Chapter

Evaluation of Principal Component Analysis Variants to Assess Their Suitability for Mobile Malware Detection

Padmavathi Ganapathi, Shanmugapriya Dhathathri and Roshni Arumugam

Abstract

Principal component analysis (PCA) is an unsupervised machine learning algorithm that plays a vital role in reducing the dimensions of the data in building an appropriate machine learning model. It is a statistical process that transforms the data containing correlated features into a set of uncorrelated features with the help of orthogonal transformations. Unsupervised machine learning is a concept of self-learning method that involves unlabelled data to identify hidden patterns. PCA converts the data features from a high dimensional space into a low dimensional space. PCA also acts as a feature extraction method since it transforms the 'n' number of features into 'm' number of principal components (PCs; $m < n$). Mobile Malware is increasing tremendously in the digital era due to the growth of android mobile users and android applications. Some of the mobile malware are viruses, Trojan horses, worms, adware, spyware, ransomware, riskware, banking malware, SMS malware, keylogger, and many more. To automate the process of detecting mobile malware without human intervention, machine learning methods are applied to discover the malware more precisely. Specifically, unsupervised machine learning helps to uncover the hidden patterns to detect anomalies in the data. In discovering hidden patterns of malware, PCA is an important dimensionality reduction technique that can be applied to transform the features into PCs containing important feature values. So, by implementing PCA, the correlated features are transformed into uncorrelated features automatically to explore the anomalies in the data effectively. This book chapter explains all the variants of the PCA, including all linear and non-linear methods of PCA and their suitability in applying to mobile malware detection. A case study on mobile malware detection with variants of PCA using machine learning techniques in CICMalDroid_2020 dataset has been experimented. Based on the experimental results, for the given dataset, normal PCA is suitable to detect the malware data points and forms an optimal cluster.

Keywords: cyber security, dimensionality reduction, machine learning, mobile malware, principal component analysis, variants of PCA

1. Introduction

PCA is a statistical technique for compressing the content of large datasets into a smaller number of summary indices that can be examined and evaluated more quickly. Principal component analysis (PCA) is a multivariate statistical methodology that is frequently utilized nowadays [1]. It is a factor analysis-based statistical method that is widely used in the disciplines of pattern recognition and signal processing. It is a dimensionality reduction technique that condenses a large number of variables into a smaller set while maintaining the majority of the larger dataset. Because smaller datasets are easier to examine and visualize, machine learning algorithms can assess data more efficiently and rapidly without dealing with extra impediments.

PCA is also commonly employed in exploratory data analysis and prediction model construction. It is frequently used for dimensionality reduction, which involves projecting each data point onto only the first few principal components (PCs) in order to obtain lower-dimensional data with the least amount of variance. The first PC is a direction that lowers the predicted data variance. The i^{th} PC minimizes the variance of the projected data by being the inverse of the first $i - 1$ PC.

The primary components of the data covariance matrix can be proven to be eigenvectors. As a result, Eigen decomposition of the data covariance matrix or singular value decomposition of the data matrix is typically used to extract primary components. PCA, closely related to factor analysis, is the most fundamental of the real eigenvector-based multivariate techniques. On the other hand, factor analysis makes additional domain-specific assumptions about the underlying structure and solves matrix eigenvectors. Canonical correlation analysis (CCA) is also tied to PCA. PCA suggests a new orthogonal coordinate system for defining variance in a single dataset, whereas CCA proposes coordinate systems for describing cross-covariance across two datasets.

The purpose of the research is to explore and suggest a suitable type of PCA to reduce the data dimensions, which helps to identify the malware data points significantly.

2. Concept of PCA

PCA attempts to project high-dimensional data onto the feasible smallest-dimensional space. PCA takes into account the variance of each character because a high attribute indicates good class separation, and so minimizes dimensionality. Image processing, movie recommendation systems, and optimizing power distribution across numerous communication channels are some of PCA real-world use cases. It preserves the essential variables while rejecting the less important ones because it is a feature extraction approach [2].

The mathematical concepts employed in the PCA are:

- Variance and covariance
- Eigenvalues and eigenvectors

2.1 Common terms used in PCA

The following are the standard terms widely used in the PCA are discussed below:

Dimensionality: The dimensionality of a dataset refers to the number of characteristics or variables in it. It refers to the total number of columns in a dataset.

Correlation: It expresses the degree to which two variables are intertwined. For example, when one variable is changed, the other variable is also changed. The correlation value can be anywhere between −1 and +1. If the variables are inversely proportional to each other, the result is −1, and if they are directly proportional to each other, the result is +1.

Orthogonal: As a result, the variables have no relationship with one another, and their correlation is zero.

Eigen vectors: If Av is the scalar multiple of v and one has a non-zero vector v and a square matrix M, then v is an eigenvector.

Eigen values: An Eigenvalue is a number that indicates the variance in a specific direction.

Variance: A variance is used to calculate the fluctuation of data points dispersed over the multidimensional graph. In mathematics, it is the average squared deviation from the mean value. The following formula is used to determine $\mathrm{Var}(x)$:

$$\mathrm{Var}(x) = \frac{\sum \left(X_{i-}\overline{X}\right)^2}{N} \tag{1}$$

Covariance: Covariance can determine the degree to which comparable components from two sets of grouped data move in the same direction. It is used to uncover relationships and correlations between dataset attributes in layman's terms. The following is the formula for calculating the Cov (x, y):

$$\mathrm{Cov}\ (x) = \frac{\sum \left(X_i - \overline{X}\right)\ \left(Y_i - \overline{Y}\right)}{N} \tag{2}$$

Covariance matrix: The covariance matrix shows how two variables are related.

Principal components: The new set of data variables created from the original dataset is referred to as PCs. The newly created data variables are pretty valuable and self-contained. They have access to all of the essential data from the original variables.

3. Principal components algorithm

As previously said, the PCs are the converted new characteristics or the result of PCA. The total number of PCs in the dataset is equal or fewer than the total number of original features. Some of the features of these primary components are as follows:

- The significant component must be a linear combination of the original features, and these components must be orthogonal, meaning there must be no link between the two variables.

- As the number of components increases from 1 to n, their relevance decreases, showing that the PC1 is the most essential and the PCn is the least important.

3.1 Steps involved in the PCA algorithm

To carry out the process of PCA the following are the five significant steps to be followed [3]:

I. Standardization

II. Covariance matrix computation

III. Computation of eigenvectors and eigenvalues of the covariance matrix to identify the PCs

IV. Feature vector creation

V. Recast the data along the axes of the PC

3.1.1 Standardization

This step normalizes a set of continuous beginning variables so that their effects on the analysis are consistent.

Standardization is essential before PCA since it is sensitive to the variances of the original variables. Suppose the initial variable ranges differ significantly. In that case, the variables with more comprehensive ranges will outnumber those with smaller ranges (for example, a variable ranging from 0 to 100 will outnumber a variable ranging from 0 to 1), resulting in a skewed outcome. As a result, converting the data to equal scales could be a possible solution to this issue. Subtracting the mean and dividing by the standard deviation for each variable value can be done numerically.

$$Z = \frac{\text{Value} - \text{mean}}{\text{Standard deviation}} \tag{3}$$

After the standardization is complete, all variables will be changed to the same scale.

3.1.2 Covariance matrix computation

The goal of this step is to determine how the variables in the input dataset differ from the mean with each other and whether there is a link between them. Because the variables may become so intertwined that they contain redundant information, the covariance matrix is constructed to find these correlations.

The covariance matrix, a symmetric matrix with $\boldsymbol{p} \times \boldsymbol{p}$ entries, contains all possible pairs of starting variables and their covariances (where p is the number of dimensions). The covariance matrix for a three-dimensional dataset with three variables x, y, and z is a 3×3 matrix of the form:

$$\begin{matrix} \mathbf{Cov}(\boldsymbol{x},\boldsymbol{x}) & \mathbf{Cov}(\boldsymbol{x},\boldsymbol{y}) & \mathbf{Cov}(\boldsymbol{x},\boldsymbol{z}) \\ \mathbf{Cov}(\boldsymbol{y},\boldsymbol{x}) & \mathbf{Cov}(\boldsymbol{y},\boldsymbol{y}) & \mathbf{Cov}(\boldsymbol{y},\boldsymbol{z}) \\ \mathbf{Cov}(\boldsymbol{z},\boldsymbol{x}) & \mathbf{Cov}(\boldsymbol{z},\boldsymbol{y}) & \mathbf{Cov}(\boldsymbol{z},\boldsymbol{z}) \end{matrix} \tag{4}$$

The variances of each starting variable are shown on the main diagonal (top left to bottom right) since a variable covariance with itself equals its variance (Cov(a, a) = Var(a)). The entries of the covariance matrix are symmetric about the principal diagonal because covariance is commutative (Cov(a,b) = Cov(b,a)). This shows that the triangle's upper and lower triangular parts are equal.

The following are the signs of covariance that are related to correlation:

- If the sign of covariance is positive, the two variables will rise or fall in lockstep (i.e., correlated with each other)
- When the sign of covariance is negative, one variable rises while the other falls (i.e., inversely correlated)

3.1.3 Computation of eigenvectors and eigenvalues of the covariance matrix to identify the principal components

In order to uncover the underlying components of the data, eigenvectors and eigenvalues are linear algebra concepts that must be computed from the covariance matrix. Before go into the details of these themes, let us establish what "principal components" mean.

PCs are new variables created by merging or linearly combining essential variables. The new variables (i.e., primary components) are uncorrelated due to these combinations, and the majority of the information from the initial variables is squeezed or compressed into the first components. So, 10-dimensional data provides ten primary components. However, PCA seeks to place as much information as possible in the first component, then as little information in the second, and so on, until the result looks like **Figure 1** below:

One can minimize dimensionality without granting too much information by splitting data into critical components and eliminating components with insufficient data. The remaining components can be regarded as new variables. Because the essential components are produced as linear combinations of the original variables, they are less interpretable and have no significant relevance.

The data directions that explain the most variance, or the lines that include the most data information, are considered essential components in geometric terms. In

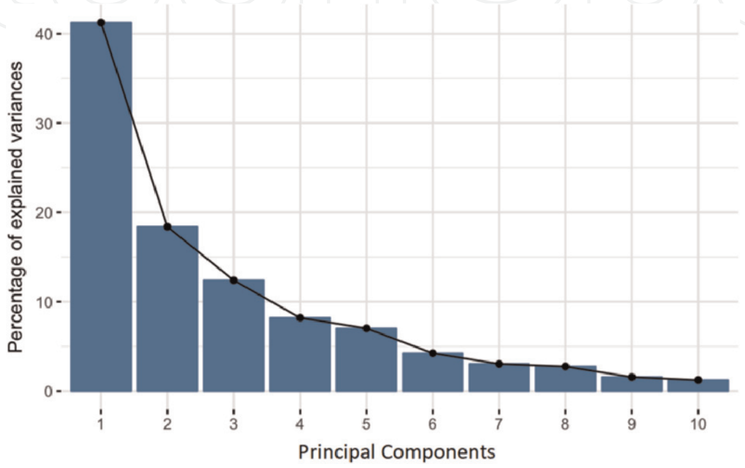


Figure 1.
Principal components vs. percentage of explained variances.

this case, the higher a line variance, the greater the dispersion of data points, and the greater the dispersion along a line, the more information it retains. Put another way; consider the essential components as additional dimensions that provide the proper viewpoint for perceiving and processing data, making it easier to spot differences between observations.

3.1.3.1 Constructing principal components with PCA

Because there are as many variables in the data as there are PCs, the first PC is designed to provide the possible variance in the dataset.

The second major component is determined in the same fashion as the first, except it must be uncorrelated (i.e., parallel to) and account for the following most significant variance. This technique is repeated until the number of variables equals the number of essential components.

3.1.3.2 Finding Eigen values and Eigen vectors

After one has determined the essential components, let us discuss eigenvalues and eigenvectors. Remember that eigenvalues and eigenvectors are always obtained in pairs, with one eigenvalue per eigenvector. In addition, the number is the same as the number of data dimensions. There are three variables in a three-dimensional dataset. Hence there are three eigenvectors with three corresponding eigenvalues. PCs are the eigenvectors of the covariance matrix, and they are the directions of the axis with the most variation. Eigenvalues are the coefficients associated with eigenvectors, whereas eigenvalues are the coefficients attached to eigenvectors. The significant components are ordered in order of significance by arranging the eigenvectors in order of their eigenvalues, from highest to lowest.

Assume the dataset is two-dimensional, with two variables x and y, and the covariance matrix eigenvectors and eigenvalues are:

$$\boldsymbol{v}\mathbf{1} = \frac{\mathbf{0.6778736}}{\mathbf{0.7351785}}\lambda_1 = \mathbf{1.284028} \tag{5}$$

$$\boldsymbol{v}\mathbf{2} = \frac{-\mathbf{0.7351785}}{\mathbf{0.6778736}}\lambda_2 = \mathbf{0.04908323} \tag{6}$$

The outcome of sorting the eigenvalues in ascending order is $\mathbf{>\lambda 2}$, suggesting that the eigenvector of the first PC is v1 and the eigenvector of the second PC is v2. To find the proportion of variance (information) that each component accounts for, divide each component eigenvalue by the total eigenvalues. In the scenario mentioned above, PC1 and PC2 are responsible for 96 and 4% of the data fluctuation.

3.1.4 Feature vector creation

The key components can be identified in order of importance by computing the eigenvectors and sorting them by their eigenvalues in decreasing order. One must decide whether to preserve all of these components or reject those with low eigenvalues and then use the remaining ones to construct the feature vector–matrix at this phase. As a result, the feature vector is just a matrix with the appropriate components eigenvectors as columns. Because only p eigenvectors (components) are left out of n, the final dataset will only have p dimensions.

Combining both eigenvectors v1 and v2 creates a feature vector, as seen in the example above:

$$\begin{bmatrix} 0.6778736 & -0.7351785 \\ 0.7351785 & 0.6778736 \end{bmatrix} \tag{7}$$

Alternatively, one can omit the less relevant eigenvector v2 and solely utilize v1 to generate a feature vector:

$$\begin{bmatrix} 0.6778736 \\ 0.7351785 \end{bmatrix} \tag{8}$$

By eliminating the eigenvector v2, the final dataset dimensionality will be reduced by one, resulting in a loss of information. The loss will be minimal because v2 only carried 4% of the data, and v1 will keep 96% of the data.

The individual must decide whether to maintain all components or delete those not as significant, like in the previous scenario. Because leaving out less significant components is unnecessary if all one wants to do is explain the data in terms of new uncorrelated variables (PCs) without attempting to reduce dimensionality.

3.1.5 Recast the data along the axes of the principal component

The data from previous phases is unchanged except for standardization; all required is to select the primary components and generate the feature vector; however, the input dataset is always in terms of the original axes (i.e., in terms of the initial variables). The third phase, PCA, shifts data from the original axis to the ones indicated by the significant components using a feature vector constructed from the covariance matrix eigenvectors. This is done by multiplying the original dataset transpose by the feature vector transpose. Therefore,

$$\text{Final dataset} = \text{feature vector}^{T}\text{standardized original dataset}^{T} \tag{9}$$

4. When to apply PCA?

PCA is widely applicable for unsupervised machine learning techniques, which helps to reduce the dimensions of the large data where the dataset does not contain the labelled column. The following are some of the situations where the PCA can be applied [4]:

Case 1: One wants to limit the number of variables but cannot figure out which ones to leave out entirely?
Case 2: One wants to make sure variables are unrelated to one another?
Case 3: One thought, if the independent variables are less interpretable?

4.1 Properties of principal components

Suppose the number of primary variables that make up the PCs is less than or equal to the number of variables or data points. In that case, the PCA is complete. The following are the characteristics of primary components [5]:

- They are a set of primary data variables projected in various directions and have qualities similar to those of the original variables.
- In machine learning and data science, dimensionality reduction is a common technique.
- They are orthogonal.
- As one finds PC one by one, the variance or variation of the PCs reduces. This means the first PC is the most volatile, while the last PC is the least volatile.

4.2 Applications of PCA

PCA has a wide range of applications, including the following:

- Face recognition
- Computer vision
- Image compression
- Bioinformatics
- Hidden pattern recognition
- Exploratory data analysis
- Noise filtering
- Finance, data mining, psychology, etc.

4.3 Pros and cons of principal component analysis

For any technique, there will be both positive and negative phases. Likewise, in PCA, its advantages and limitations are the following [6].

4.3.1 Advantages of principal component analysis

Some of the advantages of PCA are:

- Easy to compute
- Speeds up the performance of machine learning algorithms

- Counteracts the issues of high-dimensional data
- Remove correlated features
- Improves the accuracy of the algorithm
- Reduces overfitting
- Enhance visualization

4.3.2 Limitations of principal component analysis

Some of the limitations of PCA are:

- Independent variables become less interpretable
- Data standardization must perform before PCA
- The trade-off between information loss and dimensionality reduction
- Difficult to evaluate the covariance in an appropriate way
- It is sensitive to scale the features
- PCA is not robust against outliers
- PCA assumes a linear relationship between features

5. Variants of PCA

To overcome the limitations of PCA, there are different types of PCA are available that suit for the appropriate type of data are listed below [7, 8]:

- Normal PCA
- Sparse PCA
- Randomized PCA
- Incremental PCA
- Kernel PCA

5.1 Normal PCA

PCA in Machine learning is applied for unsupervised learning to reduce the dimension of the data from high dimensional space to low dimensional space. The above section discusses the standard normal PCA, which applies to most of the datasets as a default form of PCA using unsupervised learning. To construct any type of PCA especially for normal PCA the above discussed five significant steps are involved for dimensionality reduction [9, 10]. The following sessions briefly discuss the other variants of PCA in machine learning and its characteristics.

5.2 Sparse PCA

One of PCA significant flaws is that the PCs are dense in most circumstances, implying that the majority of the loadings are non-zero. The model is difficult to interpret since each significant component is a linear combination of all the original variables. However, each axis may correspond to a specific gene in machine learning tasks such as gene analytics. In such instances, one can readily analyse the model and comprehend the physical meaning of the loading and the PCs if the majority of the entries in the loadings are zeros. Sparse PCA is a variant of PCA that uses sparse loading to build interpretable models. In Sparse PCA, each PC is a linear combination of a subset of the original variables.

5.3 Randomized PCA

The PCs are estimated using the low-rank matrix approximation in traditional PCA. However, this strategy becomes costly with large datasets and makes the entire process challenging to scale. One can approximate the first K PCs faster than traditional PCA by randomizing how the dataset singular value decomposition occurs.

5.4 Incremental PCA

The above-described PCA variants need the entire training dataset to be stored in memory. Incremental PCA can be employed when the dataset is too huge to fit in memory. It divides the dataset into mini-batches, each of which can fit into memory, and then feed each mini-batch to the incremental PCA algorithm one at a time.

5.5 Kernel PCA

A typical linear technique is PCA. It works well with linearly separable datasets. However, if the dataset contains non-linear relationships, the results will be unfavourable. Kernel PCA is a technique that uses the “kernel trick” to project linearly inseparable data into a higher dimension where it may be separated linearly. Many different kernels are commonly employed, including linear, polynomial, RBF, and sigmoid.

6. Case study on variants of PCA in mobile malware detection

To explore the dimensionality reduction using normal PCA and its variants for mobile malware detection in the CICMalDroid_2020 dataset [11] are experimented. The dataset is taken from the University of New Brunswick (UNB), Canadian Institute of Cyber Security (CIC). The dataset consists of 11,598 records with 471 feature attributes. The dataset commences with 17,341 Android samples gathered from VirusTotal, the Contagio security blog, AMD, MalDozer, and other sources. Samples were obtained between December 2017 and December 2018. Detecting Android apps with malicious data points is crucial for cyber security specialists. There are five key categories in the dataset includes, Adware, Banking malware, SMS malware, Riskware, and Benign are the different forms of malicious software. The experiment is carried out in a Python Jupyter notebook environment using sklearn library [12–16].

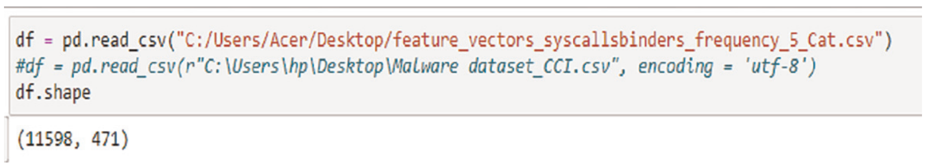
6.1 Results of PCA in machine learning for mobile malware detection

The following are the outcomes of normal PCA and variants of PCA where the 471 feature dimensions are reduced into two PCs are visualized below. **Figure 2** shows the importing data into the Python Jupyter notebook environment.

Figure 3 shows the data pre-processing to check whether the data contains any null values or not. Data pre-processing is an important step in the machine learning process, and it helps to purify the irrelevant and undefined raw data into the relevant defined form.

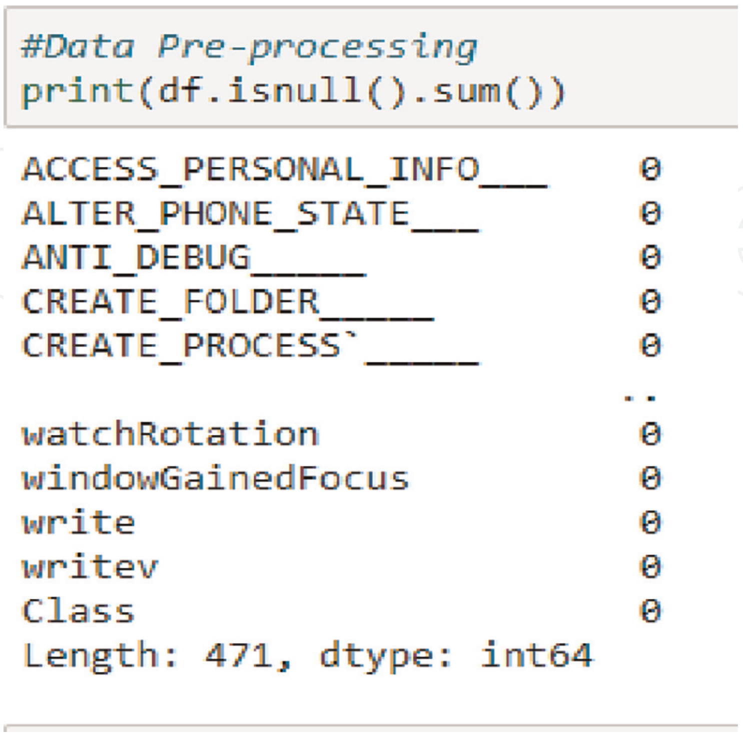
Figure 3 depicts that the dataset does not contain any null values, and it is fit for further processing (i.e.) from the results value '0'indicates there are no null values in the data.

Figure 4 shows the removal of duplicate data values to ensure the originality of the dataset. Duplicate data leads to misinterpretation of the results.



```
df = pd.read_csv("C:/Users/Acer/Desktop/feature_vectors_syscallsbinders_frequency_5_Cat.csv")
#df = pd.read_csv(r"C:\Users\hp\Desktop\Malware dataset_CCI.csv", encoding = 'utf-8')
df.shape

(11598, 471)
```

Figure 2.
Data import—CICMalDroid_2020 dataset.



```
#Data Pre-processing
print(df.isnull().sum())

ACCESS_PERSONAL_INFO___     0
ALTER_PHONE_STATE___        0
ANTI_DEBUG_____             0
CREATE_FOLDER_____          0
CREATE_PROCESS`_____        0
                           ..
watchRotation               0
windowGainedFocus           0
write                       0
writev                      0
Class                       0
Length: 471, dtype: int64
```

Figure 3.
Check if the data contains any null values or not.

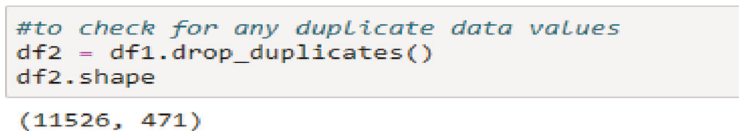
Figure 4 depicts that out of 11,598 records, 72 were duplicated, and the duplicated records were dropped. After dropping the 72 duplicate records, now the dataset consists of 11,526 instances with 471 features.

Figure 5 shows the data splitting for training and testing so that the machine learning model can detect and cluster the mobile malware data points in the dataset. Splitting the data for training and testing is a significant phase in the machine learning process. So, the data will be adequately trained and provide the best results in testing, which helps to derive a high efficacy rate.

Figure 5 explains that the dataset is divided into 70% for training and 30% for testing (i.e.) out of 11,526 records, 8068 are used for training and 3458 samples are used for testing the model. Now, the dataset is suitable to perform the PCA with the machine learning technique.

Figure 6 shows the feature scaling; before applying PCA, one must scale the data so that the data can be properly scaled within a particular range appropriately to support data modelling. Without incorporating feature scaling, during the model development the data takes more time to fit into the prescribed model form.

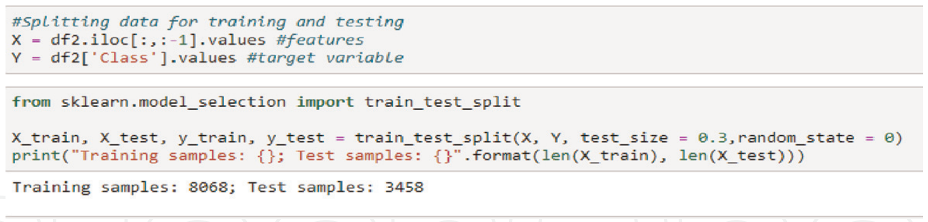
Figure 6, depicts the method for feature scaling using MinMaxScaler and standard scalar to bring the scattered data points within a typical specified range. Hence, the data is further applicable for PCA.



```
#to check for any duplicate data values
df2 = df1.drop_duplicates()
df2.shape
```

```
(11526, 471)
```

Figure 4.
Drop duplicate values.

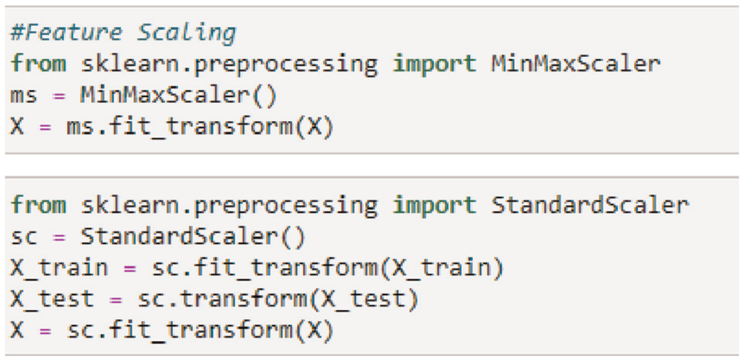


```
#Splitting data for training and testing
X = df2.iloc[:,:-1].values #features
Y = df2['Class'].values #target variable
```

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3,random_state = 0)
print("Training samples: {}; Test samples: {}".format(len(X_train), len(X_test)))
```

```
Training samples: 8068; Test samples: 3458
```

Figure 5.
Train and test split.



```
#Feature Scaling
from sklearn.preprocessing import MinMaxScaler
ms = MinMaxScaler()
X = ms.fit_transform(X)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X = sc.fit_transform(X)
```

Figure 6.
Feature scaling.

Figure 7 shows the normal PCA method used in the dataset to reduce the 471 featured dimensions into two PCs. It also shows that from the explained variance PC1 has more information than PC2. Normal PCA is the default form of PCA, more suitable for all kinds of data to reduce the dimensions effectively without any information loss.

Figure 8 shows the normal PCA results of two PCs as PC1 and PC2.

Figure 8 represents that the total 471 features are reduced into two PCs PC1 and PC2, without removing any of the data features. This helps to train the data and develop the machine learning model effectively with less memory consumption.

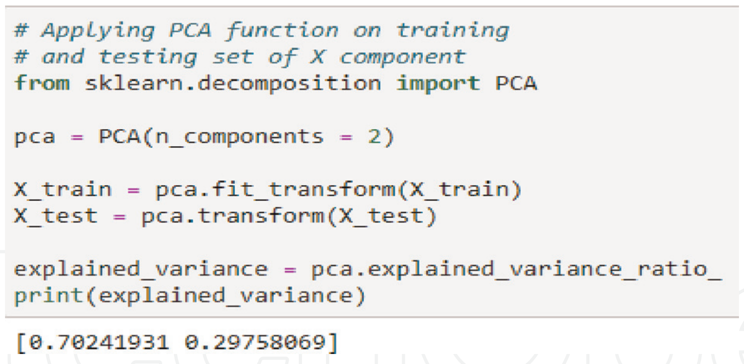
Figure 9 shows the method of sparse PCA in mobile malware data. Similarly, the sparse PCA is widely suited for sparse data so that the 471 data features are reduced into two set of PCs PC1 and PC2.

Figure 10 shows the method of randomized PCA in mobile malware data. Randomized PCA is suitable for big data processing so that the features are randomly selected to derive the two set of PCs PC1 and PC2.

Figure 11 shows the method of incremental PCA in mobile malware data. Incremental PCA is similar to randomized PCA, but it gradually increases the batch size to reduce the total number of features into two set of PCs PC1 and PC2.

Figure 12 shows the method of kernel PCA in mobile malware data. Kernel PCA is widely applicable for non-linear data modelling. It also helps to reduce the dimensions of the data based on the kernel function like gamma etc. to derive the two or more sets of PCs PC1and PC2.

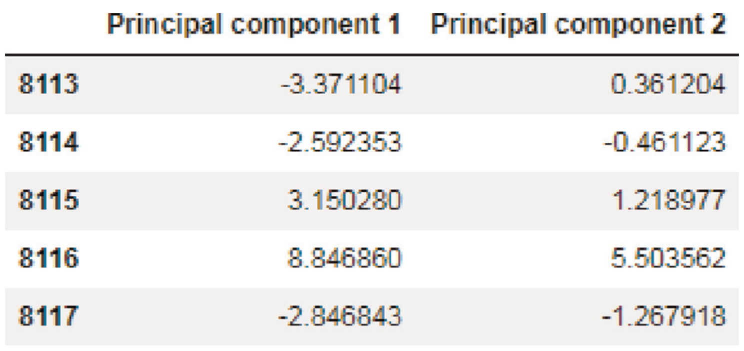
In this case study, the 471 data features of CICMalDroid_2020 dataset is transformed into two set of PCs PC1 and PC2 for all variants of PCA, depending upon the suitability of the data values.



```
# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)

X_train = pca.fit_transform(X_train)
X_test = pca.transform(X_test)

explained_variance = pca.explained_variance_ratio_
print(explained_variance)
```

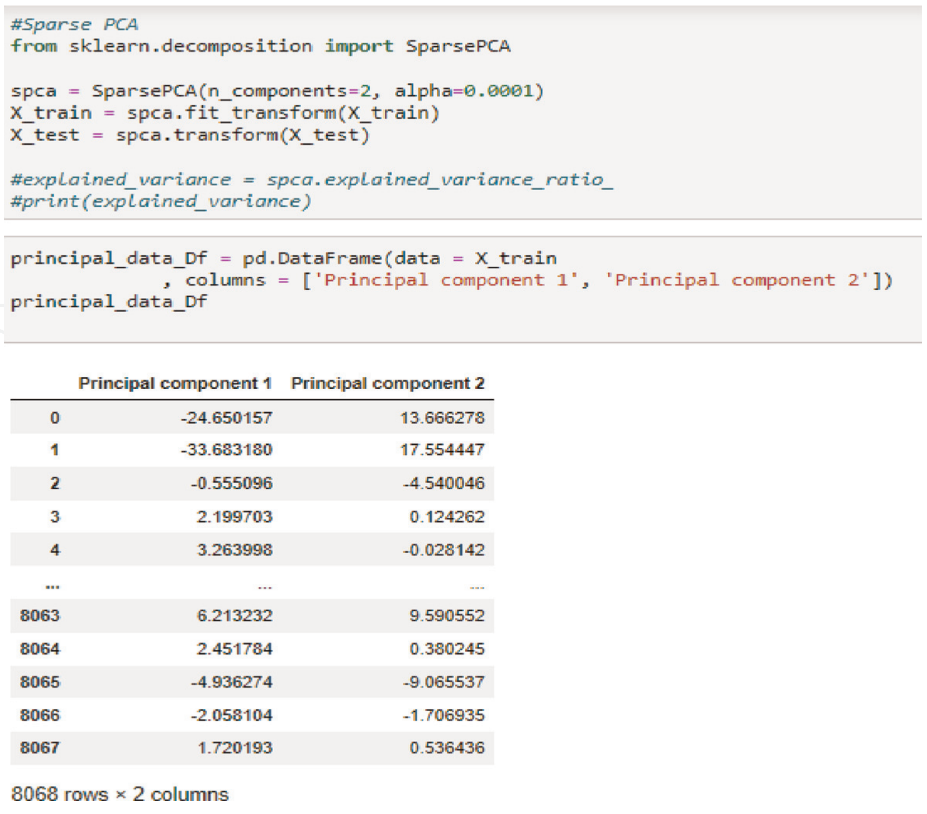
```
[0.70241931 0.29758069]
```

Figure 7.
Normal PCA.



	Principal component 1	Principal component 2
8113	-3.371104	0.361204
8114	-2.592353	-0.461123
8115	3.150280	1.218977
8116	8.846860	5.503562
8117	-2.846843	-1.267918

Figure 8.
Normal PCA with PC1 and PC2.



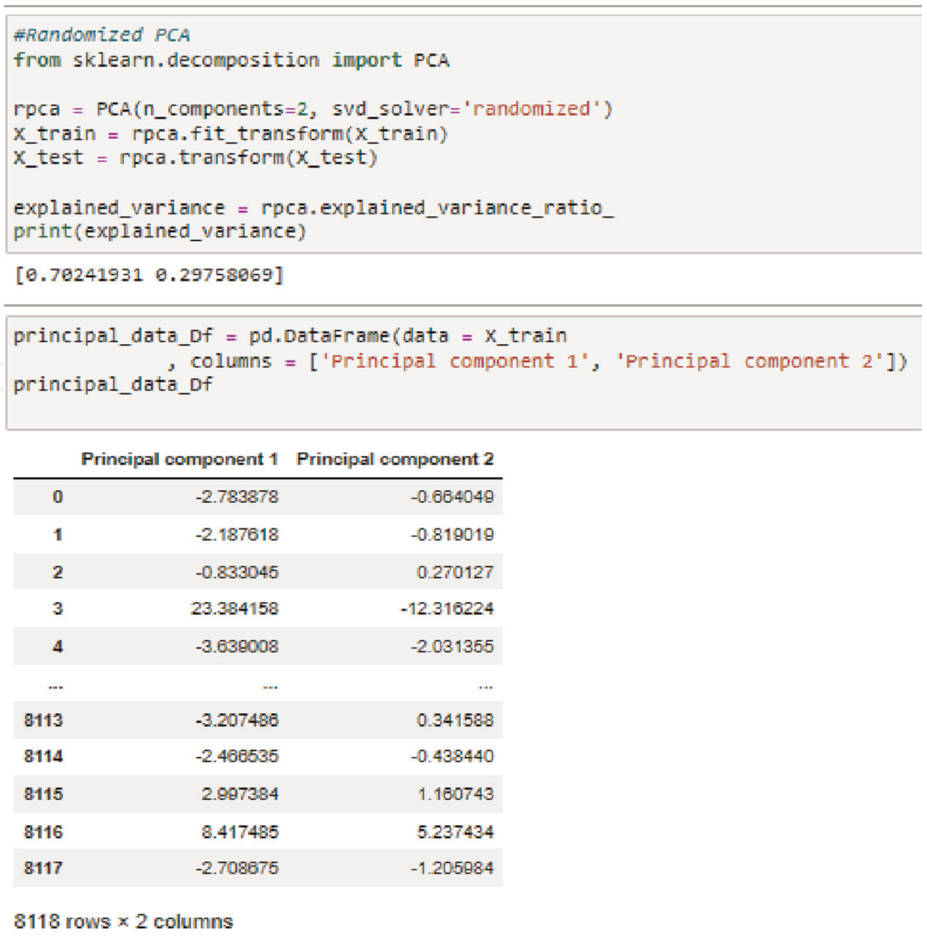
```
#Sparse PCA
from sklearn.decomposition import SparsePCA

spca = SparsePCA(n_components=2, alpha=0.0001)
X_train = spca.fit_transform(X_train)
X_test = spca.transform(X_test)

#explained_variance = spca.explained_variance_ratio_
#print(explained_variance)
```

```
principal_data_Df = pd.DataFrame(data = X_train
             , columns = ['Principal component 1', 'Principal component 2'])
principal_data_Df
```

	Principal component 1	Principal component 2
0	-24.650157	13.666278
1	-33.683180	17.554447
2	-0.555096	-4.540046
3	2.199703	0.124262
4	3.263998	-0.028142
...	...	...
8063	6.213232	9.590552
8064	2.451784	0.380245
8065	-4.936274	-9.065537
8066	-2.058104	-1.706935
8067	1.720193	0.536436

8068 rows × 2 columns

Figure 9.
Sparse PCA.



```
#Randomized PCA
from sklearn.decomposition import PCA

rpca = PCA(n_components=2, svd_solver='randomized')
X_train = rpca.fit_transform(X_train)
X_test = rpca.transform(X_test)

explained_variance = rpca.explained_variance_ratio_
print(explained_variance)
```

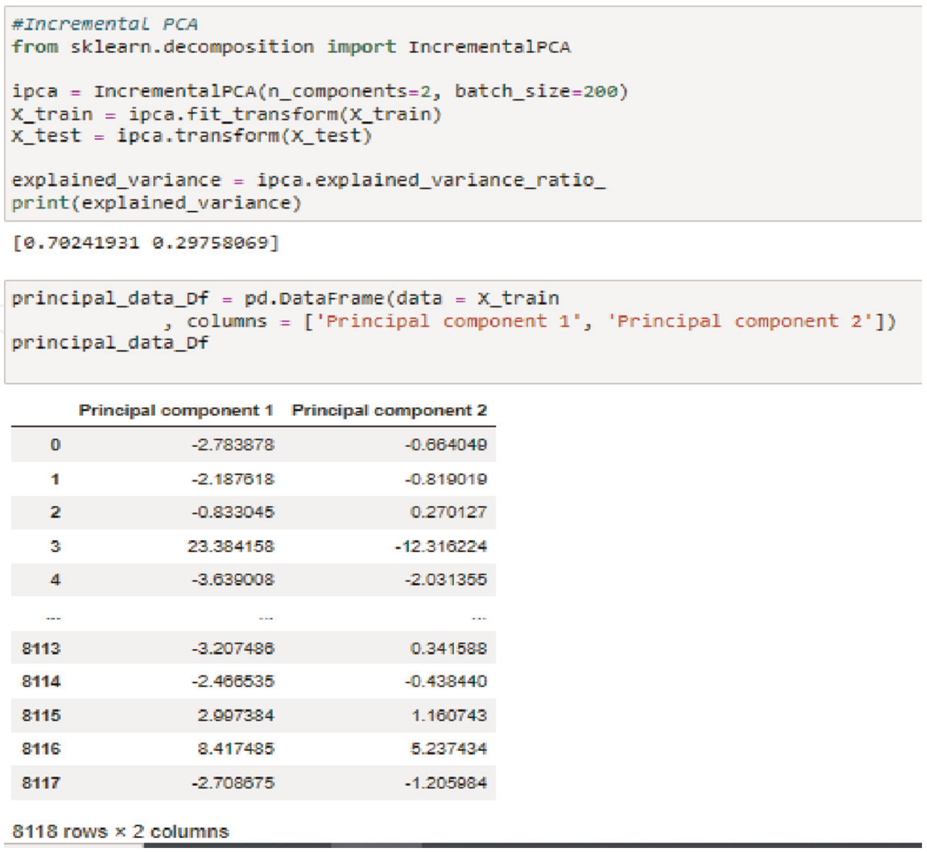
```
[0.70241931 0.29758069]
```

```
principal_data_Df = pd.DataFrame(data = X_train
             , columns = ['Principal component 1', 'Principal component 2'])
principal_data_Df
```

	Principal component 1	Principal component 2
0	-2.783878	-0.664049
1	-2.187618	-0.819019
2	-0.833045	0.270127
3	23.384158	-12.316224
4	-3.639008	-2.031355
...	...	...
8113	-3.207486	0.341588
8114	-2.466535	-0.438440
8115	2.997384	1.160743
8116	8.417485	5.237434
8117	-2.708675	-1.205984

8118 rows × 2 columns

Figure 10.
Randomized PCA.



```
#Incremental PCA
from sklearn.decomposition import IncrementalPCA

ipca = IncrementalPCA(n_components=2, batch_size=200)
X_train = ipca.fit_transform(X_train)
X_test = ipca.transform(X_test)

explained_variance = ipca.explained_variance_ratio_
print(explained_variance)
```

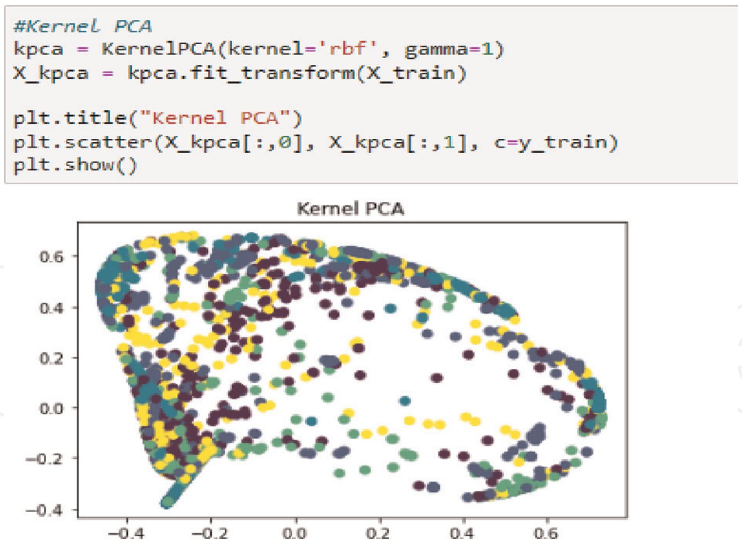
```
[0.70241931 0.29758069]
```

```
principal_data_Df = pd.DataFrame(data = X_train
            , columns = ['Principal component 1', 'Principal component 2'])
principal_data_Df
```

	Principal component 1	Principal component 2
0	-2.783878	-0.664049
1	-2.187618	-0.819019
2	-0.833045	0.270127
3	23.384158	-12.316224
4	-3.639008	-2.031355
...	...	...
8113	-3.207486	0.341588
8114	-2.466535	-0.438440
8115	2.997384	1.160743
8116	8.417485	5.237434
8117	-2.708675	-1.205984

8118 rows × 2 columns

Figure 11.
Incremental PCA.



```
#Kernel PCA
kpca = KernelPCA(kernel='rbf', gamma=1)
X_kpca = kpca.fit_transform(X_train)

plt.title("Kernel PCA")
plt.scatter(X_kpca[:,0], X_kpca[:,1], c=y_train)
plt.show()
```

Figure 12.
Kernel PCA.

6.2 Observations

Based on the results obtained from the variants of PCA for mobile malware detection depicts that for the given CICMalDroid_2020 dataset is discussed. It is a numerical labelled data that highly supports normal standard PCA technique. It helps to reduce the dimensions of the PCA from 471 features into two sets of PCs (PC1 and PC2). It supports to processing the data model quickly and forms the clusters

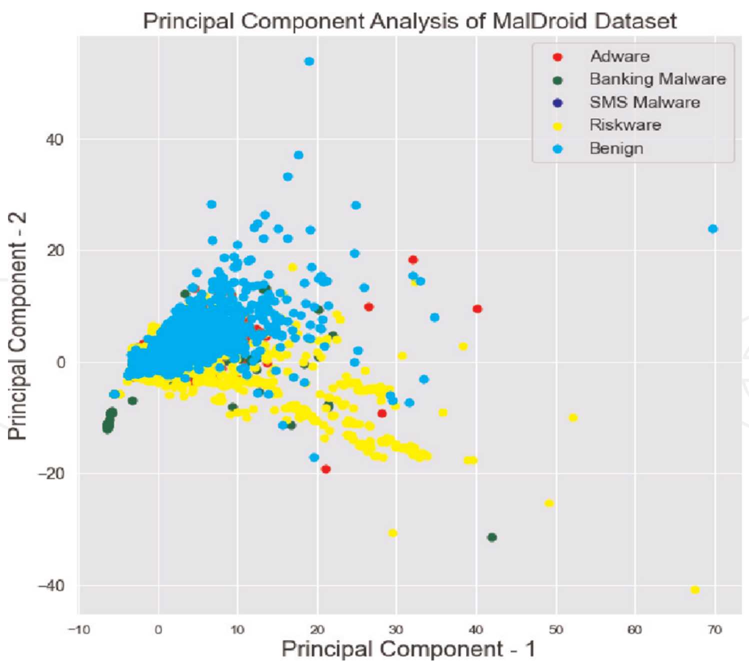


Figure 13.
Class of malwares in the CICMalDroid_2020 dataset.

Class	Type of malware	Size
1	Adware	1253
2	Banking malware	2100
3	SMS malware	3904
4	Riskware	2546
5	Benign	1795

Table 1.
CICMalDroid_2020 dataset—sample size

effectively. **Figure 13** shows the group of clusters that discovered the five different malware involved in the CICMalDroid_2020 dataset based on PC1containing huge information about the dataset of normal PCA.

Table 1 shows the size of malware samples available under the category of Adware, Banking malware, SMS malware, Riskware, and Benign.

Hence, the other variants of PCA, such as sparse PCA are applicable for sparse data, randomized PCA and incremental PCA are suitable for big data processing and kernel PCA is widely supported by non-linear data modelling. So, depending upon the type of data and their accessibility, the appropriate type of PCA is incorporated into machine learning algorithms specifically for unsupervised learning for dimensionality reduction to develop a suitable predictive model. It also helps to identify the hidden patterns of the data effectively.

7. Conclusion

This book chapter on “Evaluation of PCA Variants to assess their suitability for Mobile Malware Detection” describes briefly the concept of PCA, the common terminologies involved in PCA, PCs in PCA, mathematical properties of PCA, steps involved in PCA algorithm, explanation about the process of developing the PCA model in a machine learning perspective, applications of PCA, advantages and limitations of PCA. Different variants of PCA are experimented with a small case study by exploring the various type of PCA. This helps to find out the suitable variant of PCA for mobile malware detection in the CICMalDroid_2020 dataset based on the machine learning framework. As an outcome, for the given dataset, the normal standard PCA provides the appropriate results for the PCs to discover the malware data points accurately. Thus, this chapter will be a ready reckoner for the learners to know about the concept of PCA in machine learning, and its variants suitable for mobile malware detection are discussed in detail.

Acknowledgements

This work is supported by the DST-CURIE-AI project during 2021–2023 by the Centre for Cyber Intelligence established under the Centre for Machine Learning and Intelligence, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore-43, Tamilnadu, India.

Conflict of interest

The authors declare no conflict of interest.

Nomenclature

PCA	Principal component analysis
CCA	Canonical correlation analysis
CIC	Canadian Institute of Cyber Security
Var	Variance
Cov	Covariance
UNB	University of New Brunswick
PC	Principal components
X, Y, Z	Variables
RBF	Radial basis function
n	Number of features
m	Number of PCs
v	Eigenvector
λ	Eigenvalue

Author details

Padmavathi Ganapathi[1]*, Shanmugapriya Dhathathri[2] and Roshni Arumugam[3]

1 Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamilnadu, India

2 Department of Information Technology, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamilnadu, India

3 Centre for Cyber Intelligence, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamilnadu, India

*Address all correspondence to: padmavathi_cs@avinuty.ac.in

References

[1] Available from: https://www.sartorius.com/en/knowledge/science-snippets/what-is-principal-component-analysis-pca-and-how-it-is-used-507186. [Accessed: 26 April 2022]

[2] Available from: https://en.wikipedia.org/wiki/Principal_component_analysis. [Accessed: 26 April 2022]

[3] Available from: https://builtin.com/data-science/step-step-explanation-principal-component-analysis. [Accessed: 26 April 2022]

[4] Available from: https://www.javatpoint.com/principal-component-analysis [Accessed: 06 March 2022]

[5] Available from: https://intellipaat.com/blog/a-brief-introduction-to-principal-component-analysis/. [Accessed: 26 April 2022]

[6] Available from: https://www.i2tutorials.com/what-are-the-pros-and-cons-of-the-pca/. [Accessed: 27 April 2022]

[7] Available from: https://www.keboola.com/blog/pca-machine-learning. [Accessed: 27 April 2022]

[8] Available from: https://aiaspirant.com/types-of-pca/. [Accessed: 27 April 2022]

[9] Available from: https://www.analyticssteps.com/blogs/introduction-principal-component-analysis-machine-learning. [Accessed: 27 April 2022]

[10] Available from: https://intellipaat.com/blog/a-brief-introduction-to-principal-component-analysis/. [Accessed: 27 April 2022]

[11] Available from: https://www.unb.ca/cic/datasets/maldroid-2020.html. [Accessed: 06 March 2022]

[12] Benahmed L, Houichi L. The effect of simple imputations based on four variants of PCA methods on the quantiles of annual rainfall data. Environmental Monitoring and Assessment. 2018;**190**(10):1-14. DOI: 10.1007/s10661-018-6913-y

[13] Wang Z, Han D, Li M, Liu H, Cui M. The abnormal traffic detection scheme based on PCA and SSH. Connection Science. 2022;**34**(1):1201-1220. DOI: 10.1080/09540091.2022.2051434

[14] Manzano C, Meneses C, Leger P, Fukuda H. An empirical evaluation of supervised learning methods for network malware identification based on feature selection. Complexity. 2022; **2022**:1-18. DOI: 10.1155/2022/6760920

[15] Aurangzeb S, Anwar H, Naeem MA, Aleem M. BigRC-EML: Big-data based ransomware classification using ensemble machine learning. Cluster Computing. 2022. DOI: 10.1007/s10586-022-03569-4

[16] Rajadurai H, Gandhi UD. An empirical model in intrusion detection systems using principal component analysis and deep learning models. Computational Intelligence. 2020;**37**(3): 1111-1124. DOI: 10.1111/coin.12342