Study of Dimensionality Reduction and Clustering Machine Learning Algorithms for the Analysis of Ship Engine Data

Theodoros Dimitriou[1], Emmanouil Skondras[1], Christos Hitiris[2], Cleopatra Gkola[3], Ioannis S. Papapanagiotou[1], Dimitrios J. Vergados[3], Georgia Fasoula[4], Stratos Koumantakis[5], Angelos Michalas[2(✉)], and Dimitrios D. Vergados[1]

[1] Department of Informatics, University of Piraeus, Piraeus, Greece
{theodim,skondras,jpapapanagiotou,vergados}@unipi.gr
[2] Department of Electrical and Computer Engineering, University of Western Macedonia, Kozani, Greece
{c.hitiris,amichalas}@uowm.gr
[3] Department of Informatics, University of Western Macedonia, Kastoria, Greece
{c.gkola,dvergados}@uowm.gr
[4] Internet Business Hellas, Athens, Greece
geo.fasoula@ibhellas.gr
[5] MAS S.A., Athens, Greece
skoumantakis@maseurope.com

Abstract. Machine Learning (ML) is being successfully applied to ship engine management with proven economic and environmental benefits by engine performance optimization, timely fault detection and appropriate service planning. However, the data preparation for usage in ML algorithms provides several advantages including faster training and improved performance of the algorithm, improved visualization of the dataset, noise reduction, dataset simplification, avoidance of the curse of dimensionality and improved resource utilization. In this paper, two key techniques of the ML algorithms, that can be applied for data preparation and organization of ship engine data are studied, namely the dimensionality reduction and the data clustering. Dimensionality reduction involves the reduce of the number of input variables or features in a dataset, by retaining as much valuable information as possible. On the other hand, clustering ML techniques help to uncover insights and reduce data complexity through the organization of the data into clusters. Evaluation results demonstrate the usefulness of both techniques.

Keywords: Unsupervised Machine Learning · Dimensionality Reduction · Data Clustering · Ship Engine Data

1 Introduction

The problem of handling a high volume of data in Machine Learning (ML) [1] is a critical challenge. As the information stored to ML datasets grows exponentially, it becomes increasingly difficult to efficiently process, store, and analyze the data. This high volume

L. A. Maglaras and C. Douligeris (Eds.): WiCON 2023, LNICST 527, pp. 82–96, 2024.
https://doi.org/10.1007/978-3-031-58053-6_6

of data can lead to issues such as extended training times, increased computational and storage requirements, and the risk of overfitting, where models may fit noise rather than true patterns. Effective data management and feature selection become critical to mitigate these challenges and ensure that machine learning models can operate effectively and make accurate predictions in the face of vast datasets.

Dimensionality reduction [2] and clustering [3] are considered as two critical parts of the ML science. In particular, dimensionality reduction involves the reduction of the number of input variables or features in a dataset, while retaining as much valuable information as possible. This process offers several advantages for the ML algorithms. Indicatively, the smaller datasets resulting from dimensionality reduction are faster to process. This situation leads to faster training of the ML algorithms, as well as it improves their performance in data prediction tasks. Also, it has to be noted that it is challenging to visualize data in high-dimensional spaces. Dimensionality reduction techniques can transform the data into lower dimensions, making it easier to visualize and interpret, which can aid in data analysis. Furthermore, high-dimensional data often contains noise and irrelevant features. Dimensionality reduction helps eliminate or reduce the impact of these noisy features, leading to cleaner and more relevant data.

Another important advantage of the dimensionality reduction is referred as "avoidance of the curse of dimensionality". Specifically, high-dimensional data can suffer from the "curse of dimensionality", where the data become sparse, and distances between data points lose their meaning. Dimensionality reduction can mitigate this problem by preserving the most important dimensions. Last but not least, improved memory and storage utilization are accomplished through the dimensionality reduction. In particular, reduced dimensionality means less resources are required to store the data. Common dimensionality reduction techniques include the Principal Component Analysis (PCA) [4] and the Recursive Feature Elimination (RFE) [5].

Data clustering is also a significant tool in ML. Similar to dimensionality reduction, data clustering is also a useful technique in ML, providing several advantages. Indicatively, data clustering helps uncover hidden patterns and structures within a dataset by grouping similar data together. This can lead to insights and discoveries that might not be apparent when examining the data as a whole. Furthermore, data clustering reduces the complexity of a dataset by dividing it into distinct clusters. This simplification can make it easier to understand, visualize, and analyze large datasets. Also, outlier values that exist into a dataset can be identified through the application of data clustering. This is valuable in a variety of problems including the detection of engines malfunction.

Clustering can also be used as a feature engineering technique. Instead of using the original features, cluster labels can be used as new features, potentially improving model performance in classification or regression tasks. Examples of clustering algorithms include the K-Means [6] and the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [7].

In both dimensionality analysis and data clustering, the choice of the appropriate algorithm depends on the specific characteristics of the dataset and the goals that the analysis should satisfy.

The main contribution of this work, is that based on the analysis performed, the most effective ML algorithms for performing dimensionality reduction and clustering in datasets containing ship engine data will be selected. These algorithms will be applied for the analysis of real ship engine data that will be produced from sensors installed in specific ships, in future work.

The remainder of the paper is organized as follows: Sect. 2 performs an overview of existing ML algorithms for dimensionality reduction and clustering. Subsequently, in Sect. 3 the performance of each algorithm is evaluated. Finally, Sect. 4 concludes our work.

2 Application of Machine Learning Algorithms for Dimensionality Reduction and Data Clustering

Dimensionality reduction and data clustering are two of the most important categories of the unsupervised ML algorithms.

Dimensionality reduction improves the performance of ML algorithms through the mitigation of overfitting, achieves faster training and predicting, enhances the data visualization, reduces the noise data in a dataset, helps the avoidance of the curse of dimensionality, as well as decreases the memory and storage requirements. As a result, it optimizes the data, making them more manageable and relevant while preserving essential information, ultimately leading to more efficient and effective machine learning models.

Data clustering offers several advantages, since it helps the discovery of hidden patterns of data and simplifies complex datasets. Also, data clustering assists to the exploration of unlabeled data, and helps preprocess data by addressing missing values and outliers. Moreover, it improves data understanding and aids in data-driven decision-making processes.

2.1 Dimensionality Reduction Algorithms

In this subsection, two well-known ML algorithms for performing dimensionality reduction are described, namely the Principal Component Analysis (PCA) and the Recursive Feature Elimination (RFE) algorithms.

The Principal Component Analysis (PCA) Algorithm. The PCA algorithm is used to decompose a multivariate dataset into a sequence of - mutually orthogonal components that explain, to the greatest extent, the variation in the dataset. PCA performs linear dimensionality reduction using Singular Value Decomposition (SVD) of the data, projecting it into a lower dimensional space than the original. PCA is implemented in scikit-learn library [8] as a transformer object that uses the fit method to extract the mutually orthogonal components [9, 10], and then it can be used on new data to project them onto those components.

Code Listing 1 shows how to use the PCA algorithm of the scikit-learn library, and how to check the ratio of explained variance and the unique values of the components.

- In lines 1 and 2, the PCA class from the decomposition module of the scikit-learn library, and the NumPy (np) library is imported, which is widely used for numerical computations in Python.
- Line 3 defines a two-dimensional input dataset as a NumPy array, with eight points and two attributes.
- Line 4 creates an instance of the PCA class, with the number of components equal to 2 (n_components = 2). Before the instance is assigned to the model variable, it is trained with the fit method, which takes the input data as a parameter and returns the trained model.
- Line 5 requests the display of the percentage of explained (interpreted) variation for each component, which are shown in line 6. Specifically, the first and second components show percentages of explained variation of 97.619048% and 2.380952% respectively of the total variation. Both components together account for 100% of the total variance since the example dataset contains only these two features.
- Finally, line 7 requests the display of the unique values of each component, which are displayed in line 8.

Code Listing 1. Usage Example of the PCA algorithm.

```
1: >> from sklearn.decomposition import PCA
2: >> import numpy as np
3: >> data = np.array([[4,-4],[2,-2],[1,0],[0,-1],[0,1],[-1,0],[-2,2],[-4,4]])
4: >> model = PCA(n_components=2).fit(data)
5: >> model.explained_variance_ratio_
6: array([0.97619048, 0.02380952])
7: >> model.singular_values_
8: array([9.05538514, 1.41421356])
```

The Recursive Feature Elimination (RFE) Algorithm. The RFE algorithm is a recursive process that ranks features according to some measure that expresses their importance. In each iteration, the importance of each feature is assessed. Then, the attribute

with the least importance is removed from the attributes considered. Another possibility is to remove an entire group of features at each iteration (instead of just a single feature), in order to speed up the process. Repeating the process removing one feature at a time is necessary. This is because in some cases the relative importance of each attribute may change when it is evaluated against a different subset of attributes during the stepwise elimination process. RFE was exploited in the present project, with the aim of investigating the performance relationship of features of ship engine data, using a number of features as input for training to select a subset of the most important features as output.

Code Listing 2 shows how to use the scikit-learn library's RFE algorithm to select the most important features from a dataset using a Support Vector Regression (SVR) linear regression model.

- Line 1 imports the RFE class to the algorithm, line 2 produces an indicative dataset using the make_regression class of the sklearn.feature package and line 3 imports the LinearRegression model class.
- Line 4 uses the make_regression function to create the X, Y dataset which includes 100 samples of 15 features, setting the random state parameter equal of 5.
- Line 5 initializes the estimator to be used in RFE, creating a linear support vector regression model using the LinearRegression class.
- Line 6 initializes the variable of the model, by an instance of the RFE class, which takes as arguments the estimator, along with the number of features to select which in this case is equal to 4.
- Line 7 fits the model to the dataset (X, Y) using the fit method.
- Line 8 retrieves the ranking of the features by reading the model.ranking_ property, which returns an array of integers representing the ranking of each attribute. A rank of 1 indicates that the feature is selected. The output is presented at line 9, which shows that the fourth, the sixth, the seventh and the fifteenth features are selected since they have a rank equal to 1. The remaining features are not selected.

Code Listing 2. Usage Example of the RFE algorithm.

```
1: >> from sklearn.feature_selection import RFE
2: >> from sklearn.datasets import make_regression
3: >> from sklearn.linear_model import LinearRegression
4: >> X, Y = make_regression(n_samples = 100, n_features = 15, random_state=5)
5: >> estimator = LinearRegression()
6: >> model = RFE(estimator, n_features_to_select=4)
7: >> model.fit(X, Y)
8: >> model.ranking_
9: array([10, 6, 9, 1, 5, 1, 1, 12, 11, 7, 4, 2, 3, 8, 1])
```

2.2 Clustering Machine Learning Algorithms

In this subsection, two well-known ML algorithms for performing Clustering of data are studied, namely the K-Means and the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithms.

The K-Means Algorithm. The K-Means algorithm is a popular unsupervised learning algorithm used for data clustering. The algorithm creates a set of groups (clusters) and categorizes in each group the data it receives as input based on the similarity between them. The steps of the K-Means algorithm, follow below:

1. Definition of the number of groups (clusters) to be created: In this step, the number (k) of clusters to be produced is determined by the user.
2. Initialization: k points are randomly selected from the data, as initial centers of the groups (centroids).
3. Placement-categorization of data into clusters: For each of the input data, its distance from the center of each cluster is calculated and it is placed in the cluster where the smallest distance is observed.
4. Update centers: Cluster centers are recalculated as the average of the data values placed in each cluster.
5. Iteration: Steps 3 and 4 are repeated until the centers of the clusters no longer change and convergence of the algorithm has been achieved

Code Listing 3 shows the use of the K-Means implementation of the scikit-learn library.

- In lines 1 and 2, the necessary libraries (KMeans and numpty) are imported into the program.
- In line 3, the dataset is defined on which the K-Means algorithm will be applied, in order to group the data present there by creating clusters. The given dataset includes 16 samples with 2 feature values per sample.
- In line 4, initially the appropriate values to the initialization parameters of the KMeans class are set. In particular, for the parameter n_clusters we set the value 4, so that 4 clusters will be created in which the samples present in the dataset will then be grouped. Also, for the random_state parameter we set the value 5, which means that every time a new sample is introduced into a cluster, then the centroid of the cluster will be recalculated as the average value of the samples present in the cluster. For the parameter n_init we set the value auto, which means that the number of executions of the algorithm will depend on the number of samples present in the dataset. Subsequently, the dataset that the algorithm is applied to the KMeans object to perform clustering.
- In line 5, we request the clusters in which each one of the 16 samples of the dataset is grouped. The result of this command is shown in line 6.
- In line 6, we notice that the first 4 samples of the dataset (i.e. [0, 1], [1, 2], [2, 1] and [1, 0],) are grouped in cluster 0, the next 4 samples of the dataset (i.e. [0, 9], [1, 10], [2, 9] and [1, 8]) are grouped into cluster 3, the next 4 samples of the dataset (i.e. [8, 9], [9, 10], [10, 9] and [9, 8]) are grouped into cluster 1, and the final 4 samples of the dataset (i.e. [8, 1], [9, 2], [10, 1] and [9, 0]) are grouped into cluster 2.

- In line 7 we ask the algorithm to show us the centroids of the four clusters created. The result of this command is shown in line 8.
- In line 8, we notice that the center of cluster 0 is equal to [1., 1.]. We recall that cluster 0 contains four samples namely the [0, 1], [1, 2] and [1, 0]. Based on these samples, we confirm that the center of cluster 0 has been correctly calculated, as the average value obtained from the three samples of that particular cluster. The centers of the rest clusters are evaluated similarly.
- In line 9, we request the algorithm to categorize into the clusters four new samples, namely the [1, 1], the [1, 9], the [9, 9] and the [9, 1]. The result of this command is shown in line 10.
- In line 10, we notice that the algorithm categorized sample [1, 1] into cluster 0, sample [1, 9] into cluster 3, sample [9, 9] into cluster 1 and sample [9, 1] into cluster 2.

Code Listing 3. Usage Example of the K-Means algorithm.

```
1:  >> from sklearn.cluster import KMeans
2:  >> import numpy as np
3:  >> data = np.array([[0,1],[1,2],[2,1],[1,0],      [0,9],[1,10],[2,9],[1,8],
                        [8,9],[9,10],[10,9],[9,8],   [8,1],[9,2],[10,1],[9,0]])
4:  >> algorithm = KMeans(n_clusters=4, random_state=5,
        n_init="auto").fit(data)
5.  >> algorithm.labels_
6:  array([0, 0, 0, 0, 3, 3, 3, 3, 1, 1, 1, 1, 2, 2, 2, 2], dtype=int32)
7:  >> algorithm.cluster_centers_
8:  array([[1., 1.], [1., 9.], [9., 9.], [9., 1.]])
9:  >> algorithm.predict([[1,1],[1,9],[9,9],[9,1]])
10: array([0, 3, 1, 2], dtype=int32)
```

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) Algorithm. The DBSCAN algorithm detects and clusters the data located in high-density areas, while ignoring the data located in low-density areas since it considers them as noise.

The DBSCAN algorithm starts by randomly selecting a value from the dataset. Next, the distance of neighboring values from the selected value is examined. If the density of neighboring values is high enough (greater than the minimum acceptable density defined by the user), then a cluster is created, in which the selected value and neighboring values are placed. Conversely, if the density of neighboring values is not high enough, the algorithm continues by looking at the next value present in the dataset.

The DBSCAN algorithm continues this process by incrementally examining all values present in the dataset. Finally, the values that do not belong to any cluster are classified as noise. Code Listing 4 shows the use of the DBSCAN algorithm of the scikit-learn library.

- In lines 1 and 2, the necessary libraries (DBSCAN and numpty) are imported into the program.
- Line 3 defines the dataset on which the DBSCAN algorithm will be applied, in order to group the data into clusters. The given dataset includes 5 samples with 2 feature values per sample.
- In line 4, we initialize the DBSCAN algorithm. In particular, the eps parameter expressing the maximum acceptable distance between two neighbors is set to 2. The min_samples parameter expresses the minimum required number of samples that must be present in a cluster and in our case, it is set equal to 2.
- In line 5, the DBSCAN algorithm is applied to the dataset using the fit command.
- In line 6, we request the clusters in which each one of the samples of the dataset is grouped. The result of this command is shown in line 7.
- In line 7, the labels of the clusters to which the samples are grouped are displayed. We notice that 2 clusters were created. In the first cluster, with label 0, the first 2 samples of the dataset, namely the [2, 1] and the [1, 2], are included. Moreover, the samples [5, 4] and [4, 5] are grouped into the second cluster with label 1. Finally, the third sample [3, 3] cannot be grouped to any cluster, because this sample is characterized as noise by the DBSCAN algorithm and thus it appears to be placed into a cluster with label −1.

Code Listing 4. Usage Example of the DBSCAN algorithm.

```
1: >> from sklearn.cluster import DBSCAN
2: >> import numpy as np
3: >> data = np.array([[2,1],[1,2],[3,3],[5,4],[4,5]])
4: >> model = DBSCAN(eps=2, min_samples=2)
5: >> model.fit(data)
6: >> model.labels_
7: array([ 0,  0, -1,  1, 1])
```

3 Performance Analysis

In this section, the dimensionality reduction algorithms (PCA and RFE) and the clustering algorithms (K-Means and DBSCAN) are evaluated using the Condition Based Maintenance of Naval Propulsion Plants (CBM) dataset [11, 12]. The CBM dataset contains 11934 records with data about ship engines. Each record consists of the 16 features presented in Table 1. For the evaluation of the algorithms, only the 14 features will be considered, since the features 9 (gas turbine compressor inlet air temperature) and 12 (gas turbine compressor inlet air pressure) obtain the same value in the entire records and, thus, they are considered as non-important for our analysis.

It has to be noted that the CBM dataset has not been analysed yet in the existing literature, in situations where dimensionality reduction or data clustering is applied. Thus, our analysis provides useful insights about the structural characteristics of the dataset.

Table 1. The features included to the CBM dataset.

No.	Feature Name	Abbreviation	Measurement Unit
1	Lever position	lever_position	[1–9]
2	Ship speed	ship_speed	Knots
3	Gas Turbine (GT) shaft torque	gt_shaft	Kilonewton/meter
4	GT shaft rate	gt_rate	Rounds per Minute
5	Gas Generator (GG) rate	gg_rate	Rounds per Minute
6	Starboard Propeller Torque	sp_torque	Kilonewton
7	Port Propeller Torque	pp_torque	Kilonewton
8	High Pressure (HP) Turbine exit temperature	hpt_temp	Celsius degrees
9	GT Compressor inlet air temperature	gt_c_i_temp	Celsius degrees
10	GT Compressor outlet air temperature	gt_c_o_temp	Celsius degrees
10	HP Turbine exit pressure	hpt_pressure	Bar
11	GT Compressor inlet air pressure	gt_c_i_pressure	Bar
12	GT Compressor outlet air pressure	gt_c_o_pressure	Bar
13	GT exhaust gas pressure	gt_exhaust_pressure	Bar
14	Turbine Injection Control	turbine_inj_control	Presentence (%)
15	Fuel flow	fuel_flow	Kilograms/second
16	Lever position	lever_position	[1–9]

3.1 Evaluation of Dimensionality Reduction Algorithms

This subsection presents the results of the execution of the PCA and the RFE algorithms using the CBM dataset.

Therefore, an instance of the PCA class using the CBM training dataset was created and the explained variance of each component of the resulting new vector space basis was extracted as a result in the explained_variance_ratio_ variable, as shown in Table 2. As an indication of the effectiveness of PCA, the first component explains 99.96569% of the total variance in the CBM data set, with the next two components in order of rank, explaining 0.03% and 0.002% respectively, and with the others following.

In addition, the cumulative variance of the training dataset interpreted by the components is shown in Fig. 1. We observe that the first component explains almost all (99.96569%) of the variance, with subsequent components contributing minimally and from the 4th component onwards, the contribution is practically zero.

Table 2. The explained variance of each component in the new vector space basis.

Component	Explained variance
1	0.999656935554048
2	0.000320778697501
3	2.10E−05
4	1.15E−06
5	8.59E−08
6	4.34E−08
7	1.43E−08
8	1.40E−09
9	2.67E−11
10	3.77E−13
11	1.47E−13
12	3.97E−14
13	1.73E−16
14	1.58E−39

The execution of the RFECV [13] that finds the optimal number of features for RFE using cross-validation, it is obtained that the optimal number of dimensions is also quite different for the two target decay features. For the prediction of the gas turbine compressor decay coefficient "gt_c_decay" using the Extra Trees Regressor algorithm, the optimal number of dimensions is 7 (Fig. 2), while for the gas turbine decay coefficient "gt_t_decay" the optimal number of dimensions is 11 (Fig. 3).

Accordingly, in performing the RFE, using as desired number of attributes equal to 7 for the gas turbine compressor decay coefficient (gt_c_decay) and 11 for the gas turbine decay coefficient (gt_t_decay), it was observed that the order of ranking the importance of the attributes is different for each of the two target decay attributes, as shown in Table 3. For the prediction of gt_c_decay the 7 most important features are: gt_c_o_temp, gg_rate, gt_exhaust_pressure, gt_shaft, hpt_pressure, gt_rate, hpt_temp. Similarly, for the prediction of gt_t_decay the 11 most important features are: gt_c_o_pressure, fuel_flow, gg_rate, turbine_inj_control, gt_shaft, gt_rate, hpt_temp.

To further evaluate the results of the RFE algorithm, the importance of each feature is evaluated as extracted during the training of the ML algorithms Random Forest Regressor [14], the Extra Trees Regressor [15] and the Decision Tree Regressor [16]. The importance of each feature is ranked based on its contribution for predicting the gt_c_decay feature. Figure 4 shows the rankings produced by the three ML algorithms, indicating that the same seven features of the case of RFE are considered as the most important features.

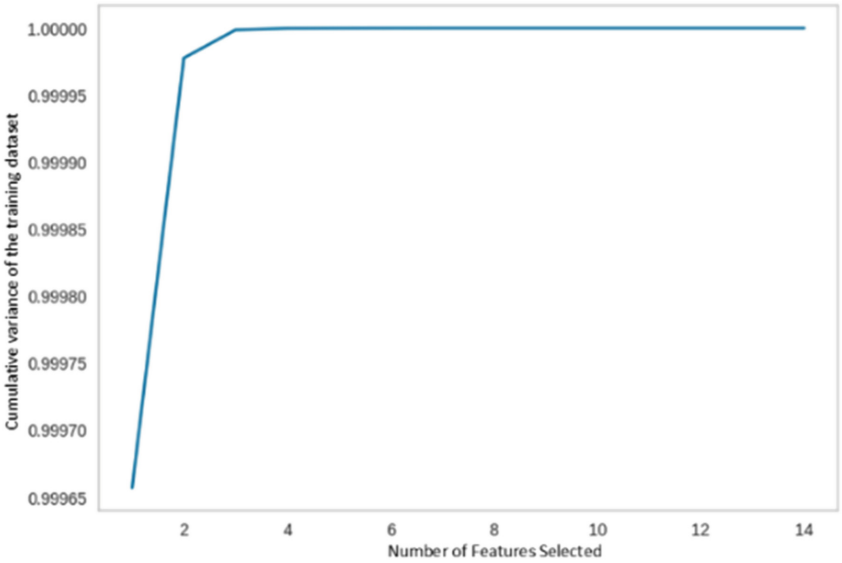

Fig. 1. Cumulative variance of the training dataset as interpreted by the components.

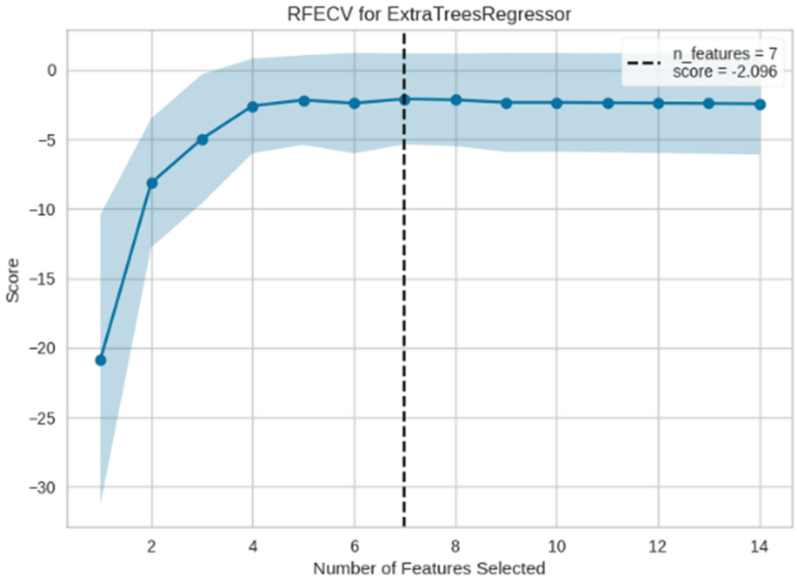

Fig. 2. The cross-validation method for predicting the gas turbine compressor decay factor gt_c_decay.

According to the experimental results, it has to be noted that both PCA and RFE can be considered as appropriate algorithms for performing dimensionality reduction in ship engine data.

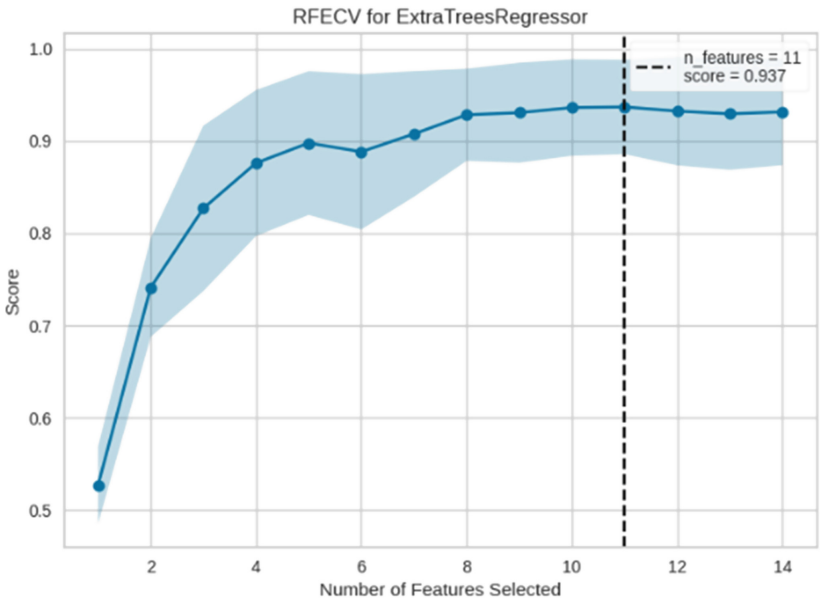

Fig. 3. The cross-validation method for predicting the gas turbine decay factor gt_t_decay.

Table 3. Rank of feature importance for the gt_c_decay and the gt_t_decay.

Name	gt_c_decay rank	gt_t_decay rank
lever_position	13	14
ship_speed	14	13
gt_shaft	4	5
gt_rate	6	6
gg_rate	2	3
sp_torque	8	11
pp_torque	9	12
hpt_temp	7	7
gt_c_o_temp	1	10
hpt_pressure	5	9
gt_c_o_pressure	12	1
gt_exhaust_pressure	3	8
turbine_inj_control	10	4
fuel_flow	11	2

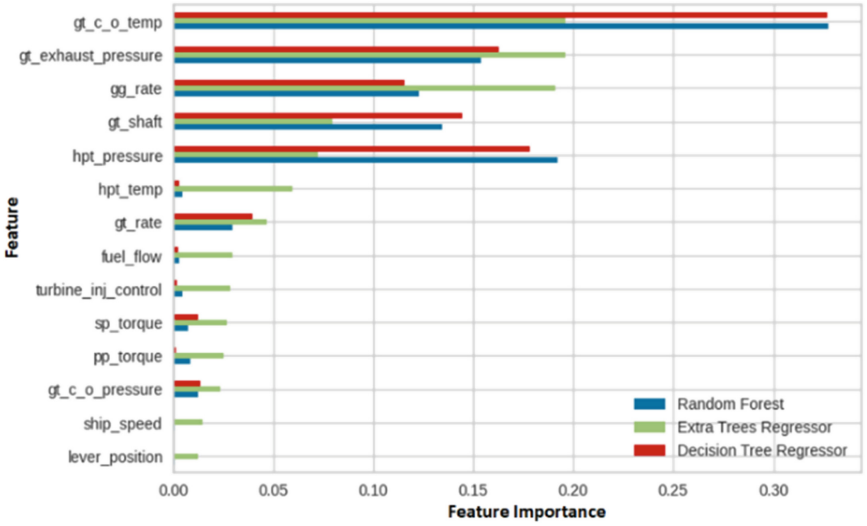

Fig. 4. Ranking of features by importance for predicting gt_c_decay, from the three decision tree algorithms.

3.2 Evaluation of Clustering Algorithms

For the evaluation of the K-Means and the DBSCAN algorithms, an indicative subset of values of the lever_position and the fuel_flow features of the CBM dataset is used.

Regarding the initialization of the two clustering algorithms, the n_clusters parameter of the K-Means is initialized with a value equal to 2, so that 2 clusters will be created in which the samples present in the dataset will be grouped. Also, the random_state parameter is initialized with a value equal to 5, which means that every time a new sample is grouped into a cluster, then the centroid of the cluster will be recalculated as the average value of the samples present in the cluster. Furthermore, the DBSCAN algorithm has been applied by setting the eps parameter equal to 2, which is the maximum acceptable distance between two values in order to be considered as neighbors. Also, the min_samples parameter of the DBSCAN is also set to be equal to 2, and thus each cluster created will include at least 2 samples.

Figure 5 presents the results obtained after the application of the two algorithms. As it can be observed the K-Means algorithm created two clusters. The first cluster contains three samples and the second cluster contains the remaining two samples. It is obvious that the algorithm grouped the samples into the two clusters based on the distances among the values of the samples. Accordingly, the DBSCAN algorithm also created two clusters. However, in this case each cluster contains two samples, while one sample is considered as outlier (or noise) by the DBSCAN algorithm and it is not grouped to any cluster. Since DBSCAN can detects outlier values, it can be considered as more appropriate than the K-Means for the analysis of ship engine data, because in such cases outlier values may indicate failure in engine components.

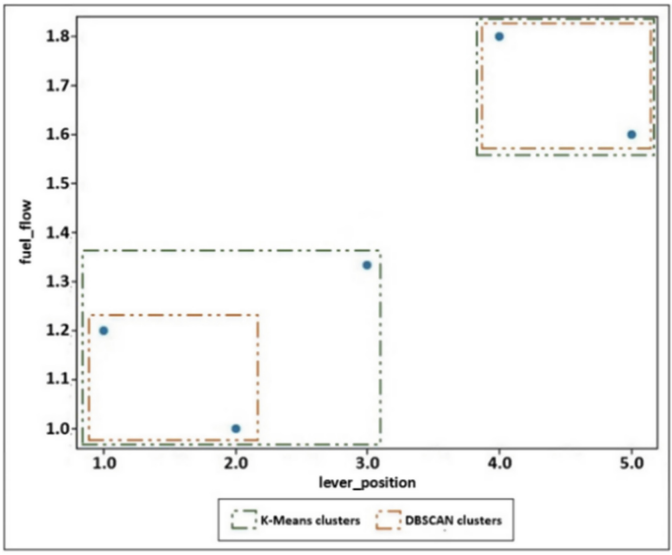

Fig. 5. The clusters created using the K-Means and the DBSCAN algorithms.

4 Conclusion

In this work, useful algorithms for performing dimensionality reduction as well as data clustering are studied. In particular, dimensionality reduction offers important advantages to ML algorithms, such as faster training of these algorithms, simplification of complex datasets and improved memory and storage utilization. Regarding this category of ML algorithms, two alternative techniques studied, namely the PCA and the RFE algorithms. On the other hand, the advantages offered by data clustering techniques include the uncover of insights and the reduction of data complexity through the organization of the data into clusters. During the study of this category of algorithms, two alternative techniques also studied, namely the K-Means and the DBSCAN algorithms. The functionality of both dimensionality reduction and data clustering algorithms are evaluated through experiments, where the usefulness of both algorithm categories demonstrated. Based on the analysis performed, future work includes the application of the most effective dimensionality reduction and clustering ML algorithms in data generated from ship engines' operation.

Acknowledgements. This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH-CREATE-INNOVATE (project code: T2EDK-C.873).

References

1. Dogan, A., Birant, D.: Machine learning and data mining in manufacturing. Expert Syst. Appl. **166**, 1–22 (2021)

2. Zhang, G., Wang, Z., Huang, H., Li, H., Sun, T.: Comparison and evaluation of dimensionality reduction techniques for the numerical simulations of unsteady cavitation. The acoustic signature of a propeller-hydrofoil system in the far field. Phys. Fluids **35**(7) (2023)
3. Ezugwu, A.E., et al.: A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. Eng. Appl. Artif. Intell. **110**, 1–43 (2022)
4. Park, J., Oh, J.: Analysis of collected data and establishment of an abnormal data detection algorithm using principal component analysis and K-nearest neighbors for predictive maintenance of ship propulsion engine. Processes J. **10**(11), 1–13 (2022)
5. Habibi, A., Delavar, M.R., Sadeghian, M.S., Nazari, B., Pirasteh, S.: A hybrid of ensemble machine learning models with RFE and Boruta wrapper-based algorithms for flash flood susceptibility assessment. Int. J. Appl. Earth Obs. Geoinf. **122**, 1–18 (2023)
6. Ikotun, A.M., Ezugwu, A.E., Abualigah, L., Abuhaija, B., Heming, J.: K-means clustering algorithms: a comprehensive review, variants analysis, and advances in the era of big data. Inf. Sci. **622**, 178–210 (2023)
7. Xu, X., Cui, D., Li, Y., Xiao, Y.: Research on ship trajectory extraction based on multiattribute DBSCAN optimisation algorithm. Pol. Marit. Res. 136–148 (2021)
8. Scikit-learn library. https://scikit-learn.org. Accessed 30 Oct 2023
9. Decomposing signals in components, Scikit learn. https://scikit-learn.org/stable/modules/decomposition.html. Accessed 30 Oct 2023
10. Principal Component Analysis (PCA) class, Scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. Accessed 30 Oct 2023
11. The Condition Based Maintenance of Naval Propulsion Plants (CBM) dataset. https://www.kaggle.com/datasets/elikplim/maintenance-of-naval-propulsion-plants-data-set. Accessed 30 Oct 2023
12. Coraddu, A., Oneto, L., Ghio, A., Savio, S., Figari, M., Anguita, D.: Machine learning for wear forecasting of naval assets for condition-based maintenance applications. In: IEEE International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS), pp. 1–5 (2015)
13. Recursive Feature Elimination with Cross-Validation (RFECV) scikit-learn class. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFECV.html. Accessed 30 Oct 2023
14. Graw, J.H., Wood, W.T., Phrampus, B.J.: Predicting global marine sediment density using the random forest regressor machine learning algorithm. J. Geophys. Res. Solid Earth **126**(1), 1–14 (2021)
15. John, V., Liu, Z., Guo, C., Mita, S., Kidono, K.: Real-time lane estimation using deep features and extra trees regression. In: Bräunl, T., McCane, B., Rivera, M., Yu, X. (eds.) PSIVT 2015. LNCS, vol. 9431, pp. 721–733. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29451-3_57
16. Colditz, R.R.: An evaluation of different training sample allocation schemes for discrete and continuous land cover classification using decision tree-based algorithms. Remote Sens. **7**(8), 9655–9681 (2015)