Current Genomics, 2022, *23*, 353-368

RESEARCH ARTICLE

A Deep Clustering-based Novel Approach for Binning of Metagenomics Data

Sharanbasappa D. Madival[1], Dwijesh Chandra Mishra[1,*], Anu Sharma[1], Sanjeev Kumar[1], Arpan Kumar Maji[2], Neeraj Budhlakoti[1], Dipro Sinha[1] and Anil Rai[1]

[1]*Division of Agriculture Bioinformatics, ICAR-IASRI, New Delhi- 110012, India;* [2]*Division of Computer Applications, ICAR-IASRI, New Delhi- 110012, India*

ARTICLE HISTORY

Received: July 04, 2022
Revised: August 30, 2022
Accepted: September 02, 2022

DOI:
10.2174/1389202923666220928150100

Abstract: ***Background*:** One major challenge in binning Metagenomics data is the limited availability of reference datasets, as only 1% of the total microbial population is yet cultured. This has given rise to the efficacy of unsupervised methods for binning in the absence of any reference datasets.

***Objective*:** To develop a deep clustering-based binning approach for Metagenomics data and to evaluate results with suitable measures.

***Methods*:** In this study, a deep learning-based approach has been taken for binning the Metagenomics data. The results are validated on different datasets by considering features such as Tetra-nucleotide frequency (TNF), Hexa-nucleotide frequency (HNF) and GC-Content. Convolutional Autoencoder is used for feature extraction and for binning; the K-means clustering method is used.

***Results*:** In most cases, it has been found that evaluation parameters such as the Silhouette index and Rand index are more than 0.5 and 0.8, respectively, which indicates that the proposed approach is giving satisfactory results. The performance of the developed approach is compared with current methods and tools using benchmarked low complexity simulated and real metagenomic datasets. It is found better for unsupervised and at par with semi-supervised methods.

***Conclusion*:** An unsupervised advanced learning-based approach for binning has been proposed, and the developed method shows promising results for various datasets. This is a novel approach for solving the lack of reference data problem of binning in metagenomics.

Keywords: Binning, convolutional autoencoder, deep clustering, metagenomics, genomic features, K-means.

1. INTRODUCTION

Microorganisms are microscopic organisms that cannot be seen through our naked eyes. These organisms are Omni present, *i.e.* from a classroom to hot springs. It is a well-known fact that one gram of soil may contain 18,000 different organisms having their genome. The role of these microorganisms is huge in our ecosystem, especially in agriculture; the soil microbes perform an essential role by maintaining a symbiotic relationship with the plants. Apart from this, these microorganisms' significant roles are in animal, fish and poultry science. The term 'Metagenomics' was first coined by Handelsman in 1998 [1]. Metagenomics can be defined as the direct genetic analysis of genomes contained in an environmental sample [2]. Metagenomics plays an important role in enabling researchers to study microorganisms in-vivo. Metagenomics study provides more exposure to the genetic information of those microbes and tells the better way to utilise them. However, Metagenomics data containing different strains have almost identical genetic constitutions and genome architecture. So it is not possible to isolate each strain separately. Binning indicates the process of classifying DNA sequences into clusters that might be the true representative of an individual genome or genomes from taxonomically related microorganisms. It has been used to reconstruct the genomes of individual species from communities, including samples from complex environments such as sediments and reactors. Binning can be classified in two ways, *i.e.* taxonomy dependent binning and taxonomy independent binning. Taxonomy Dependent Binning methods are subject to sufficient levels of similarity between reads and sequences/models in reference databases. Taxonomic Independent Binning methods simply group/bin reads in a given dataset based on their mutual similarity and do not involve a database comparison step [3]. One of the major challenges in taxonomy-dependent binning of Metagenomics data is the limited availability of reference datasets as only 1% of the total microbial population is yet cultured. In case of the unavailability of a reference genome, binning has to be performed in an unsupervised manner.

*Address correspondence to this author at the Division of Agriculture Bioinformatics, ICAR-IASRI, New Delhi- 110012, India;
E-mail: dwij.mishra@gmail.com

1875-5488/22 $65.00+.00

The unsupervised binning approach uses genomic signatures such as GC contents, tetramer compositions, coverage profile, taxonomic position analysis *etc*. for clustering the fragmented genomes. Using k-mers (4 or 6) also facilitates biological advantages, such as the detection of genomic biases resulting from the observed avoidance of specific palindromic words of length k from genomes [4, 5]. An extensive study of the previous work showed that the distribution of Guanine and Cytosine content in each genome is unique. Hence, GC content is an essential parameter for distinguishing the microbial community [6-8].

Further from the literature survey, it was evident that although several techniques like k-means, SOM, and Hierarchical Clustering have been used for binning Metagenomics data, input data's high dimensionality poses many challenges to statistical and computational bottlenecks in their execution. The available dimensionality reduction techniques such as Principle Component analysis (PCA) [9], Correspondence Analysis (CA) [10], and Multidimensional Scaling (MDS) [11] have been applied in such cases. But these techniques lead to a loss of information

With the advent of advanced machine learning techniques such as Support Vector Machine [12], Random Forest [13], Neural Network-based deep learning [14] *etc*., this problem can be resolved. Among all these techniques, the performance of deep learning is the best. However, these machine learning techniques are primarily supervised learning techniques that can only be applied when a reference dataset is available. An Autoencoder is an unsupervised learning technique based on deep learning architecture that learns efficient data representations (encoding) by training the network to ignore signal "noise". Deep clustering [15] is an autoencoder-based approach used to extract the informative features from the available dataset and makes the data with reduced dimensions suitable for binning. However, their applicability in binning of Metagenomics data is still unexplored. To fill this gap, we proposed a study that uses a Convolutional Autoencoder (CAE) based deep clustering technique for binning the Metagenomics data that combines dimensionality reduction, feature selection, and clustering in a single framework.

This paper describes a novel convolutional autoencoder-based unsupervised method for binning metagenomics data. The significant contributions discussed in this paper are:

i. Novel Convolutional Autoencoder (CAE) based deep clustering technique for taxonomic independent binning for metagenomics dataset is developed.

ii. Most of the work is done using Tetra-nucleotide frequency, but this work includes Hexanucleotide frequency as well.

iii. The optimum k value of binning is decided by plotting the WSS plots, also known as elbow curve plots.

The organisation of the paper is studied under different sections as follows. The introduction section is followed by the details of the materials and methodology section for achieving the objectives set for the study. Section 3 gives the detailed results obtained from the study. This also includes the evaluation parameters such as the Rand index, Silhouette index, Accuracy, Precision, Recall, F1 score and comparison of progressive approaches using state-of-the-art techniques in tabular form. The result section is followed by a discussion and the last entire study is summarised in the conclusion section. This manuscript ends with references, acknowledgement, conflict of interest and data availability.

2. MATERIALS AND METHODS

2.1. Materials

This section includes the various programming languages such as Python, PERL and R software and dataset and the details of the validation techniques used in this study to evaluate the developed approach.

2.1.1. Dataset

The data used for this study was downloaded from the MyCC section of the SOURCEFORGE website (https://sourceforge.net/projects/sb2nhri/files/MyCC/Data/). Three metagenomics datasets have been taken for study, *viz*. 10s [16], Sharon [17] and 25s. 10s and 25s datasets are simulated datasets of ten and twenty-five are already known species; Sharon is a real dataset containing 32 unknown species.

2.2. Methods

2.2.1. Feature Matrix Generation

Features like TNF, HNF and GC content are calculated from the FASTA files of different datasets using metaCluster R package [18]. There are six cases (two cases under each dataset) based on various combinations of these features have been studied. These cases are as follows:

Case1: TNF + GC under 10's dataset

Case2: HNF + GC under 10's dataset

Case3: TNF + GC under 25's dataset

Case4: HNF + GC under 25's dataset

Case5: TNF + GC under Sharon dataset

Case6: HNF + GC under Sharon dataset

2.2.2. Pre-processing

In this, noise present in the dataset is removed, and the feature matrix is resized into different configurations without a loss of features. Resizing is done because the Feature matrix with 4097×4097(HNF-GC) and 257×257(TNF×GC) will require a lot of computational power and storage capacity. So to make this process easy, resizing the dataset into smaller sizes is done without losing any features and keeping all features intact. In this study, in all cases, we resized the feature matrix into 64×64.

2.2.3. Deep Clustering

Deep clustering frameworks combine feature extraction, dimensionality reduction and clustering into an end-to-end model, allowing the convolutional neural networks to learn suitable representations to adapt to the clustering module's assumptions and criteria used in the model. Deep clustering can be broadly categorised into three categories: Autoencoder-based, Generative model-based and direct optimization-based. In this study, Convolutional Autoencoder (CAE) based deep clustering has been used.

2.2.4. Convolutional Autoencoder (CAE)

Convolutional Autoencoder (CAE) [19] is an Autoencoder-based architecture in which the encoder and decoder are embedded with CNN layers instead of ANN. Autoencoder is an unsupervised deep learning algorithm that takes data as input and tries to reconstruct it using a number of bits from the bottleneck, also known as latent space. Autoencoder works similarly to PCA, but the difference between them lies in the transformation part, *i.e*. PCA uses linear transformation whereas Autoencoder uses non-linear transformations. Because of slicing and stacking the data, a huge amount of information is lost. Instead of stacking the data, the Convolution Autoencoder (CAE) keeps the spatial information about the input data as they are and extract required unique features gently in the Convolution layer. Fig. (**1**) demonstrates that 2-dimensional data is extracted into a thick square (Conv1), and then it continues to become a long cubic in the second layer (Conv2) and another longer cubic in the third layer (Conv3). This process is designed in such a way that it retains the maximum spatial relationships in the data.

Convolution Autoencoder works in 3 different layers Convolutional layer, Activation function and Max pooling layer.

2.2.4.1. Convolutional Layer

This step creates many small pieces of data called feature maps or features. These squares preserve the relationship between input data within the dataset. Here, each feature is scanned through the original dataset. This process produces the scores called filtering. After scanning, each feature produces filtered data with high and low scores. If there is a perfect match, there is a high score in that square. The score is low or zeroes if there is a less or no match. Strides are another parameter of the convolutional layer. It is the number of cells shifting over the input matrix. When we keep the stride value as 1, the filters shift one cell at a time. In this study, we kept strides=2 and filter sizes as 32, 64, 128 and kernel_size (5,5,3) fixed.

2.2.4.2. Activation Functions

An activation function is added to an artificial neural network to help the network learn complex patterns of the data. Different activation functions are used, such as the Step function, ReLU, Leaky ReLU, and Sigmoid activation function. In this study, we used the ReLU activation function.

2.2.4.3. Max Pooling Layer

Pooling compresses the dataset size. If window size 2x2 is called the pool size, which scans through each of the

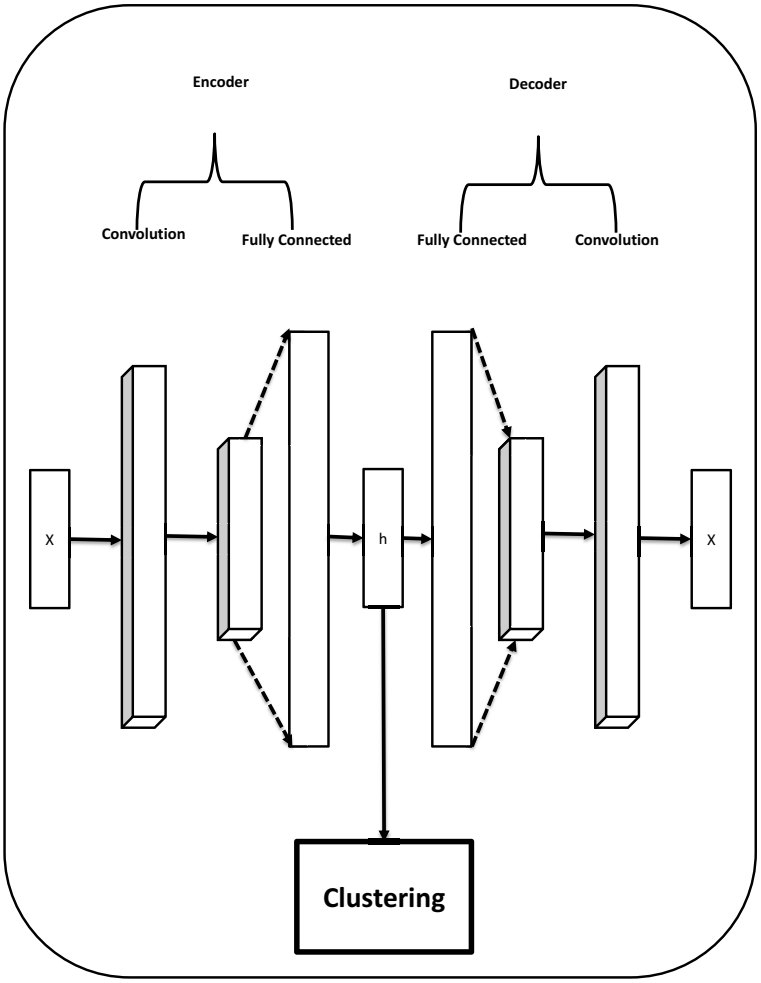

Fig. (1). Basic architecture of convolutional autoencoder.

filtered matrices and assigns the maximum value of that 2x2 window to a 1x1 square in a new cell. The maximum value in the 2 x 2 window is the highest score, so the highest score is assigned to the 1 x 1 square. After pooling, a new stack of a smaller filtered matrix is produced. After that, splitting the smaller filtered cells and stacking them into a list is done.

The above three layers are building blocks in convolutional layers and are implemented in Keras [20] library. One can build many convolution layers in the Convolution Autoencoder. There are three layers labelled Conv1, Conv2, and Conv3 in the encoding part. So, we built it accordingly. Notice that Conv1 is inside Conv2 and Conv2 is inside Conv3. Then it continues to add the decoding process. The Keras API requires the declaration of the model and the optimisation method. The model includes all layers required in the computation of the decoded outputs given the input data and then compiled. Mean Square Error (MSE) is a loss function that computes the distance between the current work of the algorithm and the expected output. It is used to evaluate how your algorithm models the data. Adam is the type optimiser used to change the attributes of neural networks, such as weights and learning rate, to reduce losses.

2.2.5. Training the Model

We had given pre-processed feature matrix as an input. The batch_size is the number of samples, and the epoch is the number of iterations. We specify shuffle=True to require shuffling the train data before each epoch. We kept some parameters constant such as optimiser= Adam, loss function=, Mean square error for all kinds of datasets, number of features=100 by varying number of epochs and batch size. After completion of training, features with reduced dimensions are stored in the flattened (hidden) layer of CAE. These features are used for clustering purposes. In the case of the TNF+ GC feature matrix case, we kept batch size to 10 and epochs to 50, as only a limited number of features are there. For the HNF+ GC feature matrix, we kept batch size to 100 and epochs to 350 because more features and observations data were generated.

2.2.6. Clustering

Clustering is the task of dividing the population or data points into several groups such that data points in the same group are more similar to other data points in the same group and dissimilar to the data points in other groups. It is a collection of objects based on similarity and dissimilarity between them. There are different types of clustering, such as hierarchical clustering, Gaussian mixture model clustering, k-means, k-medoid spectral clustering, and DBSCAN [21] clustering *etc*. In this study, k-means clustering was applied to the encoded output obtained from CAE.

2.2.7. WSS Plot

The first step was estimating the number of clusters required for effective data analysis. For this purpose, the Elbow method is used. The K-means Clustering works by defining the clusters so that the total variation within a given cluster is minimum [22]. WSS denotes the total within the cluster sum of squares, indicating how compact the cluster is. Our aim while using the clustering method is to keep the WSS as small as possible. This leads to the formation of effective clusters for the data analysis. The Elbow method is a method that considers the total WSS as a function of the number of clusters suitable for the particular dataset and enables us to choose the appropriate number of clusters in such a way that adding more clusters does not have any impact on the data analysis results

Within cluster sum of squares: $\mathrm{WSS} = \sum_{i=1}^{N_c} \sum_{x \in c_i} d\left(x, \bar{x}_{c_i}\right)^2$

Between cluster sum of squares: $\mathrm{BSS} = \sum_{i=1}^{N_c} |C_i| \, . \, d\left(x, \bar{x}_{c_i}\right)^2$

$C_i =$ i^{th} Cluster, N_c= #clusters, $\bar{\mathbf{x}}_{c_i}$ =cluster centroid, $\bar{\mathbf{x}}_{c_i}$ =sample mean.

WSS means the sum of distances between the points and the corresponding centroids for each cluster and BSS means the sum of distances between the centroids and the total sample mean multiplied by the number of points within each cluster. So, WSS is the measure of compactness, and BSS is the measure of separation. For clustering to be successful, we need to get the lower WSS and the higher BSS. WSS plots the graph between a number of clusters and the total within-cluster variance. Based on this plot, we decide on optimum clusters.

2.3. Evaluation of the Developed Approach

Rand index, Silhouette index, Accuracy, Precision, recall and F1 Score were used to evaluate the quality of clusters.

2.3.1. Rand Index

The Rand index [23] is a measure of the similarity between two data clustering approaches.

Given a set S of n elements S= {O1, O2, … , On} with two subsets S1 and S2 partitioned of S to compare S1 = {1,2, … , n}, a partition of S into R subsets, and S2 = {1, 2, … , n}, a partition of S into s subsets [23], define the following:

a- Number of pairs in an element of S that are in the same subset of X and in the same subset of Y.

b- Number of pairs in an element of S that are in the different subset of X and in the different subset of Y.

c- Number of pairs in an element of S that are in the same subset of X and in the different subset of Y.

d- Number of pairs in an element of S that are in the different subset of X and in the same subset of Y.

$$\text{Rand Index} = \frac{a+b}{a+b+c+d} \quad (1)$$

Intuitively, a + b can be considered as the number of agreements between X and Y, and c + d as the number of disagreements between X and Y. R ranges between 0 to 1. The higher R-value indicates better clustering.

Rand index is calculated when we have both actual data and predicted data. Due to the lack of reference data, we labelled the predicted clusters to prepare our real data. The remaining sequences of other species are clustered under one set, and a group with a maximum number of sequences from a single species is numbered appropriately.

2.3.2. Silhouette Index

It measures how similar a contig is to its bin (cohesion) compared to other bins (separation). For example, If an (i) is the average distance of contig's to other coting's in the same bin and b(i) is the average distance of contig i to the contig's in its nearest neighbour bin, then the Silhouette index is given by –

$$S(i) = \frac{(b(i)-a(i))}{\max[a(i),b(i)]} \quad (2)$$

Where,

- $a(i)$ is the average dissimilarity of the i^{th} contig to all other contigs in the same bin.
- $b(i)$ is the average dissimilarity of i^{th} contig with all contig in the closest bin.

S value ranges between [-1, 1]. If the Silhouette index value is high, the contig's well-matched to its own bin and poorly matched to neighbouring bins.

2.3.3. Confusion Matrix

This is used to describe the performance of a classification model on a set of test data for which the actual values are known.

Table 1. Showing species distribution among 7 clusters in Case 1 (TNF+GC) in the 10s dataset.

-	**Predicted**		
Actual	-	Class = Yes	Class = No
	Class = Yes	True Positive(TP)	False Negative(FN)
	Class = No	False Positive(FP)	True Negative(TN)

Accuracy is the most intuitive performance measure and is simply a ratio of correctly predicted observations to the total observations.

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \quad (3)$$

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations.

$$\text{Precision} = \frac{TP}{TP+FP} \quad (4)$$

Recall (Sensitivity) - It is the ratio of correctly predicted positive observations to all observations in actual class

$$\text{Recall} = \frac{TP}{TP+FN} \quad (5)$$

F1 score - It is the weighted average of Precision and Recall.

$$\text{F1 score} = \frac{2*(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \quad (6)$$

In the case of multi-class classification, we adopt averaging methods for Precision, Recall, and F1 score calculation, resulting in a set of different average scores (macro, weighted, micro) in the classification report. The macro-averaged is computed using the arithmetic mean (aka unweighted mean) of all the per-class. The weighted average is calculated by taking the mean of all per-class scores while considering each class's support. Micro averaging computes a global average F1 score by counting the sums of the True Positives (TP), False Negatives (FN), and False Positives (FP). In the case of an imbalanced dataset, the weighted average gives more accuracy than all the other averages. Fig. (**2**) explains the workflow of the developed approach in a crispy manner.

3. RESULTS

3.1. 10s Dataset

This dataset contains assembled contig sequences of 10 species in FASTA format, which includes nine bacterial and one archaebacterial species.

Features like TNF, HNF and GC Content are calculated by using metaCluster R package and then generation of combined feature matrix file using the vlookup function followed by pre-processing of the file. This generated feature matrix of sizes 3256×257(case 1), and 3256×4097 (case 2) is resized to 64×64 without losing any features. For this purpose, libraries of Python like NumPy [24], pandas [25], sklearn [26], and Metrics were used. This pre-processed data is used as input to the development of the CAE model. CAE model was developed using available libraries like Keras and Tensorflow in Python. The features with lower dimensions stored in flattened (hidden) layers are used for clustering, which is done using sklearn.kmeans library in Python. The optimum number of clusters for this dataset was obtained using the WSS plot function, which plots the within-cluster sum of squares (WSS) *vs*. the number of clusters, and Clustering results are visualised by plotting a scatter plot (true features *vs*. predicted features) using the matplotlib [27] library in Python. The same procedure has been followed in all cases.

In case 1 and case 2, TNF + GC content and HNF + GC content feature matrix of the 10s dataset is used, where one expects 10 clusters, but we got only around seven and eight groups, respectively (Tables **1** and **2**). It is found that some species occur abundantly in a single group, and one species occurs in more than one bin. For example, *Crocosphaera subtropica* occurs in two clusters (Tables **2** and **3**). It was found that within-cluster variation is much less than between-cluster variation. Looking into WSS plots, one can predict the optimum number of clusters using the elbow method, but it depends on how sharply one can judge. In our case, we considered them as seven and eight, respectively. Figs. (**3** and **4**) illustrate the optimum k value and scatter plots for case 1, while Figs. (**5** and **6**) show the optimal k value and scatter plots for case 2, respectively.

3.2. 25s Dataset

This dataset contains 25 species assembled scaffold sequences in FASTA format, which includes 25 different species of bacteria. The same procedure is followed here also. Here expected number of clusters is 25, but we got around 10 and 14 groups in case 3 and case 4, respectively. There seems to be an abundance of *Escherichia coli* and *Salmonella bon-*

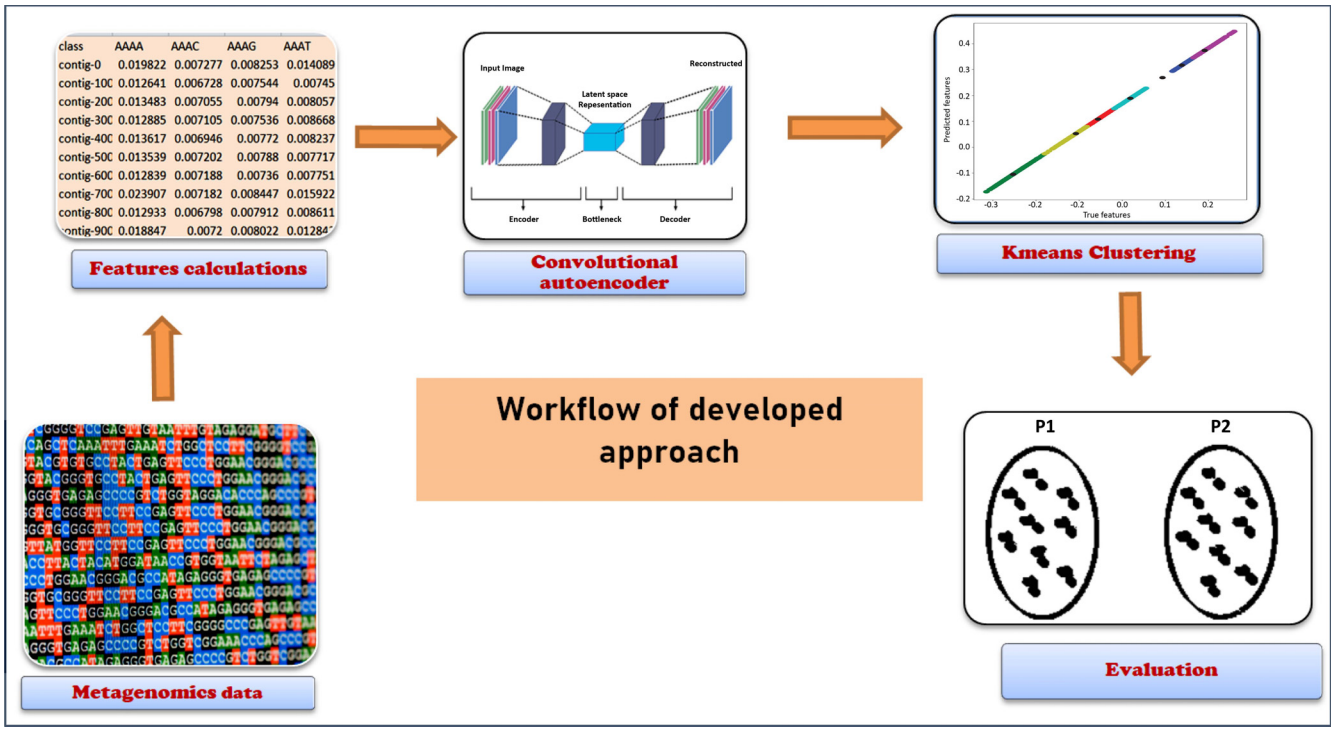

Fig. (2). Workflow of the developed approach. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

gori in the dataset. Hence they appear in more than one bin (Tables **4** and **5**). Figs. (**7** and **8**) illustrate the optimum k value and scatter plots for case 3, while Figs. (**9** and **10**) show the optimal k value and scatter plots for case 4, respectively.

Table 2. Showing species distribution among 7 clusters in Case1 (TNF+GC) in the 10s dataset.

Cluster Number	Accession Number	Species Abundance
1	*NC_003112*	*Neisseria meningitidis*
2	*NC_010546*	*Crocosphaera subtropica*
3	*NC_005296*	*Rhodopseudomonas palustri*
4	*NC_006582*	*Bacillus clausii*
5	*NC_007779* *NC_003112*	*Escherichia coli* *Neisseria meningitidis*
6	*NC_010546*	*Crocosphaera subtropica*
7	*NC_009641*	*Staphylococcus aureus*

3.3. Sharon Dataset

This dataset contains 32 species assembled sequences in FASTA format, including different genera and bacteria species. Here expected number of clusters is 32, but we got around 6 and 10 groups in case 5 and case 6, respectively. The results in the Sharon dataset are not so convincing because there seems to be an abundance of just 4-5 species out of 32 species. Because of this, the silhouette index is more advantageous in this situation than the accuracy and rand index. CARCAL, CARFMA, CARANA, CARLCI, CARLSTR, and CARPAC cover most of the groups, and the remaining organism sequences are distributed falsely among all the groups (Tables **6** and **7**). Figs. (**11** and **12**) illustrate the optimum k value and scatter plots for case 5, while Figs. (**13** and **14**) show the optimal k value and scatter plots for case 6, respectively.

Table 3. Showing species distribution among 8 clusters (HNF+GC) for the 10s dataset.

Cluster Number	Accession Number	Species Abundance
1	NC_010546	*Crocosphaera subtropica*
2	NC_008555 NC_009641	*Listeria welshimeri* *Staphylococcus aureus*
3	NC_005296 NC_007404	*Rhodopseudomonas palustri* *Thiobacillus enitrificans*
4	NC_009637	*Methanococcus maripaludis*
5	NC_007779 NC_003112	*Escherichia coli* *Neisseria meningitidis*
6	NC_005296	*Rhodopseudomonas palustri*
7	NC_003112 NC_007779 NC_010546	*Neisseria meningitides* *Escherichia coli* *Crocosphaera subtropica*
8	NC_006582	*Bacillus clausii*

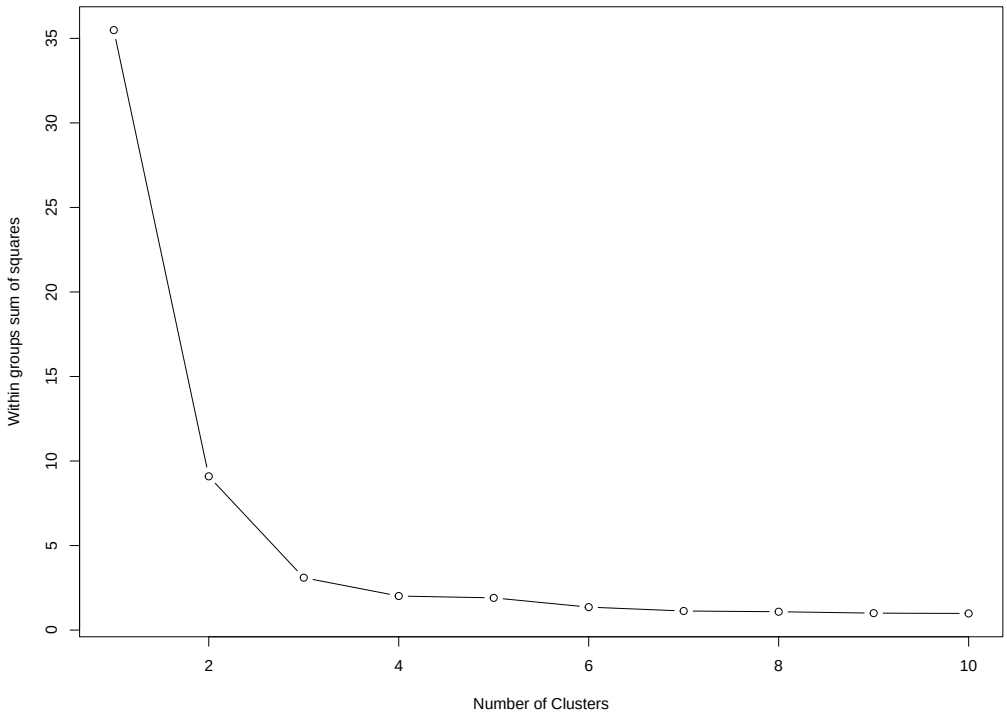

Fig. (3). WSS plot for case 1.

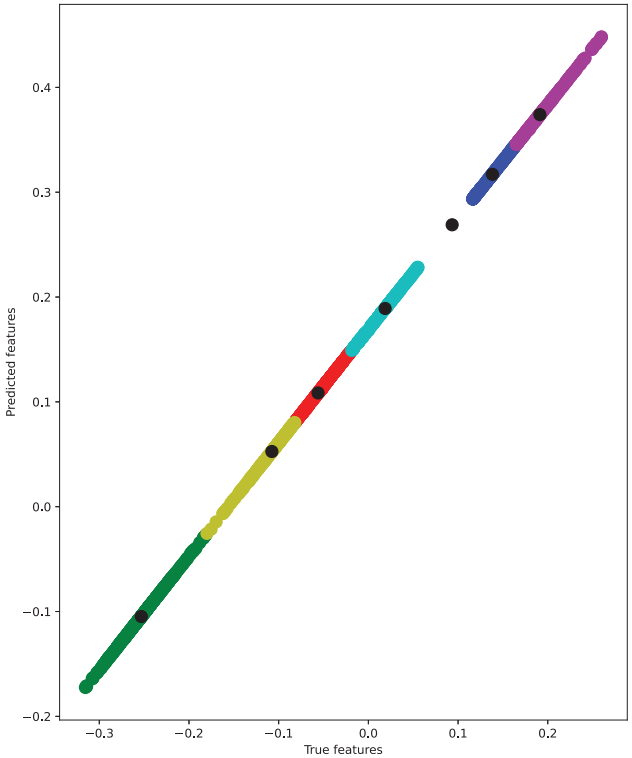

Fig. (4). Showing distribution of contigs among bins for case 1. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

3.4. Evaluation

The proposed method was compared with unsupervised Coverage and composition based binning of Metagenomes (CoMet) [28] tool for a single metagenomics sample and MetaConClust [29] - Unsupervised Binning of Metagenomics Data Using Consensus Clustering and semi-supervised tools MetaBAT [30] and MaxBin [31]. For evaluation of the quality of clusters, silhouette index, Accuracy, specificity, Precision and recall were used. Silhouette index, Accuracy, specificity, Precision and recall were calculated by using the SKlearn library in Python. Rand index was computed using R packages like phyclust [32], caret [33] and pROC [34] in R.

Table 4. Showing species distribution among 10 clusters (TNF+GC) for the 25s dataset.

Cluster	Species Abundance
1	*Salmonella bongori*
2	*Meiothermus silvanus*
3	*Escherichia coli*
4	*Clostridium thermocellum*
5	*Clostridium perfringens, Clostridium thermocellum*
6	*Escherichia coli, Salmonella bongori*
7	*Escherichia coli, Salmonella bongori*
8	*Desulfosporosinus acidophilus*
9	*Meiothermus silvanus*
10	*Escherichia coli, Salmonella bongori*

Table 5. Showing species distribution among 14 clusters (HNF+GC) for the 25s dataset.

Cluster	Species Abundance
1	*Natronobacterium gregoryi*
2	*Segniliparus rotundus*
3	*Desulfosporosinus meridiei*
4	*Escherichia coli, Salmonella bongori*
5	*Terriglobus roseus*
6	*Clostridium perfringens*
7	*Corynebacterium glutamicum, Coraliomargarita akajimensis*
8	*Natronococcus occultus, Halovivax ruber*
9	*Salmonella enterica, Thermobacillus composti*
10	*Pseudomonas stutzeri, Frateuria aurantia*
11	*Olsenella uli*
12	*Clostridium perfringens*
13	*Clostridium thermocellum, Desulfotomaculum gibsoniae*
14	*Hirschia baltica*

Accuracy is a measure for supervised learning techniques; our method is unsupervised. Therefore accuracy measure is not very suitable for comparing our method to other techniques. However, we have done this with some crude approaches. It is found that the developed approach is showing more promising results than existing reference-free learning approaches. The results are shown in Table **8**. A comparison with various existing approaches is shown in Table **9**, and a comparison with only reference-free approaches is shown in Table **10**.

Limitations include the requirement of extensive data for training, complexity in understanding models and may be chance of missing important features. Moreover, deep learning requires expensive GPUs. This increases the cost to the users. When a species is exceedingly rare, there is still room for improvement. Organisms with a low contig contribution in the dataset cannot be grouped.

4. DISCUSSION

This study used three standard benchmark datasets: 10s, 25s and Sharon species. For convenience, two cases under each dataset were studied depending on the combination of features such as TNF+GC and HNF+GC. In this study, we know that there are ten species in the case of the 10s dataset, 25 species in the case of the 25s dataset and 32 species in the case of the Sharon dataset. Ideally, we should get 10 clusters, 25 clusters, and 32 clusters; however, we got only seven and eight, ten and fourteen, six and ten optimum clusters for 10s, 25s and Sharon datasets with TNF+GC and HNF+GC features, respectively. The possible reason is that some very closely related species fall under the same cluster as per the phylogenetic analysis. A total of seven optimum clusters were found in the 10s dataset using TNF+GC features based on the proposed method, with a silhouette index of roughly 0.60, a rand index of 0.95, and an accuracy of 0.86. In contrast, eight optimum clusters were found when utilising HNF+GC features, with a Silhouette Index, Rand Index, and Accuracy of

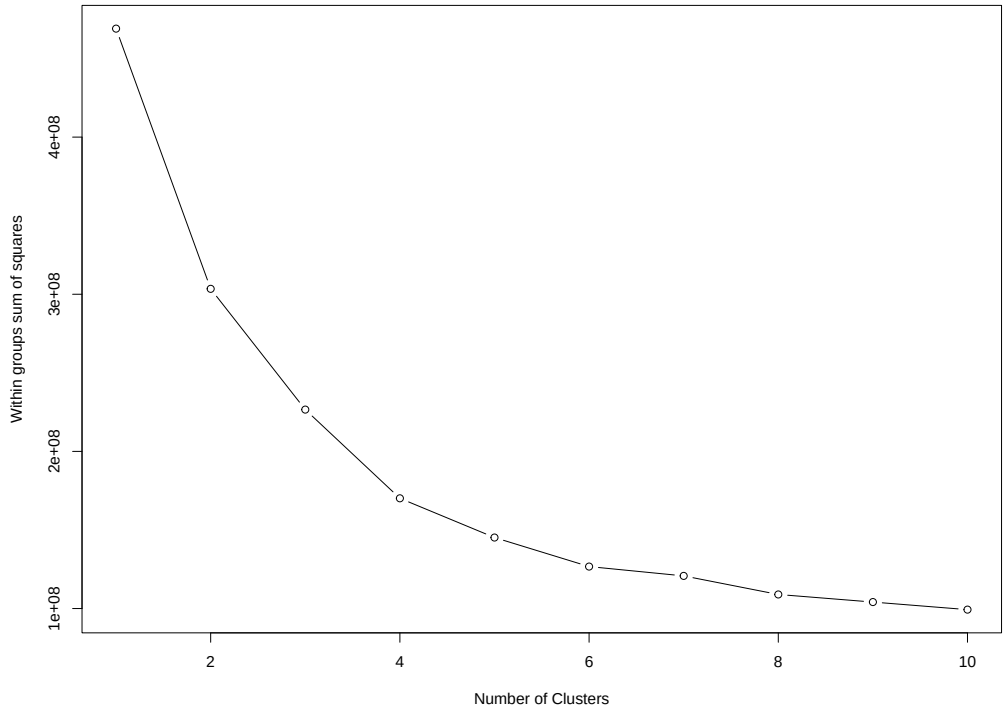

Fig. (5). WSS plot for case 2.

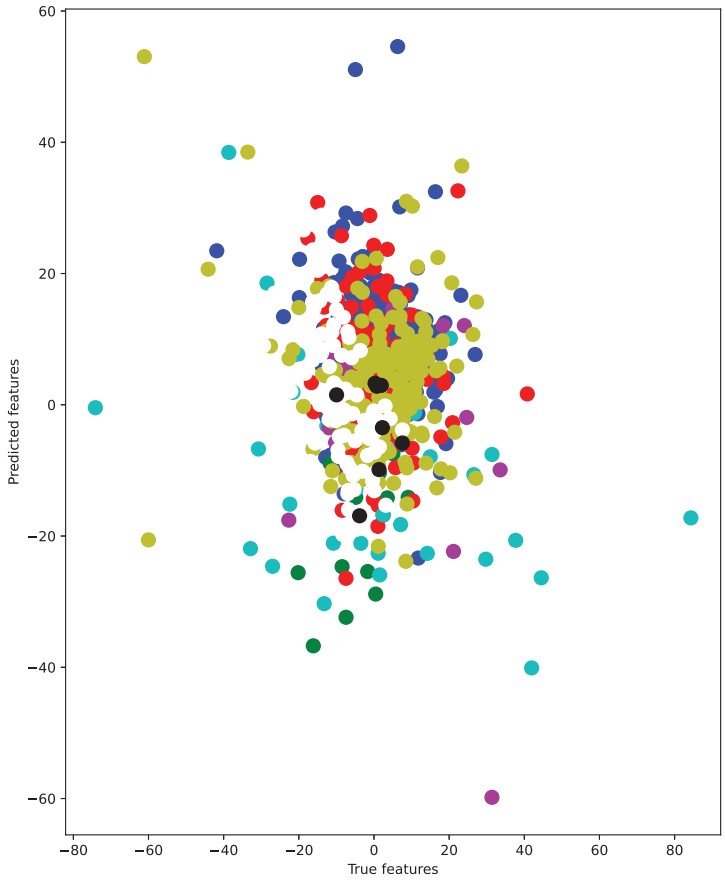

Fig. (6). Showing distribution of contigs among bins for case 2. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

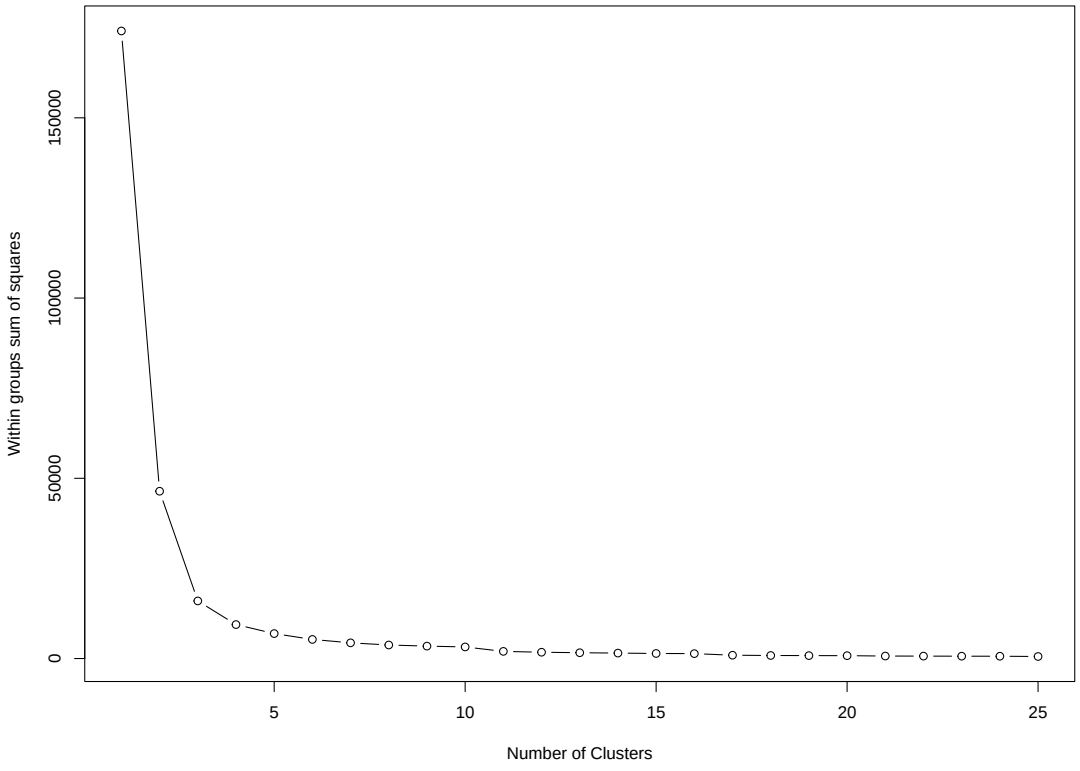

Fig. (7). WSS plot for case 3.

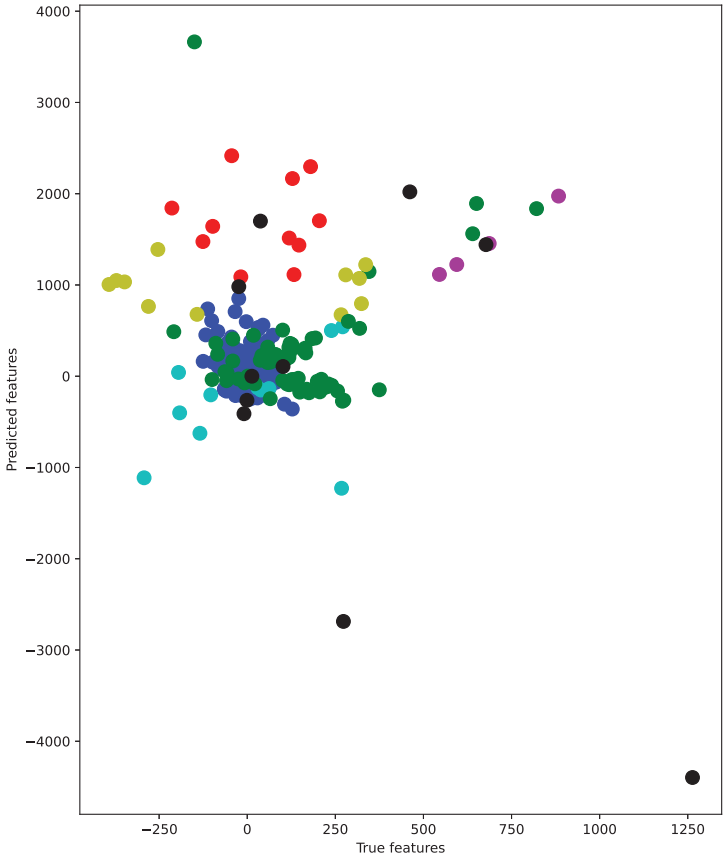

Fig. (8). Showing distribution of contigs among bins for case 3. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

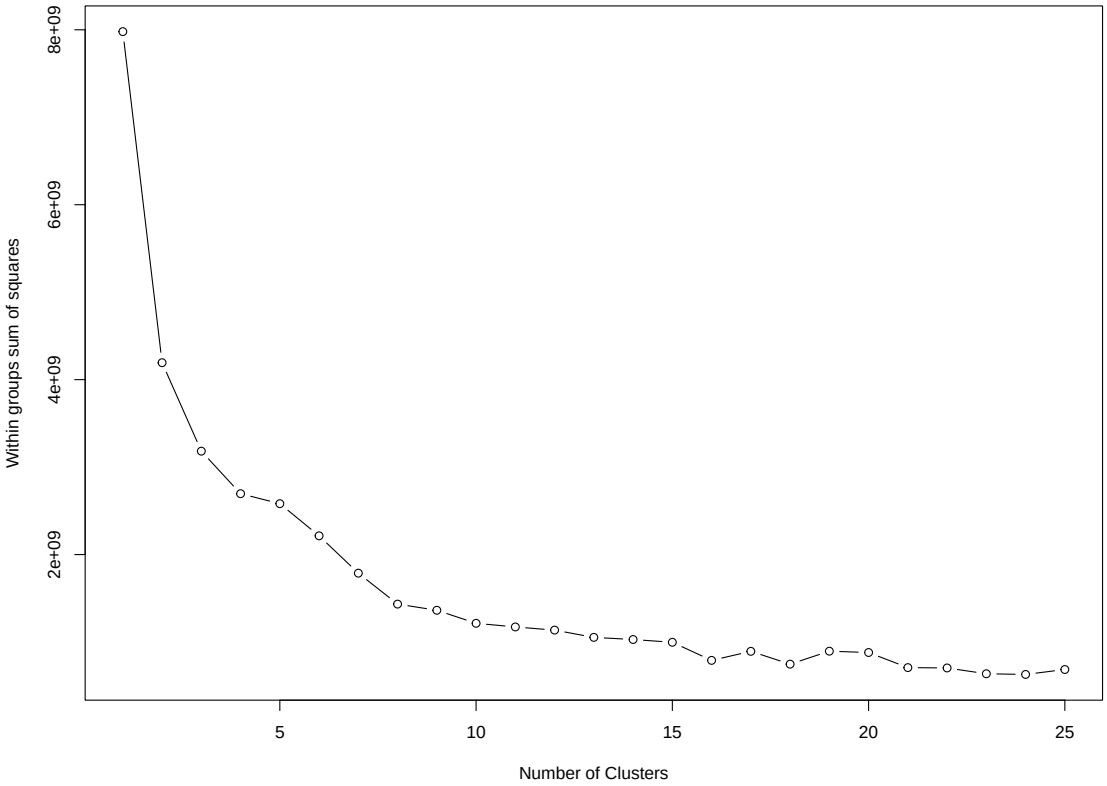

Fig. (9). WSS plot for case 4.

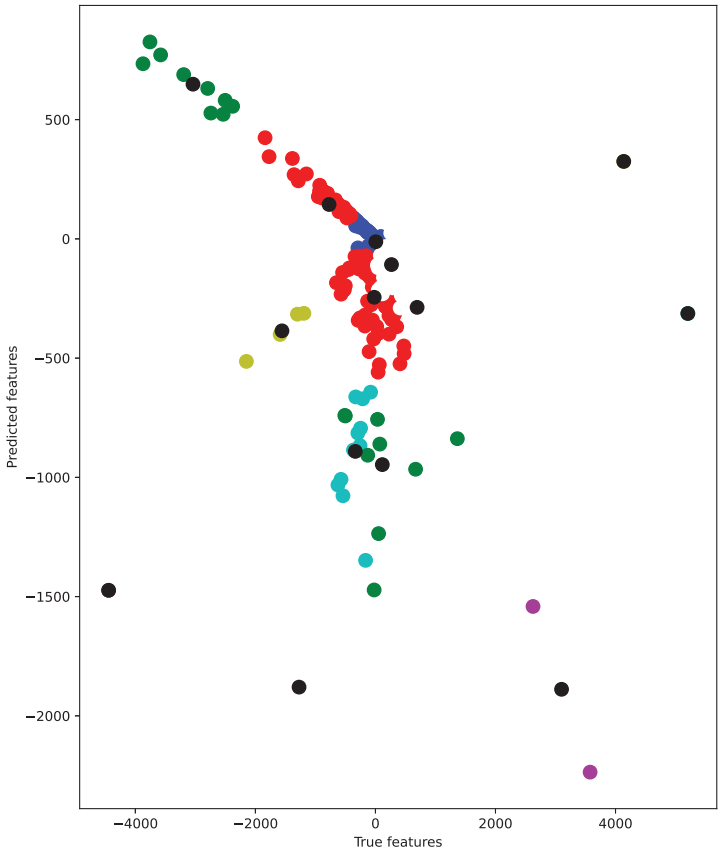

Fig. (10). Showing distribution of contigs among bins for case 4. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

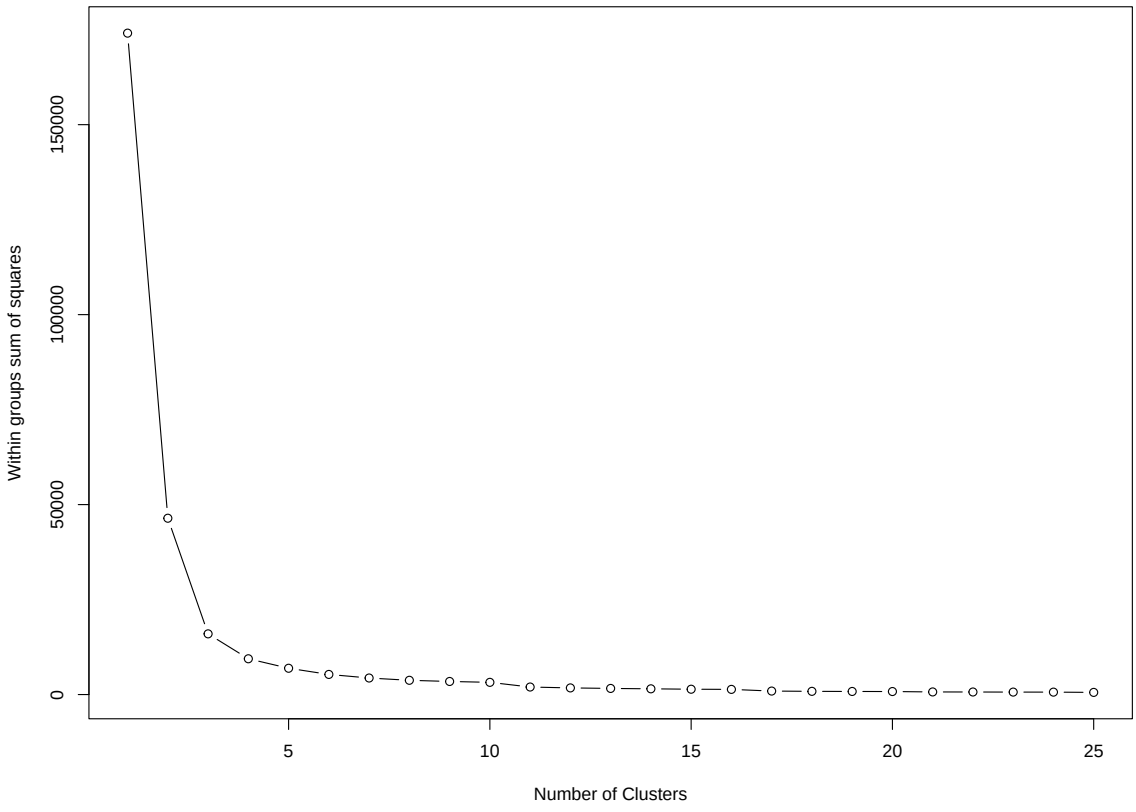

Fig. (11). WSS plot for case 5.

Table 6. Showing species distribution among 6 clusters (HNF+GC) for 25s dataset.

Clusters	Group Name	Abundant Species Name
1	CARCAL, CARFMA	*Candida albicans, Finegoldia magna*
2	CARPAC	*Propionibacterium acnes*
3	CARLCI, CARLSTR	*Propionibacterium acnes, Streptococcus* sp.
4	CARCAL,CARFMA,CARANA	*Candida albicans, Finegoldia magna, Anaerococcus* sp.
5	CARLCI, CARLSTR	*Propionibacterium acnes, Streptococcus* sp.
6	CARCAL,CARFMA,CARANA	*Candida albicans, Finegoldia magna, Anaerococcus* sp.

Table 7. Showing species distribution among 10 clusters (HNF+GC) for Sharon dataset.

Clusters	Group Name	Abundant Species Name
1	CARSTR	*Streptococcus* sp.
2	CARFMA	*Finegoldia magna*
3	CARPAC	*Propionibacterium acnes*
4	CARLCI	*Leuconostoc citreum*
5	CARLCI	*Leuconostoc citreum*
6	CARFMA	*Finegoldia magna*
7	CARPAC	*Propionibacterium acnes*
8	CARCAL	*Candida albicans*
9	CARPAC	*Propionibacterium acnes*
10	CARFMA	*Finegoldia magna*

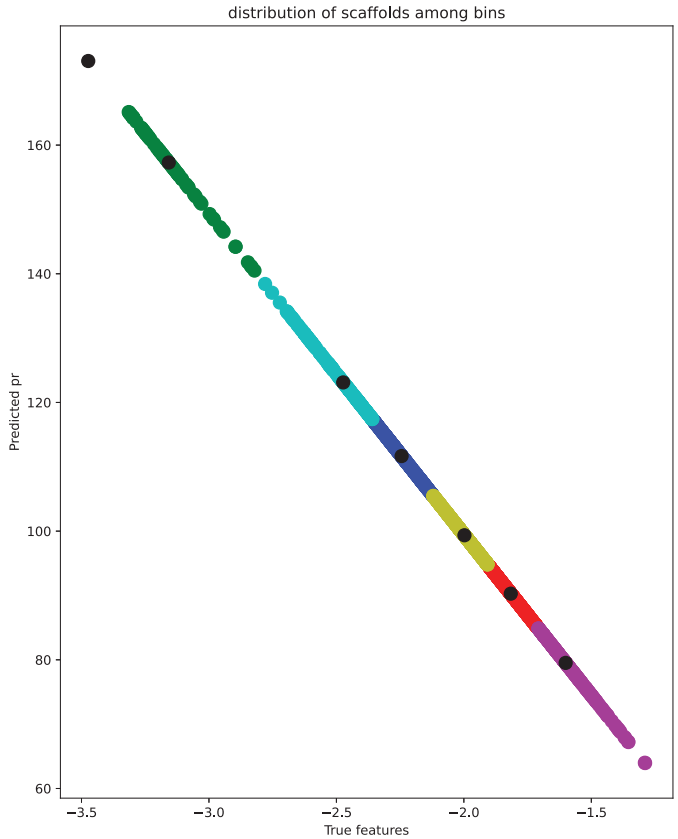

Fig. (12). Showing distribution of contigs among bins for case 5. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

Table 8. Showing the results of the proposed method on various datasets.

Dataset		TNF +GC Features			HNF+GC Features		
		10s	**Sharon**	**25s**	**10s**	**Sharon**	**25s**
Rand Index		0.95	0.88	0.91	0.76	0.68	0.69
Silhouette Index		0.60	0.55	0.53	0.62	0.54	0.68
Optimum Clusters		7	10	6	8	10	14
Accuracy		0.86	0.73	0.77	0.78	0.68	0.72
Precision	Micro	0.86	0.73	0.77	0.78	0.46	0.72
	Macro	0.72	0.66	0.67	0.63	0.46	0.78
	Weighted	0.75	0.56	0.62	0.61	0.50	0.52
Recall	Micro	0.86	0.73	0.77	0.78	0.88	0.72
	Macro	0.88	0.86	0.90	0.88	0.46	0.93
	Weighted	0.86	0.73	0.77	0.78	0.46	0.72
F1- score	Micro	0.86	0.73	0.77	0.78	0.46	0.72
	Macro	0.79	0.73	0.75	0.73	0.61	0.84
	Weighted	0.80	0.63	0.68	0.69	0.31	0.61

Table 9. Showing comparison with existing approaches using TNF+GC features.

Tool	10s Dataset				Sharon Dataset			25s Dataset		
-	**Bins**	**ML Approach**	**Silhouette Index**	**Rand Index**	**Bins**	**Silhouette Index**	**Rand Index**	**Bins**	**Silhouette Index**	**Rand Index**
Proposed	7	unsupervised	0.60	0.95	6	0.55	0.88	10	0.53	0.91
comet	17	unsupervised	-0.24	0.77	16	-0.07	0.78	16	-0.16	0.48
MetaBAT	10	Semi-supervised	-*	0.98	7	-*	0.86	23	-*	0.96
MaxBin	9	Semi-supervised	-*	0.97	4	-*	0.55	24	-*	0.97

Note: *Calculation of these values is not applicable for these tools as they are a semi-supervised tool

Table 10. Showing comparison with existing reference free-based approaches using TNF+GC features.

Dataset/ Combination of Features	10s			Sharon		
	Silhouette Index	**Rand Index**	**Accuracy**	**Silhouette Index**	**Rand Index**	**Accuracy**
Proposed Method	**0.60**	**0.95**	**0.86**	**0.55**	**0.88**	**0.73**
comet	-0.24	0.77	0.82	-0.07	0.78	0.63
MetaConClust	0.49	0.93	0.25	0.26	0.88	0.15

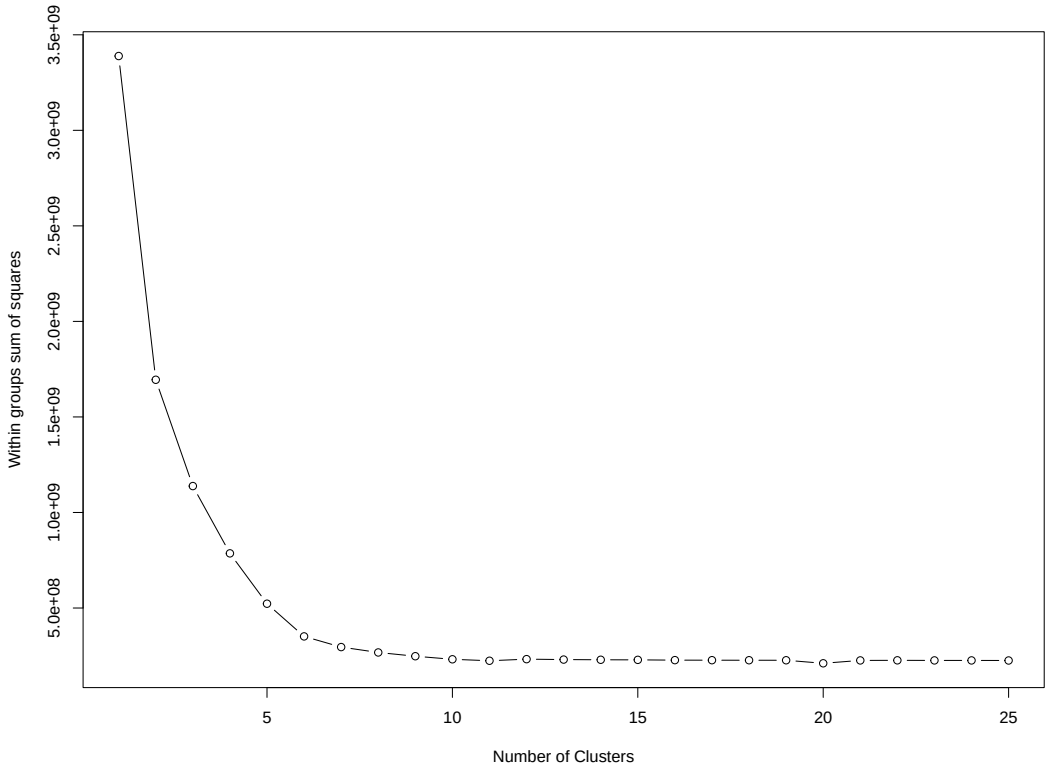

Fig. (13). WSS plot for case 6.

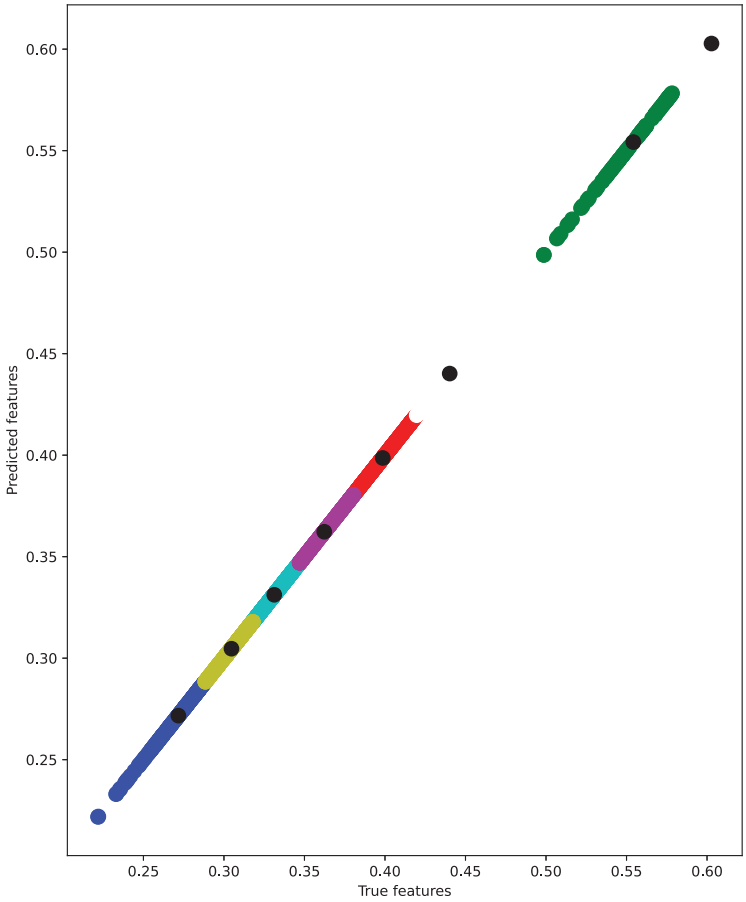

Fig. (14). Showing distribution of contigs among bins for case 6. (*A higher resolution / colour version of this figure is available in the electronic copy of the article*).

0.62, 0.76, and 0.78, respectively. This indicates that our proposed method is performing well. The proposed method was also compared with the existing tools such as a comet, MetaBAT, MaxBin, and MetaConClust using the features TNF+GC. Here, the comparison with other methods is based on TNF+GC only, as all other tools are not based on HNF features. It has been observed from the results that the proposed method is better than the comet approach and MetaConClust, which is an unsupervised method of binning. Compared to MetaBAT and MaxBin, the performance of the proposed method is at par for most of the data. This is because both MetaBat and MaxBin are semi-supervised methods and use some reference data for binning.

CONCLUSION

This study shows a paradigm shift in the application of deep learning for an unsupervised method of binning in metagenomic. It opens a new path to explore the possibility of advanced AI techniques for binning in the absence of reference data, as very few reference genomes are available in the Metagenomics study. In the future, one can extend this work by considering features such as adding some extra genomic features, setting different hyper parameters, and clustering at two stages with different clustering techniques can further be explored to improve the quality of the clusters or bins. One can also try developing a tool that is independent of datasets.

LIST OF ABBREVIATIONS

CA	=	Correspondence Analysis
CAE	=	Convolutional Autoencoder
HNF	=	Hexa-Nucleotide Frequency
MDS	=	Multidimensional Scaling
MSE	=	Mean Square Error
PCA	=	Principle Component Analysis
TNF	=	Tetra-Nucleotide Frequency

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

Not applicable.

HUMAN AND ANIMAL RIGHTS

No animals/humans were used for studies that are the basis of this research.

CONSENT FOR PUBLICATION

Not applicable.

AVAILABILITY OF DATA AND MATERIALS

The data supporting the findings of the article is available at the URL: https://sourceforge.net/projects/sb2nhri/files/MyCC/Data/. Further, all the intermediate results are available with the authors and will be made available on request.

FUNDING

None.

CONFLICT OF INTEREST

The authors declare no conflict of interest, financial or otherwise.

ACKNOWLEDGEMENTS

The authors are very thankful to PG School, ICAR-IARI and ICAR-IASRI for facilitating this work. Thanks are due to all those who helped directly or indirectly in the execution of this work.

REFERENCES

[1] Handelsman, J. Metagenomics: Application of genomics to uncultured microorganisms. *Microbiol. Mol. Biol. Rev.,* **2004**, *68*(4), 669-685. http://dx.doi.org/10.1128/MMBR.68.4.669-685.2004 PMID: 15590779

[2] Meyer, F.; Paarmann, D.; D'Souza, M.; Olson, R.; Glass, E.M.; Kubal, M.; Paczian, T.; Rodriguez, A.; Stevens, R.; Wilke, A.; Wilkening, J.; Edwards, R.A. The metagenomics RAST server – a public resource for the automatic phylogenetic and functional analysis of metagenomes. *BMC Bioinformatics,* **2008**, *9*(1), 386. http://dx.doi.org/10.1186/1471-2105-9-386 PMID: 18803844

[3] Alneberg, J.; Bjarnason, B. S.; de Bruijn, I.; Schirmer, M.; Quick, J.; Ijaz, U. Z.; Quince, C. CONCOCT: Clustering contigs on coverage and composition. arXiv **2013**, *2013*, 1312.4038.

[4] Gelfand, M.S.; Koonin, E.V. Avoidance of palindromic words in bacterial and archaeal genomes: A close connection with restriction enzymes. *Nucleic Acids Res.,* **1997**, *25*(12), 2430-2439. http://dx.doi.org/10.1093/nar/25.12.2430 PMID: 9171096

[5] Teeling, H.; Waldmann, J.; Lombardot, T.; Bauer, M.; Glöckner, F. TETRA: A web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences. *BMC Bioinform.,* **2004**, *5*(1), 163-169. http://dx.doi.org/10.1186/1471-2105-5-163 PMID: 15507136

[6] Abe, T.; Sugawara, H.; Kanaya, S.; Kinouchi, M.; Ikemura, T. Self-Organizing Map (SOM) unveils and visualizes hidden sequence characteristics of a wide range of eukaryote genomes. *Gene,* **2006**, *365*, 27-34. http://dx.doi.org/10.1016/j.gene.2005.09.040 PMID: 16364569

[7] Kislyuk, A.; Bhatnagar, S.; Dushoff, J.; Weitz, J.S. Unsupervised statistical clustering of environmental shotgun sequences. *BMC Bioinform.,* **2009**, *10*(1), 316. http://dx.doi.org/10.1186/1471-2105-10-316 PMID: 19799776

[8] Sharma, A.; Mishra, D.C.; Budhlakoti, N.; Rai, A.; Lal, S.B.; Kumar, S. Algorithmic and computational comparison of metagenome assemblers. *Indian J. Agric. Sci.,* **2020**, *90*, 5.

[9] Chatterji, S.; Yamazaki, I.; Bai, Z.; Eisen, J.A. CompostBin: A DNA composition-based algorithm for binning environmental shotgun reads. *arXiv,* **2007***, 2007*, 0708.3098. http://dx.doi.org/10.1007/978-3-540-78839-3_3

[10] Alcaraz, L.D.; Belda-Ferre, P.; Cabrera-Rubio, R.; Romero, H.; Simón-Soro, A.; Pignatelli, M.; Mira, A. Identifying a healthy oral microbiome through metagenomics. *Clin. Microbiol. Infect.,* **2012**, *18*(Suppl. 4), 54-57. http://dx.doi.org/10.1111/j.1469-0691.2012.03857.x PMID: 22647051

[11] Cox, M.A.; Cox, T.F. Multidimensional scaling. In: *Handbook of data visualisation*; Springer, **2008**; pp. 315-347. http://dx.doi.org/10.1007/978-3-540-33037-0_14

[12] Kusuma, W.A.; Akiyama, Y. Metagenome fragment binning based on characterisation vectors. **2011**, Available from: https://repository.ipb.ac.id/handle/123456789/102901?show=full

[13] Saghir, H.; Megherbi, D.B. An efficient comparative machine learning-based metagenomics binning technique *via* using Random forest. In: *IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA),* 2013 15-17 July; Milan, Italy; pp. 191-196. http://dx.doi.org/10.1109/CIVEMSA.2013.6617419

[14] Fiannaca, A.; La Paglia, L.; La Rosa, M.; Lo Bosco, G.; Renda, G.; Rizzo, R.; Gaglio, S.; Urso, A. Deep learning models for bac-

teria taxonomic classification of metagenomic data. *BMC Bioinform.,* **2018**, *19*(Suppl. 7), 198.
http://dx.doi.org/10.1186/s12859-018-2182-6 PMID: 30066629

[15] Guo, X.; Liu, X.; Zhu, E.; Yin, J. Deep Clustering with Convolutional Autoencoders. In: Liu, D.; Xie, S.; Li, Y.; Zhao, D.; El-Alfy, ES. Eds.; In: *Neural Information Processing*. Lecture Notes in Computer Science, 2017, vol. 10635, Springer, Cham, pp. 373-382.

[16] Temperton, B.; Giovannoni, S.J. Metagenomics: Microbial diversity through a scratched lens. *Curr. Opin. Microbiol.,* **2012**, *15*(5), 605-612.
http://dx.doi.org/10.1016/j.mib.2012.07.001 PMID: 22831844

[17] Sharon, I.; Morowitz, M.J.; Thomas, B.C. Time series community genomics analysis reveals rapid shifts. *Genome Res*., **2013**, *23*(1), 111-120.

[18] Herath, D.; Tang, S.L.; Tandon, K.; Ackland, D.; Halgamuge, S.K. CoMet: A workflow using contig coverage and composition for binning a metagenomic sample with high precision. *BMC Bioinform.,* **2017**, *18*(Suppl. 16), 571.
http://dx.doi.org/10.1186/s12859-017-1967-3 PMID: 29297295

[19] Sinha, D.; Sharma, A.; Mishra, D.C.; Rai, A.; Lal, S.B.; Kumar, S.; Farooqi, M.S.; Chaturvedi, K.K. MetaConClust - Unsupervised Binning of Metagenomics Data using Consensus Clustering. *Curr. Genomics,* **2022**, *23*(2), 137-146.
http://dx.doi.org/10.2174/1389202923666220413114659

[20] Richard, G.; Grossin, B.; Germaine, G.; Hébrail, G.; de Moliner, A. Autoencoder-based time series clustering with energy applications. *arXiv,* **2020,** *2020,* 2002.03624.

[21] Gulli, A.; Pal, S. *Deep learning with Keras*; Packt Publishing Ltd.: Birmingham, UK, **2017**.

[22] Hahsler, M.; Piekenbrock, M.; Doran, D. dbscan: Fast density-based clustering with R. *J. Stat. Softw.,* **2019**, *91*(1), 1-30.
http://dx.doi.org/10.18637/jss.v091.i01

[23] Aggarwal, D.; Sharma, D. Application of clustering for student result analysis. *Int. J. Recent Technol. Eng.,* **2019**, *7*(6), 50-53.

[24] Serra, A.; Tagliaferri, R. Unsupervised learning: Clustering. **2019**, Available form: https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a.

[25] van der Walt, S.; Colbert, S.C.; Varoquaux, G. The NumPy array: A structure for efficient numerical computation. *Comput. Sci. Eng.,* **2011**, *13*(2), 22-30.
http://dx.doi.org/10.1109/MCSE.2011.37

[26] McKinney, W. Pandas: A foundational Python library for data analysis and statistics. *Seman. Scholor,* **2011**, *14*(9), 61539023.

[27] Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Duchesnay, E. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.,* **2011**, *12*, 2825-2830.

[28] Ari, N.; Ustazhanov, M. Matplotlib in Python. *11th International Conference on Electronics, Computer and Computation (ICECCO),* **2014**, *2014*, 6997585.
http://dx.doi.org/10.1109/ICECCO.2014.6997585

[29] Chen, W.C. *A Quick Guide for the phyclust Package*; Iowa State University: Ames, IA, USA, **2010**.

[30] Kuhn, M.; Wing, J.; Weston, S.; Williams, A.; Keefer, C.; Engelhardt, A.; Team, R.C. Package 'caret'. *RJ.,* **2020**, *223*, 7.

[31] Robin, X.; Turck, N.; Hainard, A.; Tiberti, N.; Lisacek, F.; Sanchez, J.C.; Müller, M. pROC: An open-source package for R and S+ to analyze and compare ROC curves. *BMC Bioinformatics,* **2011**, *12*(1), 77.
http://dx.doi.org/10.1186/1471-2105-12-77 PMID: 21414208

[32] Wu, Y.W.; Tang, Y.H.; Tringe, S.G.; Simmons, B.A.; Singer, S.W. MaxBin: An automated binning method to recover individual genomes from metagenomes using an expectation-maximization algorithm. *Microbiome,* **2014**, *2*(1), 26.
http://dx.doi.org/10.1186/2049-2618-2-26 PMID: 24468033

[33] Kang, D.D.; Froula, J.; Egan, R.; Wang, Z. MetaBAT, an efficient tool for accurately reconstructing single genomes from complex microbial communities. *PeerJ,* **2015**, *3*, e1165.
http://dx.doi.org/10.7717/peerj.1165 PMID: 26336640

[34] Robin, X.; Turck, N.; Hainard, A.; Tiberti, N.; Lisacek, F.; Sanchez, J. C.; Müller, M. pROC: an open-source package for R and S to analyze and compare ROC curves. *BMC bioinformatics*., **2011**, *12*(1), 1-8.