Statistics and Applications {ISSN 2454-7395 (online)}
Volume 23, No. 2, 2025 (New Series), pp 237–266
http://www.ssca.org.in/journal

Reducing Dimensionality and Modeling Structural Dependence in Data Streams: An Approach using Copula Matrices

Fereshteh Arad[1], Ayyub Sheikhi[1] and Farshid Keynia[2]

[1]*Department of Statistics, Faculty of Mathematics and Computer, Shahid Bahonar University of Kerman, Kerman, Iran*
[2]*Department of Energy Management and Optimization, Institute of Science and High Technology and Environmental Sciences, Graduate University of Advanced Technology, Kerman, Iran*

Received: 01 September 2024; Revised: 27 March 2025; Accepted: 31 March 2025

Abstract

Dimension reduction techniques are effective in dealing with high-volume data. Considering streaming data, in this study we introduce a dimension reduction algorithm when non-linear relationships exist between variables. We propose a copula-based feature extraction algorithm to extract streaming most informative components. Using a simulation study we show that our proposed algorithm has a better performance compared to other traditional feature extraction methods. We also use the extracted components to make a prediction analysis. A real-time analysis of online Apple stock data has been provided to illustrate the merits of our approach. Moreover, we carry out feature extraction as well as a prediction analysis in the *COVID*-19 and Real-time Bitcoin dataset and show that our method benefits the prediction accuracy.

Key words: Data stream; Copula; Dimension reduction; Feature extraction.

AMS Subject Classifications: 68T05, 68W27.

1. Introduction

Recent advancements in information technology have led to the generation of a large volume of data that requires online analysis and intelligent processing to extract useful information. One key concept that holds significant importance in this field is the data stream, which is an endless collection of countable elements that enter the system continuously and are analyzed by various processes to generate new results (*c.f.*, Margara and Rabl (2019)).

The data stream has created various research challenges, especially yielding big data, so, reducing the dimension of the data has received more attention. Generally, dimension reduction refers to the process of reducing the number of random variables under considera-

Corresponding Author: Ayyub Sheikhi
Email: sheikhy.a@uk.ac.ir

tion by obtaining a set of principal variables, and is commonly used in fields such as pattern recognition, data mining, and machine learning. As some instances, we refer to van der Maaten *et al.* (2009) and Sarveshwaran *et al.* (2019), Jia *et al.* (2022), Salem and Hussein (2019).

In the context of the data stream, researchers have used dimension reduction methods to simplify the analysis of complex streaming data sets, improve the performance of data mining algorithms, and reduce storage and computational requirements. Yan *et al.* (2006) have proposed an effective dimensionality reduction for a larger scale and streaming data; Upadhyay *et al.* (2013) have investigated dimension reduction techniques in data stream mining; Xie *et al.* (2020) have introduced an efficient dimension reduction for streaming multi-view data and Zheng *et al.* (2022) have used graph diffusion for a streaming feature selection. In this work we have we propose a feature extraction algorithm as a subcategory of dimension reduction in the data stream. Bahri *et al.* (2020) have studied feature transformation techniques for data streams. Park and Lee (2020) have studied a linear dimension reduction method for streaming data. Moreover, since time series are the special case of the data stream models, dimension reduction techniques have been carried out by the researchers. For further study in this area, we refer to Wang and Megalooikonomou (2008) and Park *et al.* (2010), Krawczak and Szkatuła (2014).

As another challenge of the data stream, the relationship between variables varies over time by increasing the amount of data. The concept of dependence between variables can be explained using copula functions which can address the linear and nonlinear dependencies of the variables (*c.f.*, Sheikhi *et al.* (2022a)). Houari *et al.* (2016) have introduced a new technique for reducing the dimensionality of multi-dimensional data using Copulas. Qu (2012) have refereed to the feature extraction using the copula function. Simard and Remillard (2013) utilized the copula function to model time series model also presented a prediction method for the time series using a copula function.

Also, nowadays, GPU graphics processors are used to perform complex calculations in fields such as machine learning, scientific simulations, and image and video processing. These processing units are very efficient due to the ability to perform a large number of operations simultaneously and help improve graphics quality and processing speed (*c.f.*, Misic *et al.* (2012), Ram (2023)).

In this article, we propose our Stream Copula Feature Extraction (SCFE) algorithm to reduce the dimensionality of data in data streams by extracting the main principal components. Then, we apply the extracted components as new informative inputs for prediction purposes several studies have been done in this context. Sheikhi *et al.* (2022b) have been working on the topic of dimension reduction in neural networks using copula function, Zeng and Wang (2022) have introduced a dimension reduction approach called Neural Copula. Carrillo *et al.* (2021) have presented a machine learning prediction algorithm based on a copula function. Toharudin *et al.* (2019) have used a combination of copula and neural networks for forecasting. Qu (2012) in their article focuses on feature extraction using Archimedes copula. However, those methods were inefficient in dealing with streaming data. The structure of this work is as follows. Section 2 introduces a couple-based feature extraction algorithm and uses the extracted components to carry out a prediction analysis. Section 3 is devoted to providing numerical analysis of the proposed algorithm by simulated and real

datasets, and finally, Section 4 brings some concluding remarks.

2. Stream copula feature extraction

One of the most important methods in dimensionality reduction is feature extraction. Feature extraction leads to dimensionality reduction by combining the main features. From this perspective, constructing a new set of components is usually more compact and has a greater distinguishing property (*c.f.*, Mutlag *et al.* (2020)). The main question here is what to do if the variables are not necessarily related linearly? This question will become an important issue when these non-linear connections vary over time.

Feature extraction methods may consider non-linear relationships but do not consider the dependency between variables (*c.f.*, Fukunaga and Short (1978), Zhu (2010)), so there is a need for a method to consider the dependency between variables. On the other hand, in the data stream, variables may show different behaviors at each moment, and exploring these behaviors will help researchers extract the most informative components. In this study, we attempt to copula function to study the dependency between variables in streaming data and we present a method for feature extraction.

Consider a continuous random vector $(X_1, X_2, ..., X_n)$. Let F_j be the marginal cumulative distribution function (*CDF*) of X_j for $j = 1, 2, ..., n$, and F be the joint *CDF*. We apply the probability integral transform and define $U_j := F_j(X_j),\ j = 1, 2, ..., n$. Since X_j is assumed to be continuous, $U_j \sim U(0, 1)$ follows a uniform distribution. Then the CDF of $(U_1, U_2, ..., U_n)$ is the copula of $(X_1, X_2, ..., X_n)$ (*c.f.*, Nelsen (2006)). Denoting this copula function as C, we have

$$C(u_1, u_2, ..., u_n) = F(F_1(x_1), F_2(x_2), ..., F_n(x_n))$$

Assume that at the time t, the vectors $\boldsymbol{x}_1^{(t)}, \boldsymbol{x}_2^{(t)}, \boldsymbol{x}_3^{(t)}, ..., \boldsymbol{x}_n^{(t)}$, are pairwise copula related random vectors. In this case, the Kendall's association is as $\boldsymbol{\tau}^{(t)} = (\tau_{i,j})^{(t)},\quad i, j = 1, 2, ..., n,\ t = 1, 2, ...,$ where

$$(\tau_{i,j})^{(t)} = \tau_{X_i^{(t)}, X_j^{(t)}} = 4 \int_0^1 \int_0^1 C_{X_i^{(t)}, X_j^{(t)}}(u_i^{(t)}, u_j^{(t)}) dC_{X_i^{(t)}, X_j^{(t)}}(u_i^{(t)}, u_j^{(t)}) - 1, \quad i, j = 1, 2, ..., n.$$

It is known that the copula association measures such as Spearman's ρ and Kendall's τ can better capture the non-linear dependencies in contrary to Pearson's correlation (*c.f.*, Zeng and Wang (2022)). As a motivated example for implementing the time, consider z consist of $t = 25$ observation from a normal distribution and z_1, z_2, z_3, z_f and z_e are respectively, linear, quadratic, cubic, cumulative normal, and an exponential transform of z. Figure 4-a depicts the differences between the values of pairwise Pearson's correlation and the values of pairwise Kendall's association of variables z, z_1, z_2, z_3, z_f and z_e for the first five time points; Figure 3-b shows this difference for the first 10 time points, and so on, Figure 3-f compares these two measures for all 25 time points.

As Figure 3 reveals, due to the small number of observations, *i.e.*, $t = 5$, there may be a linear relationship between the variables, but with the passage of time and the increase of observations, *e.g.*, $t = 25$, the intensity of a linear relationship decreases while the Kendall's

Table 1: Pairwise Pearson's and Kendall measures at time 1 (first five observations), time 2 (first ten observations),..., time 5 (all twenty five observations)

time	Pearson's correlation matrix					Kendall's association matrix				
	1	0.867	0.870	0.934	0.910	1	0.800	1	1	1
	0.867	1	0.995	0.636	0.993	0.800	1	0.800	0.800	0.800
time1	0.870	0.995	1	0.638	0.995	1	0.800	1	1	1
	0.934	0.636	0.638	1	0.704	1	0.800	1	1	1
	0.910	0.993	0.995	0.704	1	1	0.800	1	1	1
	1	0.731	0.819	0.950	0.864	1	0.280	1	1	1
	0.731	1	0.968	0.498	0.968	0.280	1	0.280	0.280	0.280
time 2	0.819	0.968	1	0.601	0.993	1	0.280	1	1	1
	0.950	0.498	0.601	1	0.668	1	0.280	1	1	1
	0.864	0.968	0.993	0.668	1	1	0.280	1	1	1
	1	0.523	0.782	0.959	0.807	1	0.270	1	1	1
	0.523	1	0.871	0.305	0.907	0.270	1	0.270	0.270	0.270
time 3	0.782	0.871	1	0.577	0.987	1	0.270	1	1	1
	0.959	0.305	0.577	1	0.619	1	0.270	1	1	1
	0.807	0.907	0.987	0.619	1	1	0.270	1	1	1
	1	0.519	0.770	0.961	0.805	1	0.400	1	1	1
	0.519	1	0.865	0.310	0.903	0.400	1	0.400	0.400	0.400
time 4	0.770	0.865	1	0.566	0.984	1	0.400	1	1	1
	0.961	0.310	0.566	1	0.623	1	0.400	1	1	1
	0.805	0.903	0.984	0.623	1	1	0.400	1	1	1
	1	0.400	0.752	0.967	0.783	1	0.013	1	1	1
	0.400	1	0.810	0.206	0.859	0.013	1	0.013	0.013	0.013
time 5	0.752	0.810	1	0.562	0.982	1	0.013	1	1	1
	0.967	0.206	0.562	1	0.704	1	0.013	1	1	1
	0.783	0.859	0.982	0.613	1	1	0.013	1	1	1

association measure remains stable. Moreover, see Table 1 for the comparison of these two pairwise measures at five time points.

The main aim of this section is carrying out a competitive feature extraction by extracting more informative components of features when the features do not have necessarily linear relationships. In this regard, inspired by (*c.f.*, Witten *et al.* (2009)), we obtain the ordered copula-based principal components by solving the following equation.

$$(\boldsymbol{\tau}^{(t)} - \hat{\lambda}^{(t)}\boldsymbol{I}_n)\hat{\boldsymbol{g}}^{(t)} = \mathbf{0}, \quad t = 1, 2, ..., T, \tag{1}$$

where $\boldsymbol{I}_n$ is an identity matrix of dimension n. Denote $\hat{\lambda}_1^{(t)}, \hat{\lambda}_2^{(t)}, \cdots, \hat{\lambda}_n^{(t)}$ are descending ordered eigenvalues of the matrix $\boldsymbol{\tau}^{(t)}$ at time t and their corresponding eigenvectors are $\hat{\boldsymbol{g}}_1^{(t)}, \hat{\boldsymbol{g}}_2^{(t)}, \cdots, \hat{\boldsymbol{g}}_n^{(t)}$.

Let define the matrix of all observations at time t as $\boldsymbol{X}^{(t)} = (\boldsymbol{x}_1^{(t)}, \boldsymbol{x}_2^{(t)}, \boldsymbol{x}_3^{(t)}, ..., \boldsymbol{x}_n^{(t)})$ and matrix of all eigenvectors as $\hat{\boldsymbol{G}}^{(t)} = (\hat{\boldsymbol{g}}_1^{(t)}, \hat{\boldsymbol{g}}_2^{(t)}, \cdots, \hat{\boldsymbol{g}}_n^{(t)})$ then by multiplying these two matrices we obtain the new set of observation as $\boldsymbol{Q}^{(t)} = \boldsymbol{X}^{(t)}\hat{\boldsymbol{G}}^{(t)}$. In the sequel, regarding to extracting more informative principal component, we use the matrix $\boldsymbol{Q}^{(t)}$. The first column

of this matrix includes the a larger portion of the total variance, the second column stands in the second rank, and so on. If we also want to calculate the cumulative explained variance of these columns, we can evaluate the explained variance of the l-th columns as

$$P_l^{(t)} = \frac{\hat{\lambda}_l^{(t)}}{\sum_{j=1}^{n} \hat{\lambda}_j^{(t)}},\ l = 1, 2, ..., n. \tag{2}$$

Example 1: Suppose there are three variables at three times $X_1^{(t)}$, $X_2^{(t)}$ and $X_3^{(t)}$, $t = 1, 2, 3$ in which the relationship between these variables vary over time. Suppose at time 1, variables $X_1^{(1)}$ and $X_2^{(1)}$ are related through the Clayton copula (*c.f.*, Cuvelier and Noirhomme-Fraiture (2005)) with $\theta^1 = 0.5$, and variables $X_1^{(1)}$ and $X_3^{(1)}$ are related through the Gumbel copula (*c.f.*, Wang *et al.* (2010)) with $\eta^1 = 0.8$, and variables $X_2^{(1)}$ and $X_3^{(1)}$ are related through the Gaussian copula with $\rho^1 = 0.3$. Using the equation (1), we obtain $\boldsymbol{\tau}^{(1)}$ as the first row of Table 2 and hence using 2 the cumulative explained variance as $P_1^{(1)} = 0.41$, $P_1^{(1)}+P_2^{(1)} = 0.80$ and $\sum_{l=1}^{3} P_l^{(1)} = 1$. Similarly, see the second and the third row of this table for different assumptions and results at times 2 and 3. As Table 2 reveals, as time moves forward, the association measure shows stronger relationship. For example, the sum of the explained variances of the first two extracted components at time 1 is 0.81, while for time 2 and 3 will be 0.91 and 0.93 respectively.

Table 2: Pairwise Pearson's and Kendall measures at time 1 (first five observations), time 2 (first ten observations),..., time 5 (all twenty observations)

Time	Assumptions	$\boldsymbol{\tau}^{(t)}$	$\sum_{l=1}^{k} P_l^{(t)}$
1	$X_1^{(1)}, X_2^{(1)} \sim Cl(\theta^{(1)} = 0.5)$ $X_1^{(1)}, X_3^{(1)} \sim G(\eta^{(1)} = 0.8)$ $X_2^{(1)}, X_3^{(1)} \sim Ga(\rho^{(1)} = 0.3)$	$\boldsymbol{\tau}^{(1)} = \begin{bmatrix} 1 & 0.20 & -0.25 \\ 0.20 & 1 & 0.19 \\ -0.25 & 0.19 & 1 \end{bmatrix}$	$P_1^{(1)} = 0.41$ $P_1^{(1)} + P_2^{(1)} = 0.80$ $\sum_{l=1}^{3} P_l^{(1)} = 1$
2	$X_1^{(2)}, X_2^{(2)} \sim Cl(\theta^{(2)} = 0.6)$ $X_1^{(2)}, X_3^{(2)} \sim G(\eta^{(2)} = 0.7)$ $X_2^{(2)}, X_3^{(2)} \sim C^t(\zeta_1^{(2)} = 0.7)$	$\boldsymbol{\tau}^{(2)} = \begin{bmatrix} 1 & 0.23 & -0.42 \\ 0.23 & 1 & 0.49 \\ -0.42 & 0.49 & 1 \end{bmatrix}$	$P_1^{(2)} = 0.51$ $P_1^{(2)} + P_2^{(2)} = 0.91$ $\sum_{l=1}^{3} P_l^{(2)} = 1$
3	$X_3^{(3)}, X_2^{(3)} \sim Cl(\theta^{(3)} = 0.6)$ $X_1^{(3)}, X_3^{(3)} \sim Ga(\rho_1^{(3)} = 0.8)$ $X_2^{(3)}, X_3^{(3)} \sim Ga(\rho_2^{(3)} = 0.9)$	$\boldsymbol{\tau}^{(3)} = \begin{bmatrix} 1 & 0.23 & 0.59 \\ 0.23 & 1 & 0.71 \\ 0.59 & 0.71 & 1 \end{bmatrix}$	$P_1^{(3)} = 0.68$ $P_1^{(3)} + P_2^{(3)} = 0.93$ $\sum_{l=1}^{3} P_l^{(3)} = 1$

The following theorem states that we reach faster to the higher levels of percentage of the variance's explanation by passage of the time.

Theorem 1: Let $\lambda_1^{(t)} > \lambda_2^{(t)} > \cdots, > \lambda_n^{(t)}$ be the eigenvalues of Kendall's association matrix of time $t = 1, 2, 3...$ and define $f(c)^{(t)} = 1 - \sum_{j=1}^{k} P_j^{(t+c)}$ and $f^{(t)} = 1 - \sum_{j=1}^{k} P_j^{(t)}$ then $f(c)^{(t)} = o(f^{(t)})$.

Proof: According to the definition of 2, $\exists\ k_0$ s.t $k_0 = \underset{u \in N}{argmax} \sum_{j=1}^{u} P_j^{(t)}$ and $\forall k > k_0$ we have $0 < \sum_{j=1}^{k} P_j^{(t)} < \sum_{j=1}^{k} P_j^{(t+c)} \leq 1$ and $\lim_{t\to\infty} \sum_{j=1}^{k} P_j^{(t+c)} = 1$, therefore, it can be said that $0 < \lim_{t\to\infty} 1 - \sum_{j=1}^{k} P_j^{(t)} < 1$. Also, as $\lim_{t\to\infty} 1 - \sum_{j=1}^{k} P_j^{(t)} \neq 0$ the limit ratio can be considered equal to the ratios and $\lim_{t\to\infty} 1 - \sum_{j=1}^{k} P_j^{(t+c)} = 0$ so we have

$$\lim_{t\to\infty} \frac{f(c)^{(t)}}{f^{(t)}} = \lim_{t\to\infty} \frac{1 - \sum_{j=1}^{k} P_j^{(t+c)}}{1 - \sum_{j=1}^{k} P_j^{(t)}} = 0$$

□

One result of the above theorem is the following corollary.

Corollary 1: When faced with a data stream as time moves forward, $\lim_{t\to\infty} \sum_{j=1}^{k} P_j^{(t+c)}$ for $k_0 < k \leq n$ In such a way $k_0 = \underset{u \in N}{argmax} \sum_{j=1}^{u} P_j^{(t)}$ and $c = 1, 2, 3, ...$ reaches one earlier than previous times $\lim_{t\to\infty} \sum_{j=1}^{k} P_j^{(t)}$.

Proof: Considering theorem 1 and the relationship $f(c)^{(t)} = o(f^{(t)})$, since $\lim_{t\to\infty} 1 - \sum_{j=1}^{k} P_j^{(t+c)}$ approaches zero faster than $\lim_{t\to\infty} 1 - \sum_{j=1}^{k} P_j^{(t)}$, it can be said that $\lim_{t\to\infty} \sum_{j=1}^{k} P_j^{(t+c)}$ converges faster than $\lim_{t\to\infty} \sum_{j=1}^{k} P_j^{(t)}$ to one. □

The above corollary indicates that over time, as the behaviors of variables become clear, we can extract features more accurately using Kendall's correlation matrix between variables. This approach can explain a higher percentage of variance with fewer components. After extracting the relevant components based on the explained variance percentage, they can be used instead of the original features in the inputs of a prediction model such as neural networks, regression, *etc.*

Regarding to a prediction task, we can use the primary components obtained at each time as inputs of a neural network and update the weight matrix of the artificial neural network to minimize the prediction error below a certain threshold δ. This process involves obtaining the first hidden layer for each time t as follows

$$\boldsymbol{h}_1^{(t)} = f_w^{(t)}(\boldsymbol{w}_{hu}^{(t)} \boldsymbol{u}_d^{(t)} + \boldsymbol{b}_1^{(t)}) \qquad t = 1, 2, ..., \tag{3}$$

where $\boldsymbol{w}_{hu}^{(t)}$ stands for the weights between the input values and the nodes of the first hidden layer at time t, $f_w^{(t)}$ is an activation function and $\boldsymbol{b}_1^{(t)}$ is the bias term at time t. Also, in the next layers

$$\boldsymbol{h}_k^{(t)} = f_w^{(t)}(\boldsymbol{w}_{k,k-1}^{(t)}\boldsymbol{h}_{k-1}^{(t)} + \boldsymbol{b}_k^{(t)}), \quad t = 1, 2, ..., \quad k = 2, 3, ..., \tag{4}$$

in which, $f_w^{(t)}$ represents an activation function at time t, $\boldsymbol{w}_{k,k-1}^{(t)}, k = 2, 3, ...$, are the connecting weights between the k-th and $(k-1)$-th layers at time t, and $\boldsymbol{b}_k^{(t)}$ is the bias term at time t. The subsequent equation in the neural network utilizes the final layer and certain weights to generate the output $\hat{\boldsymbol{y}}^{(t)}$ as

$$\hat{\boldsymbol{y}}^{(t)} = g_w^{(t)}(\boldsymbol{w}_{hy}^{(t)}\boldsymbol{h}_d^{(t)} + \boldsymbol{b}_y^{(t)}) \qquad t = 1, 2, ..., \tag{5}$$

where $g_w^{(t)}$ can be a nonlinear function and $\boldsymbol{w}_{hy}^{(t)}$ is the weight associated with the hidden layers and outputs at time t. Additionally, $\boldsymbol{b}_y^{(t)}$ represents the bias term at time t.

Algorithm 1: Stream Copula Feature Extraction Neural Network (*SCFE-NN*) algorithms for all times.

Data: streaming data $X^{(t)}$ and $\boldsymbol{y}^{(t)}$, minimum percentage of explained variance α, and threshold value $\delta^{(t)}$

Result: Updating the weights of the neural network for the extracted components

Initialization; Obtain the copula matrix $\boldsymbol{\tau}^{(t)}$ and compute eigenvectors $\hat{\boldsymbol{g}}_1^{(t)}, ..., \hat{\boldsymbol{g}}_n^{(t)}$

***SCFE*:**

set $l = 1$ and obtain $\boldsymbol{Q}^{(t)} = \boldsymbol{X}^{(t)}\hat{\boldsymbol{g}}_1^{(t)}$ and it's $P_l^{(t)}$ using 2

while $\sum_{k=1}^{l} P_k^{(t)} < \alpha$ **do**

  $l = l + 1$

  construct $\boldsymbol{Q}^{(t)} = \boldsymbol{X}^{(t)}(\hat{\boldsymbol{g}}_1^{(t)}, ..., \hat{\boldsymbol{g}}_l^{(t)})$ and obtain it's $P_l^{(t)}$

***SCFE-NN*:**

Set the j-th input of NN as $\boldsymbol{u}_j^{(t)} = \boldsymbol{X}^{(t)}\hat{\boldsymbol{g}_j}^{(t)}$ and and construct $\boldsymbol{U}^{(t)} = (\boldsymbol{u}_1^{(t)}, \boldsymbol{u}_2^{(t)}, \boldsymbol{u}_k^{(t)})$

while $||\hat{\boldsymbol{y}}^{(t)} - \boldsymbol{y}^{(t)}||_2 < \delta^{(t)}$ **do**

  Update the NN's weights and biases using 3-5.

Based on the results of theorem 1, by passage of time, the values of $\sum_{k=1}^{l} P_k^{(t)}$ will be increased, so we may perform the algorithm 1 for initial values of time and increase t until $\sum_{k=1}^{l} P_k^{(t)}$ meets the threshold of minimum explained variance (α).

3. Numerical study

In this Section we assess the performance of our approach using a simulation and we apply our contribution to analyze two real world data sets. Two main approaches have been used to optimize the model. First, to optimize the parameters of the copula function, the `Bicopselect` command is used in the copula library in `R` software, this command uses the *MLE* method, this method finds the best fit for the data and allows us to accurately model the dependence between the variables. In the next step, a neural network is used for more complex modeling, and a piece of code written in `R` software is used to optimize the network parameters such as units, stepmax, number-layers using mean absolute error (*MAE*) criterion. To evaluate and ensure the generalizability of the models, the 5-fold cross-validation method has been used. The codes of the article have been implemented in Google colab's gpu[1] space, the output of the results is given below.

3.1. Simulation study

In order to perform a simulation analysis, we consider Gaussian, t and Gumbel copulas for generating data set. Assume that $X_1, X_2, .., X_5 \sim G_5(\boldsymbol{R}_1)$ and $X_6, ..., X_{10} \sim t_5(\boldsymbol{R}_2, \nu)$, where G_5 and t_5 are respectively five dimensional Gaussian and t copulas where

$$\boldsymbol{R}_1 = \begin{bmatrix} 1 & 0.5 & 0.6 & 0.65 & 0.8 \\ 0.4 & 1 & 0.5 & 0.4 & 0.5 \\ 0.6 & 0.5 & 1 & 0.5 & 0.6 \\ 0.65 & 0.4 & 0.5 & 1 & 0.6 \\ 0.8 & 0.5 & 0.6 & 0.6 & 1 \end{bmatrix}, \boldsymbol{R}_2 = \begin{bmatrix} 1 & 0.3 & 0.4 & 0.65 & 0.6 \\ 0.3 & 1 & 0.3 & 0.5 & 0.4 \\ 0.4 & 0.3 & 1 & 0.4 & 0.5 \\ 0.65 & 0.5 & 0.4 & 1 & 0.6 \\ 0.6 & 0.4 & 0.5 & 0.6 & 1 \end{bmatrix}$$

are respectively their correlation matrices. Also, we assume that $X_{11}, X_{12}, .., X_{15}$ follow Gumbel's copula with $\theta = 15$. We generate 1000 random samples from these 15 variables considering their marginals come from a normal distribution.

In the interest of creating the target variable, y, assuming that ε is a white noise, we employ the equation $Y = \sum_{i=1}^{15} X_i + e^{\sum_{i=1}^{5} X_i} + \varepsilon$ which provides linear and nonlinear relations between inputs and output variables. Finally, suppose we have 5 time point, so, we divide the observation into 5 overlapping segments., *i.e.*, Segment 1: the first 200 cases, Segment 2: the first 400 cases,..., Segment 5: all 1000 cases.

By estimating the pairwise connection between the variables in these five segments ($t = 1, 2, .., 5$), and calculating the Kendall copula correlation matrix ($\hat{\boldsymbol{\Gamma}}^{(t)}$) and the Pearson correlation matrix ($\hat{\boldsymbol{R}}^{(t)}$), we can reduce the dimension of the variables via implementing algorithm 1. The results are summarized in Figure 4. As it can be seen from Figure 4, the Kendall's association matrix easily resolves non linear relationship between variables over time.

To assess our *SCFE-NN* approach, we compare its results with the results of the other traditional feature extraction methods. The cumulative percentage of explained variance by our approach as well as other approaches such as Linear *PCA*, polynomial Kernel *PCA*, Gaussian Kernel *PCA*, Laplacian Kernel *PCA*, and *ANOVA* Kernels *PCA* are provided in

[1]https://colab.research.google.com/drive/1xHgYWQxfIktNx6YU4CXEQSzaDoriYr8I?usp=sharing

Table 3: The cumulative percentage of explained variance by six *PCA* methods at different times

time 1	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10	comp 11	comp 12	comp 13	comp 14	comp 15
polynomial Kernel *PCA*	0.65	0.94	**0.96**	0.97	0.98	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1
Gaussian Kernel *PCA*	0.24	0.41	0.52	0.60	0.67	0.72	0.76	0.81	0.84	0.87	0.90	0.93	**0.95**	0.97	1
Laplacian Kernel *PCA*	0.28	0.43	0.52	0.60	0.66	0.71	0.76	0.80	0.84	0.88	0.90	0.93	**0.95**	0.98	1
ANOVA Kernels *PCA*	0.27	0.44	0.54	0.61	0.68	0.73	0.77	0.81	0.84	0.87	0.90	0.93	**0.95**	0.97	1
Copula *PCA*	0.50	0.64	0.75	0.86	0.91	0.93	**0.96**	0.96	0.97	0.98	0.98	0.99	0.99	0.99	1
Linear *PCA*	0.60	0.73	0.83	0.88	0.92	**0.96**	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
time 2	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10	comp 11	comp 12	comp 13	comp 14	comp 15
polynomial Kernel *PCA*	0.45	0.72	0.87	0.94	**0.97**	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
Gaussian Kernel *PCA*	0.26	0.44	0.55	0.63	0.70	0.75	0.80	0.83	0.87	0.90	0.93	**0.95**	0.96	0.98	1
Laplacian Kernel *PCA*	0.29	0.45	0.55	0.63	0.69	0.75	0.79	0.82	0.86	0.88	0.91	0.94	**0.96**	0.98	1
ANOVA Kernels *PCA*	0.29	0.48	0.58	0.66	0.72	0.77	0.81	0.84	0.87	0.90	0.93	**0.95**	0.96	0.98	1
Copula *PCA*	0.52	0.65	0.76	0.85	0.94	**0.95**	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99	1
Linear *PCA*	0.57	0.70	0.79	0.86	0.93	**0.95**	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
time 3	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10	comp 11	comp 12	comp 13	comp 14	comp 15
polynomial Kernel *PCA*	0.40	0.70	0.84	0.93	**0.96**	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
Gaussian Kernel *PCA*	0.26	0.45	0.55	0.63	0.70	0.75	0.80	0.83	0.87	0.90	0.93	0.95	**0.96**	0.98	1
Laplacian Kernel *PCA*	0.30	0.47	0.56	0.64	0.71	0.76	0.80	0.83	0.87	0.90	0.92	0.94	**0.96**	0.98	1
ANOVA Kernels *PCA*	0.29	0.48	0.58	0.66	0.73	0.77	0.81	0.85	0.88	0.91	0.93	**0.95**	0.97	0.98	1
Copula *PCA*	0.53	0.70	0.84	0.93	**0.96**	0.97	0.98	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
Linear *PCA*	0.57	0.70	0.79	0.86	0.91	**0.95**	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
time 4	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10	comp 11	comp 12	comp 13	comp 14	comp 15
polynomial Kernel *PCA*	0.49	0.77	0.90	0.93	**0.95**	0.96	0.96	0.97	0.97	0.98	0.99	0.99	0.99	0.99	1
Gaussian Kernel *PCA*	0.25	0.45	0.56	0.64	0.71	0.76	0.80	0.84	0.87	0.90	0.93	**0.95**	0.96	0.98	1
Laplacian Kernel *PCA*	0.31	0.48	0.57	0.65	0.71	0.77	0.81	0.84	0.87	0.90	0.92	0.94	**0.96**	0.98	1
ANOVA Kernels *PCA*	0.29	0.49	0.59	0.67	0.73	0.78	0.82	0.85	0.88	0.91	0.93	**0.95**	0.97	0.98	1
Copula *PCA*	0.63	0.83	0.94	**0.95**	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1
Linear *PCA*	0.63	0.82	0.89	0.94	**0.95**	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1
time 5	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8	comp 9	comp 10	comp 11	comp 12	comp 13	comp 14	comp 15
polynomial Kernel *PCA*	0.72	0.86	0.91	0.94	**0.95**	0.96	0.97	0.97	0.98	0.98	0.99	0.99	0.99	0.99	1
Gaussian Kernel *PCA*	0.32	0.45	0.56	0.64	0.71	0.75	0.80	0.83	0.87	0.90	0.93	**0.95**	0.96	0.98	1
Laplacian Kernel *PCA*	0.33	0.48	0.58	0.65	0.71	0.77	0.81	0.84	0.87	0.90	0.93	**0.95**	0.96	0.98	1
ANOVA Kernels *PCA*	0.29	0.48	0.59	0.67	0.73	0.78	0.81	0.85	0.88	0.91	0.94	**0.95**	0.97	0.98	1
Copula *PCA*	0.76	0.87	**0.95**	0.96	0.98	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1
Linear *PCA*	0.70	0.85	0.90	0.93	**0.96**	0.96	0.97	0.98	0.99	0.99	0.99	0.99	0.99	0.99	1

Table 3 for all 5 time points. As you can see, at the first time, the polynomial Kernel *PCA* reduced the dimension of the variables better than the other methods, so that by considering the first 4 components, it can explain (97%) of the total variance, but with the passage of time and the behavior of the variables being determined, other methods perform better. As you can see in time 5, the Copula *PCA* method explains (95%) of the total variance by considering the first 5 components, and it performs much better than the other two methods. See also, Figure 1, as a visualization representation.

In summary, it can be said that at the initial times when the behavior of the data is not known, the polynomial Kernel *PCA* and Linear *PCA* dimension reduction methods work well, but as time passes and the behavior of the variables becomes clear, the Copula *PCA* method superforms the dimension reduction in contrast of other methods.

Next, to check the stated algorithm on the generated data set, suppose we want the components to explain (95%) of the total variance ($\alpha_t = 0.95, \quad t = 1, 2, 3, 4, 5$), in this case, the results of dimension reduction on this data set are shown in Table 4.

As seen from Table 4 the best methods are Kernel *PCA*, Copula *PCA*, and Linear *PCA*. More precisely, for the first time, using the polynomial Kernel *PCA* method, the first 3 components account for 95% of the total variance while the number of components for this percentage are 7 and 6 respectively for Copula *PCA* and Linear *PCA*. However, with the passage of time and the increase of data, for example at the 5th time, in the polynomial Kernel *PCA* method, 5 components explain, 95% of the total variance, but the Copula *PCA* method, explains 95% of the total variance by 3 components and, and the Linear *PCA* needs 5 components to explain 95% of the total variance.

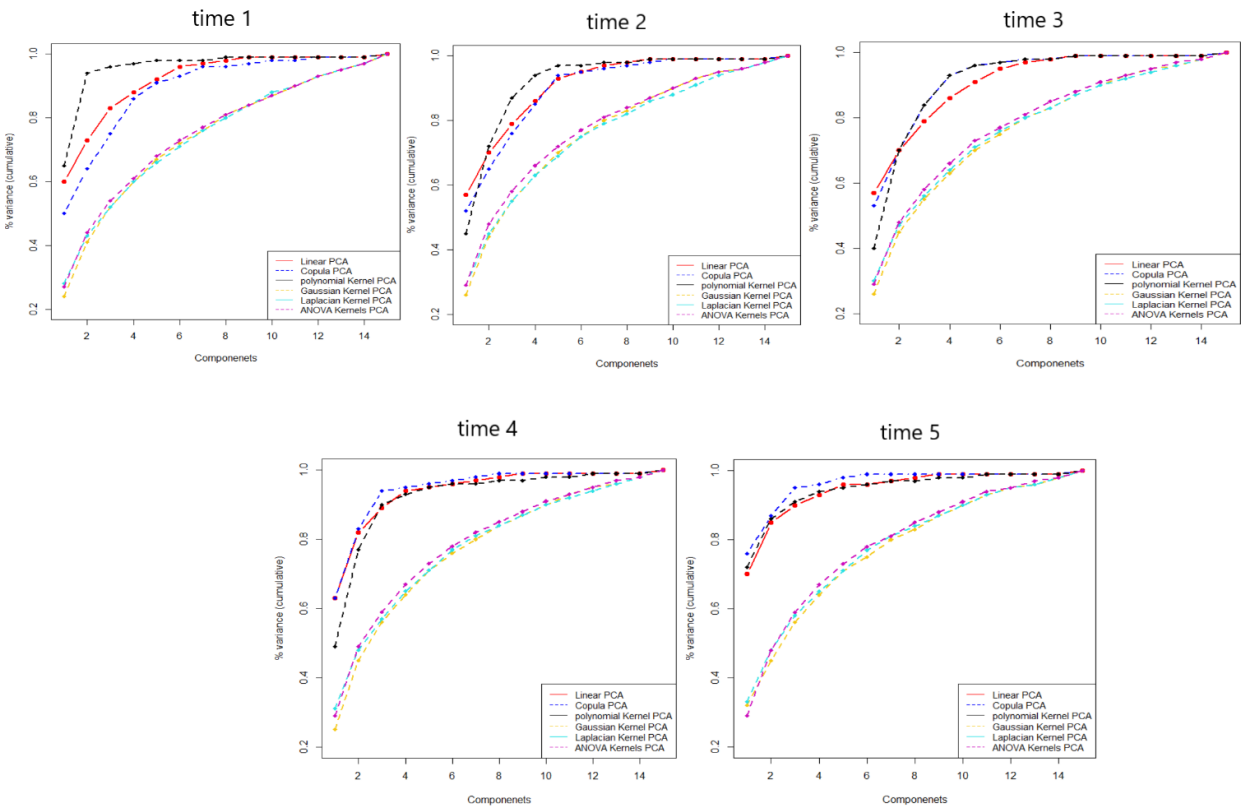

Figure 1: Percents of explained variance using six *PCA* methods at different times

Table 4: The number of components that explain 95% of the total variance in each method and at different times

time	1	2	3	4	5
polynomial Kernel *PCA*	**3**	**5**	**5**	5	5
Gaussian Kernel *PCA*	13	12	12	12	12
Laplacian Kernel *PCA*	13	13	13	13	12
ANOVA Kernels *PCA*	13	12	12	12	12
Copula *PCA*	7	6	**5**	**4**	**3**
Linear *PCA*	6	6	6	5	5

To evaluate model performance across different time points, we can use two key metrics: Reconstruction-Mean-Squared-Error (*RMSE*) (*c.f.* Milosevic *et al.* (2021)), and *Mean-Corr* (mean of correlations). *RMSE* quantifies how well a model reconstructs its input data, with lower values indicating better performance. It is calculated using the formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{l=1}^{n}(x_l - \hat{x_l})^2}$$

where n is the number of values, x_l is the original value, and $\hat{x_l}$ is the reconstructed value. *Mean-Corr*, on the other hand, assesses the relationship between original and reconstructed data, where higher values (closer to 1) show that the model preserves the patterns in the data well. It's calculated as:

$$Mean\text{-}Corr = \frac{1}{n}\sum_{l=1}^{n} Cor(x_l, \hat{x_l})$$

where $Cor(x_l, \hat{x_l})$ is the Pearson correlation between original and reconstructed data for each sample. By calculating these metrics for different models at various time intervals and organizing them into a table, you can track changes in performance, identify trends, and determine which model best maintains its effectiveness across time.

According to the output of Table 5, it can be said that Copula *PCA* has more *RMSE* and less *Mean-Corr* in the early times, but as time increases and the volume of data increases, it has a lower *RMSE* value than other models and *Mean-Corr* increases.

To examine the algorithm *SCFE-NN* in a prediction, we input variables of the neural network by the principal components that explain 95% of the total variance at each time, using the components obtained in the previous step using the Linear *PCA*, Copula *PCA*, polynomial Kernel *PCA* method. The 5-Fold cross-validation method was used for unbiased evaluation. In each fold, the parameters of the neural network were optimized separately and the obtained results are given below.

The results of the comparisons conducted are presented in the table below, As mentioned, in this research.

As shown in Table 7, the *SCFE-NN* algorithm consistently exhibits less overfitting compared to other methods. In fact, relying on the *SCFE-NN* algorithm is advisable as it provides better predictions based on the number of components it selects at any given time. It may initially select more components than other methods, but it delivers more accurate predictions. As time progresses and the function behaviors change, the *SCFE-NN* algorithm with fewer components offers better predictions.

3.2. Real data

We use our approach to carry out a feature extraction and make a prediction in online Apple stock dataset as well as *COVID*-19 dataset and the Real-time Bitcoin dataset.

Table 5: Output *RMSE* and *Mean-Corr* values for each method and at any time

time 1	*RMSE*	*Mean-Corr*
polynomial Kernel *PCA*	1.09	0.08
Gaussian Kernel *PCA*	1.09	0.06
Laplacian Kernel *PCA*	1.09	0.04
ANOVA Kernels *PCA*	1.09	0.03
Copula *PCA*	1.11	0.13
Linear *PCA*	1.24	0.16
time 2	*RMSE*	*Mean-Corr*
polynomial Kernel *PCA*	1.07	0.10
Gaussian Kernel *PCA*	1.07	0.08
Laplacian Kernel *PCA*	1.07	0.07
ANOVA Kernels *PCA*	1.07	0.05
Copula *PCA*	1.09	0.15
Linear *PCA*	1.18	0.14
time 3	*RMSE*	*Mean-Corr*
polynomial Kernel *PCA*	1.05	0.12
Gaussian Kernel *PCA*	1.05	0.09
Laplacian Kernel *PCA*	1.05	0.09
ANOVA Kernels *PCA*	1.05	0.07
Copula *PCA*	1.04	0.17
Linear *PCA*	1.15	0.11
time 4	*RMSE*	*Mean-Corr*
polynomial Kernel *PCA*	1.04	0.14
Gaussian Kernel *PCA*	1.04	0.11
Laplacian Kernel *PCA*	1.04	0.11
ANOVA Kernels *PCA*	1.04	0.11
Copula *PCA*	1.03	0.18
Linear *PCA*	1.13	0.11
time 5	*RMSE*	*Mean-Corr*
polynomial Kernel *PCA*	1.03	0.15
Gaussian Kernel *PCA*	1.03	0.12
Laplacian Kernel *PCA*	1.03	0.12
ANOVA Kernels *PCA*	1.03	0.12
Copula *PCA*	1.00	0.19
Linear *PCA*	1.10	0.12

Table 6: Values of optimized neural network parameters in each fold of each method

time	Method	Fold	units	stepmax	number-layers	*MAE*
1	linear-*PCA-NN*	1	6	$1e+06$	3	0.003
		2	4	$1e+05$	1	0.003
		3	4	$1e+06$	2	0.003
		4	4	$1e+06$	2	0.002
		5	6	$1e+06$	1	0.003
	polynomial Kernel *PCA-NN*	1	4	$1e+07$	2	0.232
		2	4	$1e+05$	1	0.259
		3	4	$1e+07$	1	0.263
		4	6	$1e+07$	1	0.295
		5	4	$1e+07$	2	0.232
	SCFE-NN	1	4	$1e+05$	2	0.003
		2	6	$1e+05$	1	0.002
		3	4	$1e+05$	1	0.003
		4	4	$1e+06$	2	0.003
		5	4	$1e+07$	1	0.003
2	linear-*PCA-NN*	1	8	$1e+06$	1	0.001
		2	6	$1e+05$	3	0.001
		3	4	$1e+05$	1	0.002
		4	8	$1e+06$	3	0.002
		5	4	$1e+06$	2	0.002
	polynomial Kernel *PCA-NN*	1	4	$1e+05$	3	0.66
		2	4	$1e+05$	1	0.57
		3	6	$1e+06$	3	0.47
		4	6	$1e+05$	3	0.55
		5	4	$1e+07$	2	0.59
	SCFE-NN	1	4	$1e+06$	2	0.001
		2	4	$1e+05$	2	0.001
		3	4	$1e+06$	2	0.002
		4	8	$1e+05$	1	0.001
		5	6	$1e+05$	1	0.001
3	linear-*PCA-NN*	1	6	$1e+05$	2	0.002
		2	6	$1e+06$	2	0.002
		3	8	$1e+07$	1	0.001
		4	6	$1e+05$	2	0.001
		5	6	$1e+05$	1	0.001
	polynomial Kernel *PCA-NN*	1	4	$1e+07$	3	0.005
		2	4	$1e+06$	2	0.001
		3	6	$1e+05$	3	0.003
		4	4	$1e+07$	3	0.013
		5	6	$1e+07$	3	0.0164
	SCFE-NN	1	8	$1e+06$	3	0.001
		2	6	$1e+07$	2	0.001
		3	6	$1e+06$	2	0.001
		4	6	$1e+06$	1	0.002
		5	4	$1e+05$	2	0.001

time	Method	Fold	units	stepmax	number-layers	*MAE*
4	linear-*PCA-NN*	1	6	$1e+05$	3	0.003
		2	8	$1e+06$	2	0.001
		3	6	$1e+05$	2	0.002
		4	8	$1e+05$	1	0.001
		5	4	$1e+05$	2	0.001
	polynomial Kernel *PCA-NN*	1	8	$1e+05$	2	0.015
		2	4	$1e+07$	3	0.017
		3	6	$1e+07$	2	0.001
		4	4	$1e+06$	1	0.001
		5	6	$1e+05$	2	0.003
	SCFE-NN	1	8	$1e+06$	3	0.001
		2	8	$1e+05$	2	0.002
		3	8	$1e+05$	2	0.003
		4	8	$1e+06$	2	0.003
		5	6	$1e+07$	2	0.002
5	linear-*PCA-NN*	1	6	$1e+05$	2	0.004
		2	4	$1e+06$	2	0.003
		3	8	$1e+05$	1	0.003
		4	4	$1e+05$	2	0.002
		5	6	$1e+05$	1	0.002
	polynomial Kernel *PCA-NN*	1	8	$1e+06$	3	0.015
		2	8	$1e+05$	2	0.016
		3	8	$1e+07$	3	0.017
		4	8	$1e+07$	3	0.016
		5	8	$1e+06$	3	0.016
	SCFE-NN	1	6	$1e+06$	3	0.003
		2	6	$1e+07$	2	0.001
		3	4	$1e+06$	2	0.003
		4	4	$1e+06$	1	0.004
		5	6	$1e+05$	2	0.004

3.2.1. Apple stock dataset

A real-time streaming dataset for implementing our *SCFE-NN* algorithm is the stock data of Apple Inc[2]. This dataset is updated daily. We consider 6 variables: "AAPL.Open", "AAPL.High", "AAPL.Low", "AAPL.Close", "AAPL.Volume", "AAPL.Adjusted" from this dataset from the date of 1-4-2008 to 20-2-2024, and the data is segmented into 4 timeframes. Assuming AAPL.Open, AAPL.High, AAPL.Low, AAPL.Volume and AAPL.Adjusted as predictors and as the response variable, we implemented algorithm 1. We first extract the most informative components from the predictor variables. Table 8 expresses the percentage of explained variance using our algorithm as well as other alternative methods.

As shown in this table, Copula *PCA* gradually explains a higher percentage of variance with fewer components over time.

[2]https://finance.yahoo.com/quote/AAPL/history

Table 7: Comparison results of algorithm model *SCFE-NN* with other models at different times

time	Estimation	linear-*PCA-NN*	polynomial Kernel *PCA-NN*	*SCFE-NN*	*PCR*
1	mean of *RMSE* train	0.011	0.300	0.008	0.690
	SD of *RMSE* train	0.001	0.016	0.001	0.070
	mean of *RMSE* test	0.210	0.602	0.136	0.600
	SD of *RMSE* test	0.197	0.500	0.164	0.150
2	mean of *RMSE* train	0.006	0.180	0.005	0.990
	SD of *RMSE* train	0.001	0.027	0.000	0.050
	mean of *RMSE* test	0.073	0.662	0.057	0.840
	SD of *RMSE* test	0.120	0.429	0.066	0.180
3	mean of *RMSE* train	0.008	0.025	0.001	2.240
	SD of *RMSE* train	0.004	0.022	0.043	0.600
	mean of *RMSE* test	0.092	0.159	0.003	1.590
	SD of *RMSE* test	0.045	0.378	0.044	0.710
4	mean of *RMSE* train	0.007	0.011	0.001	2.020
	SD of *RMSE* train	0.009	0.000	0.003	0.620
	mean of *RMSE* test	0.102	0.232	0.050	1.810
	SD of *RMSE* test	0.097	0.078	0.042	0.950
5	mean of *RMSE* train	0.008	0.027	0.008	0.70
	SD of *RMSE* train	0.003	0.002	0.001	0.280
	mean of *RMSE* test	0.092	0.201	0.078	0.340
	SD of *RMSE* test	0.064	0.039	0.061	0.170

For example, using the first extracted component, at time 1, the best method is Linear *PCA* (83.3%), and the second rank is for Copula *PCA* (81.2%). However, as time moves forward, the Copula *PCA* gradually will be better until at the final time, Copula *PCA* is the superior (90.7%). Also, if we aim to explain 99.9% of the total variance, we can specify the least number of needed components for each method. Table 11 summarizes the number of components that we need to explain 99.9% of the total variance using polynomial Kernel *PCA*, Gaussian Kernel *PCA*, Laplacian Kernel *PCA*, *ANOVA* Kernels *PCA*, Copula *PCA* and Linear *PCA*. In a nutshell, at time 1, Linear *PCA* is the best while at times 3 and 4 Copula *PCA* is highly accomplished.

Finally, due to the close competition between Copula *PCA*, Linear-*PCA*, and polynomial Kernel *PCA*, for the implementation of the *SCFE-NN* algorithm, we utilize the components extracted by these 3 methods. We execute the algorithm *SCFE-NN* and predict the response variable AAPL.Close. The differences between the predicted values and the true values will be our criterion to assess the algorithm. For this purpose, we split the data into train and test sets with respectively 70% and 30% of cases. The prediction results are stated in Table 11. As can be realized from this table, the *SCFE-NN* algorithm possesses less overfitting and provides better predictions with any number of components it extracts. Also, the values of the optimized parameters of the neural network in each fold of each method in this dataset are given in Table 10.

Table 8: The results of the percentage of variance explained by six *PCA* methods at different times in real dataset (Apple stock)

time 1 (2008/04/01-2012/03/30)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.711	0.922	0.975	**0.999**	1
Gaussian Kernel *PCA*	0.554	0.775	0.887	0.956	**1**
Laplacian Kernel *PCA*	0.561	0.753	0.866	0.948	**1**
ANOVA Kernels *PCA*	0.503	0.732	0.854	0.932	**1**
Copula *PCA*	0.812	0.985	0.998	**0.999**	1
Linear *PCA*	0.833	0.994	**0.999**	0.999	1
time 2 (2008/04/01-2016/03/31)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.644	0.932	0.977	**0.999**	1
Gaussian Kernel *PCA*	0.565	0.818	0.915	0.964	**1**
Laplacian Kernel *PCA*	0.588	0.774	0.886	0.957	**1**
ANOVA Kernels *PCA*	0.470	0.754	0.862	0.948	**1**
Copula *PCA*	0.857	0.997	0.998	**0.999**	1
Linear *PCA*	0.883	0.993	0.997	**0.999**	1
time 3 (2008/04/01-2020/03/31)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.770	0.971	0.989	**0.999**	1
Gaussian Kernel *PCA*	0.544	0.817	0.903	0.964	**1**
Laplacian Kernel *PCA*	0.552	0.773	0.874	0.945	**1**
ANOVA Kernels *PCA*	0.463	0.764	0.866	0.944	**1**
Copula *PCA*	0.887	0.998	**0.999**	0.999	1
Linear *PCA*	0.893	0.995	0.996	**0.999**	1
time 4 (2008/04/01-2024/02/20)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.688	0.981	0.996	**0.999**	1
Gaussian Kernel *PCA*	0.612	0.864	0.942	0.984	**1**
Laplacian Kernel *PCA*	0.582	0.814	0.900	0.965	**1**
ANOVA Kernels *PCA*	0.523	0.794	0.897	0.964	**1**
Copula *PCA*	0.907	0.998	**0.999**	0.999	1
Linear *PCA*	0.864	0.996	0.998	**0.999**	1

Table 9: The number of components that explain 99.9% of the total variance in each method and at different times in the real data set (Apple stock)

time	**1**	**2**	**3**	**4**
polynomial Kernel *PCA*	4	**4**	4	4
Gaussian Kernel *PCA*	5	5	5	5
Laplacian Kernel *PCA*	5	5	5	5
ANOVA Kernels *PCA*	5	5	5	5
Copula *PCA*	4	**4**	**3**	**3**
Linear *PCA*	**3**	**4**	4	4

Table 10: Values of the optimal parameters of the neural network in the Apple stock dataset

time	Method	Fold	units	stepmax	number-layers	*MAE*
1	linear-*PCA-NN*	1	4	$1e+06$	1	0.001
		2	4	$1e+07$	1	0.001
		3	4	$1e+06$	2	0.001
		4	4	$1e+07$	2	0.000
		5	6	$1e+07$	2	0.001
	polynomial Kernel *PCA-NN*	1	6	$1e+07$	3	0.004
		2	6	$1e+06$	2	0.005
		3	6	$1e+05$	2	0.004
		4	6	$1e+05$	2	0.004
		5	8	$1e+05$	2	0.005
	SCFE-NN	1	4	$1e+07$	3	0.001
		2	8	$1e+06$	3	0.001
		3	8	$1e+05$	1	0.001
		4	6	$1e+05$	3	0.000
		5	4	$1e+07$	2	0.001
2	linear-*PCA-NN*	1	6	$1e+06$	2	0.001
		2	4	$1e+05$	1	0.001
		3	4	$1e+06$	1	0.001
		4	8	$1e+05$	3	0.001
		5	6	$1e+05$	2	0.001
	polynomial Kernel *PCA-NN*	1	6	$1e+05$	2	0.003
		2	6	$1e+07$	2	0.003
		3	6	$1e+07$	2	0.003
		4	8	$1e+07$	2	0.003
		5	8	$1e+06$	2	0.004
	SCFE-NN	1	6	$1e+07$	2	0.001
		2	8	$1e+06$	1	0.001
		3	6	$1e+07$	1	0.001
		4	6	$1e+05$	1	0.001
		5	8	$1e+05$	2	0.001
3	linear-*PCA-NN*	1	6	$1e+05$	3	0.001
		2	8	$1e+05$	3	0.001
		3	4	$1e+05$	2	0.000
		4	6	$1e+05$	3	0.001
		5	8	$1e+07$	1	0.001
	polynomial Kernel *PCA-NN*	1	6	$1e+06$	2	0.004
		2	6	$1e+05$	3	0.003
		3	8	$1e+05$	2	0.003
		4	6	$1e+05$	2	0.003
		5	8	$1e+05$	2	0.004
	SCFE-NN	1	6	$1e+06$	1	0.001
		2	8	$1e+06$	2	0.000
		3	8	$1e+05$	2	0.000
		4	8	$1e+05$	2	0.001
		5	6	$1e+05$	2	0.001

time	Method	Fold	units	stepmax	number-layers	*MAE*
4	linear-*PCA-NN*	1	8	$1e+05$	3	0.002
		2	8	$1e+07$	3	0.001
		3	6	$1e+07$	2	0.000
		4	8	$1e+06$	1	0.001
		5	6	$1e+06$	2	0.002
	polynomial Kernel *PCA-NN*	1	6	$1e+06$	2	0.003
		2	6	$1e+05$	3	0.003
		3	8	$1e+07$	2	0.003
		4	6	$1e+05$	2	0.004
		5	8	$1e+05$	2	0.004
	SCFE-NN	1	6	$1e+05$	2	0.001
		2	4	$1e+06$	2	0.001
		3	8	$1e+05$	2	0.000
		4	8	$1e+06$	2	0.001
		5	6	$1e+05$	3	0.000

Table 11: Comparison results of algorithm model *SCFE-NN* with other models at different times in real time (Apple stock)

time	Estimation	linear-*PCA-NN*	polynomial Kernel *PCA-NN*	*SCFE-NN*	*PCR*
1	mean of *RMSE* train	0.003	0.007	0.003	4.98
	SD of *RMSE* train	0.001	0.001	0.000	0.01
	mean of *RMSE* test	0.003	0.014	0.003	4.92
	SD of *RMSE* test	0.001	0.009	0.001	0.01
2	mean of *RMSE* train	0.002	0.007	0.002	6.3
	SD of *RMSE* train	0.000	0.004	0.000	0.02
	mean of *RMSE* test	0.001	0.012	0.002	6.3
	SD of *RMSE* test	0.000	0.000	0.000	0.05
3	mean of *RMSE* train	0.002	0.009	0.001	25.44
	SD of *RMSE* train	0.001	0.005	0.000	0.15
	mean of *RMSE* test	0.003	0.011	0.001	25.13
	SD of *RMSE* test	0.001	0.001	0.000	0.3
4	mean of *RMSE* train	0.004	0.010	0.002	27.77
	SD of *RMSE* train	0.000	0.005	0.000	0.13
	mean of *RMSE* test	0.003	0.014	0.002	27.64
	SD of *RMSE* test	0.001	0.007	0.000	0.21

3.2.2. *COVID*-19 dataset

The second dataset we considered and implemented our algorithm is the *COVID*-19 dataset. This dataset has been recorded daily from 27-7-2020 to 22-11-2020[3]. In this experiment, we considered 8 variables including "Confirmed", "Deaths", "Recovered", "Active", "New.cases", "New.recovered", "Recovered/100.Cases", "Deaths/100.Recovered". Suppose we aim to perform dimension reduction on this considered dataset. To express the method described in this article well, we considered 3 times of 11 days from this data set and implemented the described methods on them. The results are presented in Table 12 and Figure 2.

Table 12: The results of the percentage of variance explained by Linear *PCA* and Copula *PCA* and *PCA* methods at different times in real dataset

time 1 (2020/01/22-2020/01/31)	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8
polynomial Kernel *PCA*	0.723	0.923	0.958	0.975	0.988	**0.994**	0.997	1
Gaussian Kernel *PCA*	0.432	0.718	0.848	0.911	0.947	0.976	**0.994**	1
Laplacian Kernel *PCA*	0.426	0.648	0.755	0.826	0.878	0.926	0.966	**1**
ANOVA Kernels *PCA*	0.407	0.684	0.824	0.890	0.928	0.962	0.989	**1**
Copula *PCA*	0.783	0.886	0.931	0.969	**0.996**	0.997	0.998	1
Linear *PCA*	0.768	0.940	0.975	**0.995**	0.995	0.998	0.999	1
time 2 (2020/01/22-2020/02/10)	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8
polynomial Kernel *PCA*	0.843	0.913	0.948	0.978	0.989	**0.995**	0.998	1
Gaussian Kernel *PCA*	0.503	0.750	0.864	0.932	0.957	0.979	**0.992**	1
Laplacian Kernel *PCA*	0.505	0.695	0.803	0.867	0.912	0.946	0.976	**1**
ANOVA Kernels *PCA*	0.473	0.716	0.837	0.915	0.947	0.971	0.988	**1**
Copula *PCA*	0.775	0.928	0.957	0.983	**0.996**	0.997	0.998	1
Linear *PCA*	0.827	0.953	0.985	**0.994**	0.999	0.999	0.999	1
time 3 (2020/01/22-2020/02/20)	comp 1	comp 2	comp 3	comp 4	comp 5	comp 6	comp 7	comp 8
polynomial Kernel *PCA*	0.596	0.779	0.921	0.966	0.981	**0.992**	0.997	1
Gaussian Kernel *PCA*	0.469	0.697	0.779	0.847	0.901	0.950	0.981	**1**
Laplacian Kernel *PCA*	0.481	0.661	0.749	0.814	0.872	0.925	0.973	**1**
ANOVA Kernels *PCA*	0.452	0.691	0.773	0.836	0.892	0.936	0.976	**1**
Copula *PCA*	0.760	0.942	0.978	**0.999**	0.999	0.999	0.999	1
Linear *PCA*	0.769	0.914	0.958	0.987	**0.999**	0.999	0.999	1

As can be understood from Table 12, aiming to reach 99% of the explained variance in the first time, polynomial Kernel *PCA* needs 6 components, Gaussian Kernel *PCA* method needs 7 components, the Laplacian Kernel *PCA*, *ANOVA* Kernels *PCA*, Copula *PCA*, and Linear *PCA* need respectively, 8, 8, 5 and 4 components. But at the last time, these numbers changed into 6, 8, 8, 8, 8, 4, and 5 for polynomial Kernel *PCA*, Gaussian Kernel *PCA*, Laplacian Kernel *PCA*, *ANOVA* Kernels *PCA*, Copula *PCA*, and Linear *PCA* respectively, which shows a significance improvement of Copula *PCA*.

Regarding constructing the *SCFE-NN* algorithm in this dataset, we assume New.deaths as the response variable and predict this variable using the principal components that explained 99% of the total variance in the previous best methods, *i.e.*, polynomial Kernel *PCA*, Linear *PCA*, and Copula *PCA*. We set the extracted components as inputs of a neural network, and the results of comparing different models at the specified times are presented in Table 15, Also, the values of the optimized parameters of the neural network in each fold

[3]https://www.kaggle.com/datasets/imdevskp/corona-virus-report

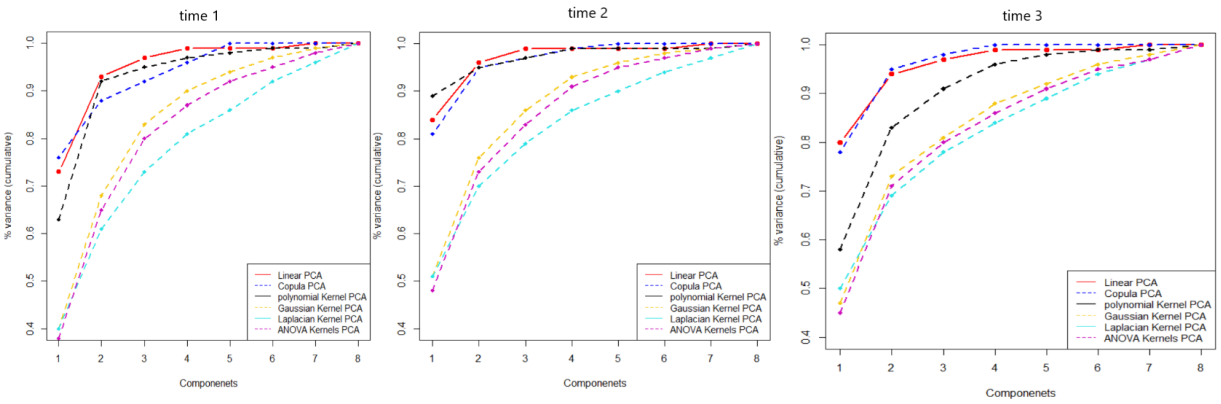

Figure 2: Percents of explained variance using linear *PCA*, Copula *PCA* and kernel *PCA* at different times in real-data (*COVID*-19)

Table 13: The number of components that explain 99% of the total variance in each method and at different times in the real data set

time	1	2	3
polynomial Kernel *PCA*	6	6	6
Gaussian Kernel *PCA*	7	7	8
Laplacian Kernel *PCA*	8	8	8
ANOVA Kernels *PCA*	8	8	8
Copula *PCA*	5	5	**4**
Linear *PCA*	**4**	**4**	5

of each method in this dataset are given in Table 14. Based on this table, the *SCFE-NN* algorithm consistently exhibits less overfitting compared to other methods and provides a more accurate prediction.

3.2.3. Bitcoin dataset

The third data set that we implemented our algorithm on is the Bitcoin data set, which in this study focuses on daily data from 1-1-2025 to 5-1-2025[4] and we have also considered 5 variables"Bitcoin.Price", "Bitcoin.Open", "Bitcoin.High", "Bitcoin.Low" and "Bitcoin.Change" from this data set. In this regard, 3 time periods 4-1-2025 to 6-1-2025, 4-1-2025 to 7-1-2025 and 4-1-2025 to 8-1-2025 have been considered for this research. The results are presented in Table 16 and Figure 6. Considering that the purpose of presenting this dataset is to implement the described method on daily data, for this reason, the bootstrap method has been used to increase the volume of data for a more accurate analysis.

As can be understood from Table 16, aiming to reach 99% of the explained variance in the first time, polynomial Kernel *PCA* needs 5 components, Gaussian Kernel *PCA* method needs 5 components, the Laplacian Kernel *PCA*, *ANOVA* Kernels *PCA*, Copula *PCA*, and Linear *PCA* need respectively, 5, 5, 3 and 2 components. But at the last time, these numbers changed into 5, 5, 5, 5, 3, and 3 for polynomial Kernel *PCA*, Gaussian Kernel *PCA*, Laplacian

[4]https://www.investing.com/crypto/bitcoin/historical-data

Table 14: Values of the optimal parameters of the neural network in the *COVID-19* dataset

time	Method	Fold	units	stepmax	number-layers	*MAE*
1	linear-*PCA-NN*	1	8	$1e+06$	2	0.215
		2	8	$1e+05$	1	0.109
		3	68	$1e+06$	3	0.070
		4	6	$1e+05$	3	0.076
		5	8	$1e+05$	3	0.165
	polynomial Kernel *PCA-NN*	1	4	$1e+05$	3	0.046
		2	8	$1e+07$	1	0.010
		3	8	$1e+05$	2	0.008
		4	4	$1e+07$	3	0.005
		5	4	$1e+05$	1	0.001
	SCFE-NN	1	6	$1e+07$	2	0.347
		2	4	$1e+05$	1	0.492
		3	6	$1e+06$	1	0.507
		4	4	$1e+05$	1	0.362
		5	8	$1e+06$	2	0.140
2	linear-*PCA-NN*	1	8	$1e+06$	3	0.677
		2	8	$1e+06$	1	0.650
		3	8	$1e+07$	1	0.705
		4	6	$1e+06$	2	0.641
		5	8	$1e+07$	1	0.558
	polynomial Kernel *PCA-NN*	1	4	$1e+06$	3	0.027
		2	8	$1e+07$	1	0.022
		3	6	$1e+06$	3	0.040
		4	8	$1e+07$	3	0.015
		5	8	$1e+05$	2	0.016
	SCFE-NN	1	4	$1e+05$	3	0.639
		2	8	$1e+06$	3	0.557
		3	6	$1e+07$	3	0.545
		4	4	$1e+05$	3	0.700
		5	8	$1e+05$	3	0.486
3	linear-*PCA-NN*	1	8	$1e+05$	3	0.663
		2	8	$1e+07$	3	0.688
		3	4	$1e+07$	2	0.639
		4	8	$1e+05$	3	0.558
		5	8	$1e+07$	1	0.738
	polynomial Kernel *PCA-NN*	1	6	$1e+06$	2	0.040
		2	4	$1e+06$	3	0.033
		3	8	$1e+07$	3	0.178
		4	4	$1e+05$	2	0.160
		5	8	$1e+05$	2	0.125
	SCFE-NN	1	4	$1e+06$	1	0.506
		2	8	$1e+06$	2	0.527
		3	8	$1e+05$	2	0.491
		4	8	$1e+05$	2	0.523
		5	8	$1e+07$	3	0.440

Table 15: Comparison results of algorithm model *SCFE-NN* with other models at different times in real time

time	Estimation	linear-*PCA-NN*	polynomial Kernel *PCA-NN*	*SCFE-NN*	*PCR*
1	mean of *RMSE* train	0.764	0.032	0.866	1.79
	SD of *RMSE* train	0.270	0.021	0.120	0.33
	mean of *RMSE* test	0.977	0.999	0.925	1.17
	SD of *RMSE* test	0.502	0.703	0.180	0.97
2	mean of *RMSE* train	0.970	0.080	0.970	1.58
	SD of *RMSE* train	0.028	0.026	0.127	0.14
	mean of *RMSE* test	1.05	1.022	0.982	1.30
	SD of *RMSE* test	0.140	1.437	0.136	0.64
3	mean of *RMSE* train	0.980	0.979	0.957	1.39
	SD of *RMSE* train	0.068	0.251	0.071	0.17
	mean of *RMSE* test	0.933	1.02	0.962	1.51
	SD of *RRMSE* test	0.308	0.601	0.263	0.85

Table 16: The results of the percentage of variance explained by Linear *PCA* and Copula *PCA* and *PCA* methods at different times in real dataset

time 1 (2025/01/04-2025/01/06)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.522	0.721	0.845	0.965	**1**
Gaussian Kernel *PCA*	0.468	0.686	0.839	0.969	**1**
Laplacian Kernel *PCA*	0.401	0.607	0.774	0.926	**1**
ANOVA Kernels *PCA*	0.404	0.625	0.793	0.949	**1**
Copula *PCA*	0.861	0.970	**0.996**	0.998	1
Linear *PCA*	0.952	**0.999**	0.999	0.999	1
time 2 (2025/01/04-2025/01/07)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.441	0.797	0.890	0.961	**1**
Gaussian Kernel *PCA*	0.308	0.576	0.792	0.918	**1**
Laplacian Kernel *PCA*	0.269	0.541	0.741	0.886	**1**
ANOVA Kernels *PCA*	0.304	0.530	0.744	0.890	**1**
Copula *PCA*	0.667	0.980	**0.999**	0.999	1
Linear *PCA*	0.714	**0.996**	0.999	0.999	1
time 3 (2025/01/04-2025/01/08)	comp 1	comp 2	comp 3	comp 4	comp 5
polynomial Kernel *PCA*	0.416	0.643	0.815	0.932	**1**
Gaussian Kernel *PCA*	0.291	0.540	0.722	0.896	**1**
Laplacian Kernel *PCA*	0.270	0.519	0.697	0.866	**1**
ANOVA Kernels *PCA*	0.296	0.503	0.686	0.862	**1**
Copula *PCA*	0.636	0.957	**0.995**	0.998	1
Linear *PCA*	0.583	0.945	**0.999**	0.999	1

Kernel *PCA*, *ANOVA* Kernels *PCA*, Copula *PCA*, and Linear *PCA* respectively, which shows a significance improvement of Copula *PCA*.

Table 17: The number of components that explain 99% of the total variance in each method and at different times in the real data set

time	1	2	3
polynomial Kernel *PCA*	5	5	5
Gaussian Kernel *PCA*	5	5	5
Laplacian Kernel *PCA*	5	5	5
ANOVA Kernels *PCA*	5	5	5
Copula *PCA*	3	3	**3**
Linear *PCA*	**2**	**2**	**3**

Regarding the construction of the *SCFE-NN* algorithm for this dataset, we consider Volume as the response variable. This variable is predicted using the principal components that account for 99% of the total variance, derived from the previously best-performing methods: polynomial Kernel *PCA*, Linear *PCA*, and Copula *PCA*. The extracted components are utilized as inputs for a neural network. The results comparing different models at specified times are presented in Table 19. Additionally, the optimized parameters of the neural network for each fold of each method in this dataset are provided in Table 18. According to this table, the *SCFE-NN* algorithm consistently demonstrates less overfitting compared to other methods, resulting in more accurate predictions.

4. Conclusion

When dealing with streaming data, quick decision-making and data management is of great importance because when faced with such data, the number of observations is high and variables are often interdependent. Additionally, non-linear relationships may become apparent over time with an increase in the number of observations in the variables. Therefore, we were looking for a method that takes these issues into account and improves prediction accuracy. Our proposed Stream Copula Feature Extraction (*SCFE*) algorithm benefits using Kendall's τ association measure to extract the best informative components. We investigate the performance of our algorithm using a simulation dataset, the online streaming real dataset of Apple Inc., and the recorded dataset of *COVID*-19. Compared to other alternative feature extraction methods, we showed that our approach yields more percentage of explained variance and our *SCFE-NN* approach provides good results in neural network prediction analysis, also showed in the Bitcoin data set that when the amount of data is small, the proposed algorithm is equal to previous methods such as *PCA*, and as time goes on, the algorithm explains a higher percentage of variance with a smaller number of components.

Our method can be extended in many fashions. In this study, the copula function was used to extract features in data streams. It is recommended for future research to utilize the copula function in other dimension reduction methods such as feature selection. Also, although we have used a neural network algorithm in the prediction part, one may consider using other machine learning methods such as a deep learning approach in the prediction analysis, on the other hand, to increase the accuracy of the models, the number of layers,

Table 18: Values of the optimal parameters of the neural network in the Bitcoin dataset

time	Method	Fold	units	stepmax	number-layers	*MAE*
1	linear-*PCA-NN*	1	6	$1e+06$	1	24.61
		2	4	$1e+05$	1	26.82
		3	4	$1e+05$	3	26.27
		4	6	$1e+05$	1	24.03
		5	8	$1e+05$	3	26.49
	polynomial Kernel *PCA-NN*	1	6	$1e+07$	1	0.0014
		2	6	$1e+07$	3	0.0003
		3	6	$1e+06$	2	0.000
		4	6	$1e+05$	3	0.001
		5	6	$1e+07$	3	0.003
	SCFE-NN	1	4	$1e+05$	1	26.82
		2	6	$1e+06$	2	22.87
		3	4	$1e+05$	1	25.95
		4	4	$1e+05$	2	21.32
		5	6	$1e+07$	2	21.32
2	linear-*PCA-NN*	1	8	$1e+05$	1	25.80
		2	8	$1e+06$	2	29.97
		3	8	$1e+06$	1	28.61
		4	6	$1e+07$	2	12.36
		5	8	$1e+05$	1	25.78
	polynomial Kernel *PCA-NN*	1	8	$1e+06$	3	0.000
		2	4	$1e+06$	1	0.011
		3	8	$1e+05$	3	0.006
		4	8	$1e+07$	3	0.000
		5	6	$1e+05$	2	0.004
	SCFE-NN	1	6	$1e+05$	3	22.29
		2	4	$1e+05$	3	24.73
		3	6	$1e+05$	1	27.98
		4	4	$1e+06$	3	22.60
		5	8	$1e+06$	2	29.64
3	linear-*PCA-NN*	1	6	$1e+05$	1	23.49
		2	8	$1e+05$	2	26.87
		3	6	$1e+07$	1	25.52
		4	6	$1e+07$	2	19.41
		5	4	$1e+05$	2	29.81
	polynomial Kernel *PCA-NN*	1	8	$1e+07$	2	0.005
		2	8	$1e+06$	1	0.002
		3	6	$1e+05$	1	0.013
		4	8	$1e+06$	1	0.003
		5	6	$1e+07$	3	0.004
	SCFE-NN	1	8	$1e+07$	1	13.22
		2	4	$1e+07$	1	22.00
		3	6	$1e+07$	1	26.72
		4	8	$1e+07$	3	22.98
		5	6	$1e+06$	3	27.94

Table 19: Comparison results of algorithm model *SCFE-NN* with other models at different times in real time

time	Estimation	linear-*PCA-NN*	polynomial Kernel *PCA-NN*	*SCFE-NN*	*PCR*
1	mean of *RMSE* train	26.394	0.008	22.94	34.258
	SD of *RMSE* train	0.554	0.012	2.37	2.752
	mean of *RMSE* test	32.297	27.89	25.58	37.003
	SD of *RMSE* test	3.225	11.727	3.021	3.985
2	mean of *RMSE* train	30.324	0.019	28.162	37.658
	SD of *RMSE* train	1.205	0.052	1.632	2.702
	mean of *RMSE* test	37.747	44.121	34.362	48.256
	SD of *RMSE* test	5.643	13.910	5.323	9.602
3	mean of *RMSE* train	29.618	8.65	27.325	42.369
	SD of *RMSE* train	1.401	12.43	1.39	2.963
	mean of *RMSE* test	35.239	58.49	30.026	49.521
	SD of *RMSE* test	3.168	24.725	3.125	10.325

units and stepmax of the neural network have been optimized.

Also, besides using Kendall's τ association measures to extract components, one may interested in extracting these components using other copula-based concordance measures such as Spearman's ρ, Spearman's foorules ϕ and Gini's γ, see *e.g.* Mesiar *et al.* (2024). We aim to extend these results using a linear combination of concordance measures in our next ongoing work but over on going work will be extending this result to sum predictive models such as multiple regression or principal component regression (*PCR*).

Acknowledgments

The authors are thankful to the editor and referees for their constructive and helpful comments which have significantly improved the article.

Conflict of interest

The authors do not have any financial or non-financial conflict of interest to declare for the research work included in this article.

References

Bahri, M., Bifet, A., Maniu, S., and Gomes, H. M. (2020). Survey on feature transformation techniques for data streams. In Bessiere, C., editor, *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2020)*, pages 4796–4802. Survey track.

Carrillo, J., Nieto, M., Velez, J., and Velez, D. (2021). A new machine learning forecasting algorithm based on bivariate copula functions. *Forecasting*, **3**, 355–376.

Cuvelier, E. and Noirhomme-Fraiture, M. (2005). Clayton copula and mixture decomposition. In *Applied Stochastic Models and Data Analysis (ASMDA 2005)*, pages 699–708. ENST Bretagne, Brest, France.

Fukunaga, K. and Short, R. D. (1978). Nonlinear feature extraction with a general criterion function. *IEEE Transactions on Information Theory*, **24**, 600–607.

Houari, R., Bounceur, A., Kechadi, M.-T., Tari, A.-K., and Euler, R. (2016). Dimensionality reduction in data mining: A copula approach. *Expert Systems with Applications*, **64**, 247–260.

Jia, W., Sun, M., Lian, J., and Hou, S. (2022). Feature dimensionality reduction: A review. *Complex & Intelligent Systems*, **8**, 2663–2693.

Krawczak, M. and Szkatuła, G. (2014). An approach to dimensionality reduction in time series. *Information Sciences*, **260**, 15–36.

Margara, A. and Rabl, T. (2019). Definition of data streams. In Sakr, S. and Zomaya, A. Y., editors, *Encyclopedia of Big Data Technologies*, pages 648–652. Springer International Publishing.

Mesiar, R., Kolesárová, A., Sheikhi, A., and Shvydka, S. (2024). Convex weak concordance measures and their constructions. *Fuzzy Sets and Systems*, **478**, 108841.

Milosevic, S., Frank, P., Leike, R. H., Müller, A., and Enßlin, T. A. (2021). Bayesian decomposition of the galactic multi-frequency sky using probabilistic autoencoders. *Astronomy & Astrophysics*, **650**, A100.

Misic, M., Durdevic, D., and Tomasevic, M. (2012). Evolution and trends in gpu computing. In *Proceedings of MIPRO 2012, 35th International Convention*, pages 289–294.

Mutlag, W., Ali, S., Mosad, Z., and Ghrabat, B. H. (2020). Feature extraction methods: A review. *Journal of Physics: Conference Series*, **1591**, 012028.

Nelsen, R. B. (2006). *An Introduction to Copulas*. Springer, New York, NY, USA.

Park, C. and Lee, G.-H. (2020). Comparison of incremental linear dimension reduction methods for streaming data. *Pattern Recognition Letters*, **135**.

Park, J.-H., Sriram, T., and Yin, X. (2010). Dimension reduction in time series. *Statistica Sinica*, **20**, 747–770.

Qu, X. (2012). Feature extraction by combining independent subspaces analysis and copula techniques. In *Proceedings of the 2012 International Conference on System Science and Engineering (ICSSE 2012)*, pages 326–329.

Ram, J. (2023). A review on gpu architectures and programming. In *Proceedings of the 2nd National Conference on Engineering Applications of Emerging Technology (in association with International Journal of Scientific Research in Science, Engineering and Technology)*.

Salem, N. and Hussein, S. (2019). Data dimensional reduction and principal components analysis. *Procedia Computer Science*, **163**, 292–299.

Sarveshwaran, V., Sevugan, A., and Swamidason, I. T. J. (2019). A review of dimensionality reduction techniques for efficient computation. *Procedia Computer Science*, **165**, 104–111.

Sheikhi, A., Arad, F., and Mesiar, R. (2022a). A heteroscedasticity diagnostic of a regression analysis with copula dependent random variables. *Brazilian Journal of Probability and Statistics*, **36**, 1–19.

Sheikhi, A., Mesiar, R., and Holeňa, M. (2022b). A dimension reduction in neural network using copula matrix. *International Journal of General Systems*, **52**, 1–16.

Simard, C. and Remillard, B. (2013). Forecasting time series with multivariate copulas. SSRN Electronic Journal, doi:10.2139/ssrn.2295120.

Toharudin, T., Caraka, R., Bachrudin, A., Abu Bakar, S., and Ambarsari, D. (2019). Copula-based feedforward neural network genetic algorithm in cargo forecasting. *International Journal of Advances in Soft Computing and its Applications*, **11**, 22–33.

Upadhyay, D., Jain, S., and Jain, A. (2013). Comparative analysis of various data stream mining procedures and various dimension reduction techniques. *International Journal of Advanced Research in Computer Science*, **4**.

van der Maaten, L., Postma, E. O., and van den Herik, H. J. (2009). Dimensionality reduction: A comparative review. *Journal of Machine Learning Research*, **10**, 66–71.

Wang, L., Guo, X., Zeng, J.-C., and Hong, Y. (2010). Using gumbel copula and empirical marginal distribution in estimation of distribution algorithm. In *3rd International Workshop on Advanced Computational Intelligence (IWACI 2010)*, pages 583–587.

Wang, Q. and Megalooikonomou, V. (2008). A dimensionality reduction technique for efficient time series similarity analysis. *Information Systems*, **33**, 115–132.

Witten, D., Tibshirani, R., and Hastie, T. (2009). A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, **10**, 515–534.

Xie, L., Guo, W., Wei, H., Tang, Y., and Tao, D. (2020). Efficient unsupervised dimension reduction for streaming multiview data. *IEEE Transactions on Cybernetics*, **52**, 1772–1784.

Yan, J., Zhang, B., Liu, N., Yan, S., Cheng, Q., Fan, W., Yang, Q., Xi, W., and Chen, Z. (2006). Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE Transactions on Knowledge and Data Engineering*, **18**, 320–333.

Zeng, Z. and Wang, T. (2022). Neural copula: A unified framework for estimating generic high-dimensional copula functions. arXiv preprint arXiv:2205.15031 [stat.ML].

Zheng, W., Chen, S., Fu, Z., Li, J., and Yang, J. (2022). Streaming feature selection via graph diffusion. *Information Sciences*, **618**, 150–168.

Zhu, Q. (2010). Reformative nonlinear feature extraction using kernel MSE. *Neurocomputing*, **73**, 3334–3337.

ANNEXURE

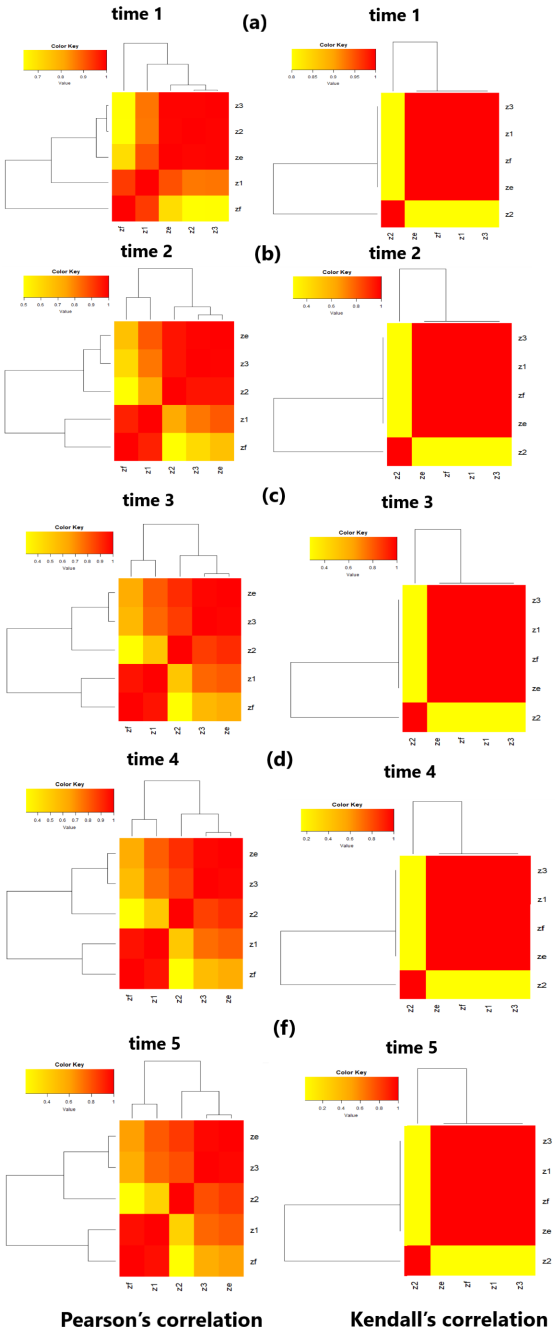

Figure 3: Heatmap plots: Left side) Pearson's correlation from time one to time five, Right side) Kendall's association from time one to time five

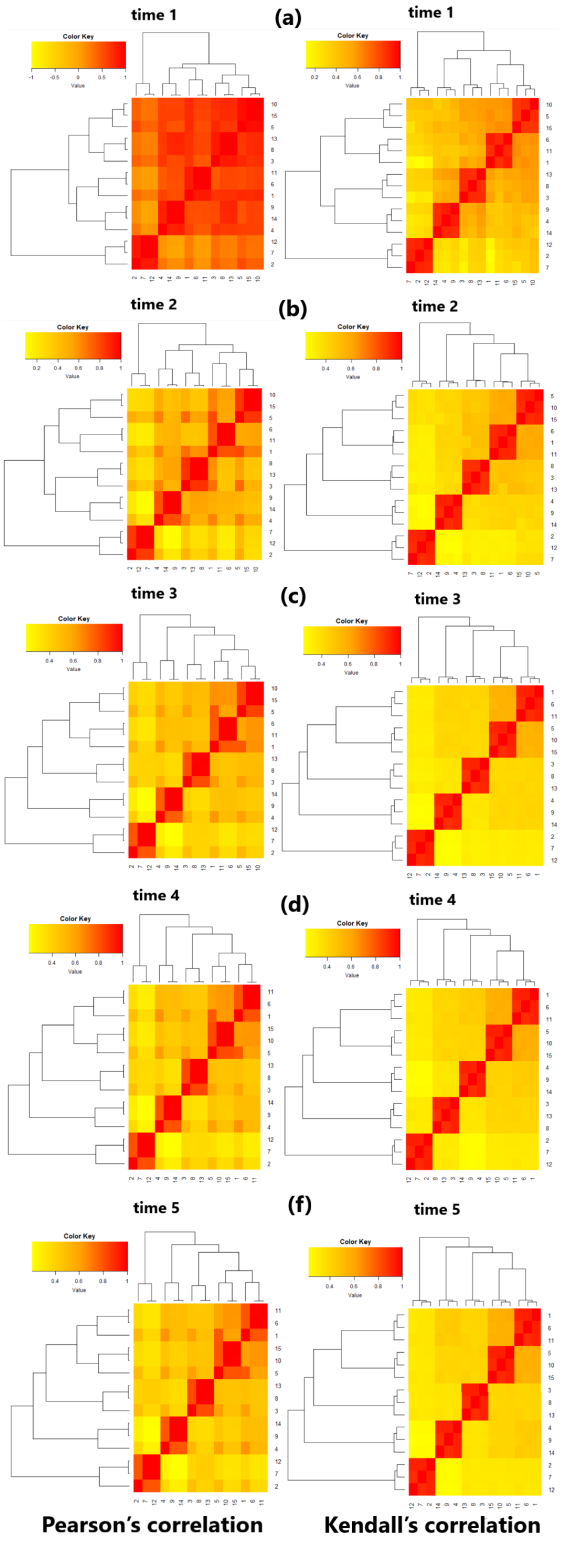

Figure 4: Heatmap plots: Left side) Pearson's correlation from time one to time five in simulated data, Right side) Kendall's association from time one to time five in simulated data

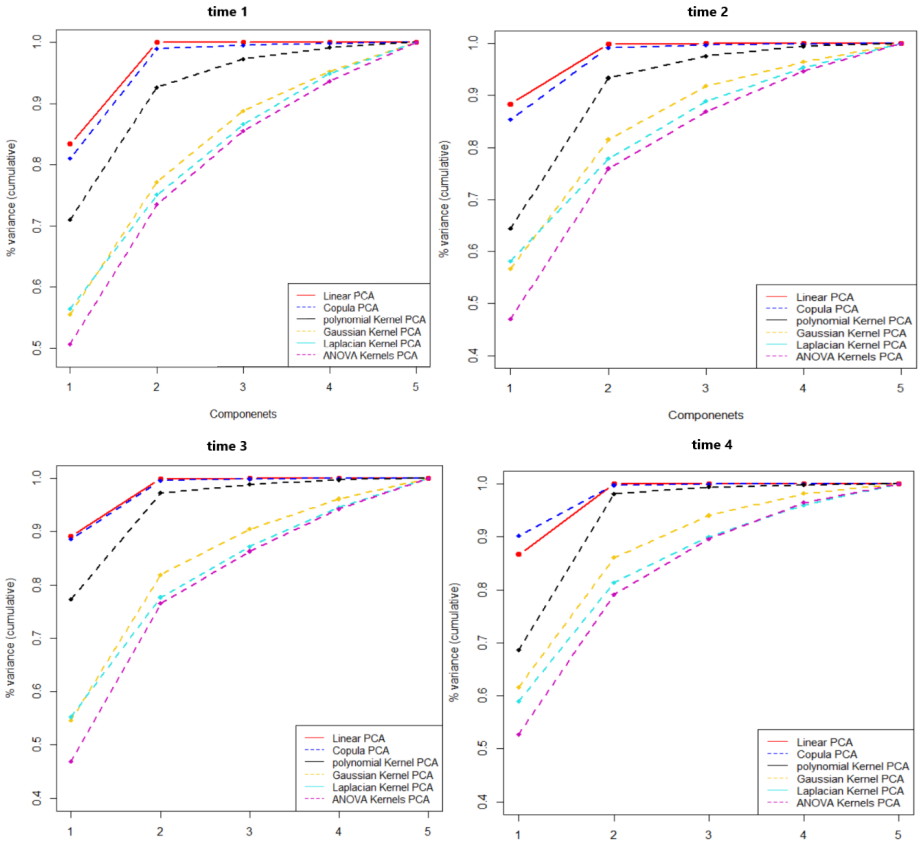

Figure 5: Percents of explained variance using linear *PCA*, Copula *PCA* and kernel *PCA* at different times in real-data (Apple stock)

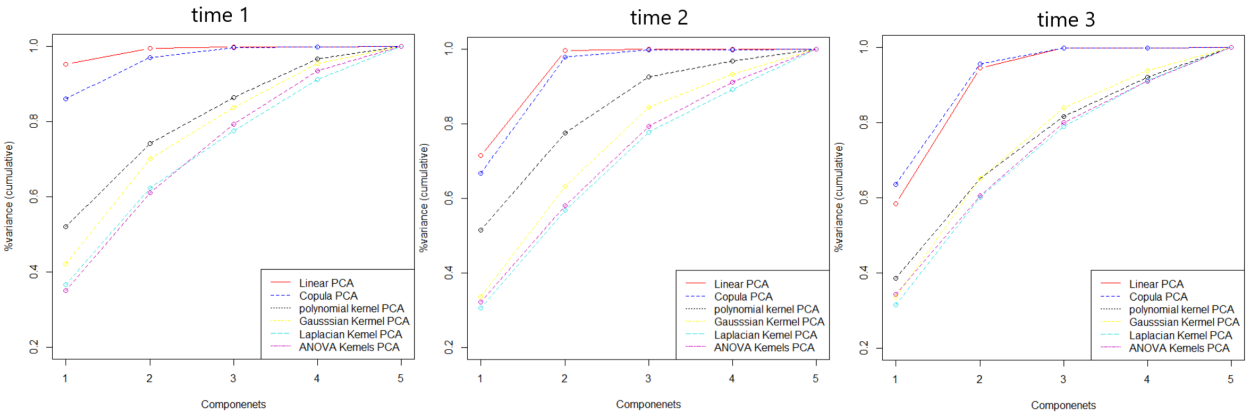

Figure 6: Percents of explained variance using linear *PCA*, Copula *PCA* and kernel *PCA* at different times in real-data (Bitcoin)