The step b and c of testing phase is different from step b of training as these steps evaluate the performance of trained classifers on unseen data.

Comment 23: How does the classification model address the issue of repetitive occurrences of the same bases or amino acids in genome sequences? Obtained frequent sequential patterns are those that do not have long repetitive occurrence of the same bases, codons or amino acids. These patterns are those that are common in many genome sequences and the sequential nature of the patterns allow them to avoid this repetition. These obtained frequent sequential pattersn are then provided as features in the classification process. This is discussed in the second paragraph of section 3.3. on page 9.

Comment 24: How are frequent sequential patterns of nucleotides, codons, and amino acids utilized to enhance the classification of various RNA virus families? Frequent sequential patterns were able to distinguish various virus families effectively. And thus the classifiers were able to distinguish effectively with frequent sequential patterns. In other words, the frequent sequential patterns extracted from the genome sequences serve as informative features that can help distinguish between different classes or categories in the classification task

Results

Comment 25: Elaborate on the role of scikit-learn in the classification process and its significance in model evaluation? In the classification process, scikit-learn plays important role as a powerful ML library that provides a wide range of tools and

algorithms for building and evaluating classification models. Its significance lies in its user-friendly interface, extensive documentation, and efficient implementation of various classification algorithms such as SVM, decision trees, and random forests. Scikit-learn simplifies the process of model training, testing, and evaluation through its standardized API and built-in functions for model performance metrics, cross-validation, and hyperparameter tuning.

Comment 26: How were the datasets divided into training and testing subsets, and what was the rationale behind using an 80The datasets were divided into training and testing subsets using a common practice known as the 80/20 split. This involves allocating 80% of the data for training the model and 20% for testing its performance. The rationale behind this split is to strike a balance between having enough data to train the model effectively and having a sufficient amount of unseen data to evaluate its generalization performance. By using an 80/20 split, we aim to ensure that the model learns from a diverse set of examples during training while also being adequately tested on a separate subset to assess its ability to generalize to new, unseen data. This approach helps prevent overfitting and provides a reliable estimate of the model's performance on real-world data.

Comment 27: Provide more details on the default hyperparameters used for classifiers during the classification process?

We used default hyperparameters of classifiers to simplify the model training process and provide a baseline performance for comparison. The default hyperparameters of each classifier is listed in Table 2.

Comment 28: How did the utilization of SPMF and Python libraries contribute to the reliability and efficiency of the experimental process?

SPMF library is used to find the frequent sequential patterns and rules. Obtained frequent sequential pattersn are then fed to classifietrs offered by sciki-learn library developed in python.

Comment 29: Explain how the Apriori, CM-SPAM, and ERMiner algorithms were applied to find compositions of nucleotides, codons, or amino acids, frequent sequences, and sequential rules within genome sequences? SPMF software offers implementation of more than 230 SPM algorithms, Apriori , CM-SPAM and ERminer were run on the developed datasets by using SPMF software. Comment 30: In the provided example (Figure 2), how are raw genome sequences transformed and represented for analysis by Apriori, CM-SPAM, and ERMiner Figure 2 also provides the transformation of raw genome sequences. First a raw genome sequence in NF/CRF, PF and codons are presented at the top of each box in Figure 2, followed by nucleotides, amino acids and codons transformations. In the transformation, each distinct base/amino acid/coodn is replaced with a unique integer. The seperator -1 is used between bases/amino acids/codons and -2 is used at the end of the sequence to show that the genome sequence has ended.

Comment 31: Can you describe the significance of the compositions, frequent sequential patterns, and sequential rules identified within genome sequences, as illustrated in Figure 2? The composition of nucleotide bases and amino acids in genome sequences provides insights about their occurrence count and their percentage. In this work, discovered frequent sequential patterns of nucleotides/amino acids and codons are used as features in the classification process. Sequential frequent patterns are

those patterns that are present in databases and their support (occurrence count) is equal to or greater than a support measure, specified by the user. Discovered sequential rules provides insights about the relationship among nucleotide/amino acids and codons. Sequential rules can also be used as features in the classification process.

Comment 32: How do the frequent sequential patterns and sequential rules contribute to characterizing and describing the sequences within genome data? Sequential frequent patterns are those patterns that are present in databases and their support (occurrence count) is equal to or greater than a support measure, specified by the user. Whereas, sequential rules are those rules that are present in databases and their support (occurrence count) and confidence is equal to or greater than the minsupt and mincof measures, specified by the user. The frequent sequential patterns of bases/amino acids and codons can be interpreted as their descriptors or features that characterize or distinguish the genome sequences compared to others. The frequent sequential patterns extracted from the genome sequences serve as informative features that can help distinguish between different classes or categories in the classification task