

## RESEARCH ARTICLE

# Efficient Machine Learning on Edge Computing Through Data Compression Techniques

NEREA GÓMEZ LARRAKOETXEA<sup>1</sup>, JOSEBA ESKUBI ASTOBIZA<sup>2</sup>, IKER PASTOR LÓPEZ<sup>2</sup>,  
BORJA SANZ URQUIJO<sup>2</sup>, JON GARCÍA BARRUETABEÑA<sup>2</sup>, AND AGUSTIN ZUBILLAGA REGO<sup>2</sup>

<sup>1</sup>Deusto University, Bilbao, 48007 Bizkaia, Spain

<sup>2</sup>Facultad de Ingeniería, Universidad de Deusto, 48007 Bilbao, Spain

Corresponding author: Nerea Gómez Larrakoetxea (ngomez@deusto.es)


**ABSTRACT** This paper discusses the increasing amount of data handled by companies and the need to use *Big Data* and *Data Analytics* to extract value from this data. However, due to the large amount of data collected, challenges related to the computational capacity of machines often arise when performing this analysis to acquire relevant information for the organization, especially when we are using edge computing. The paper aims to train machine learning models using compressed data, with two compression techniques applied to the original data. The results show that models trained with compressed data achieved similar accuracy to those trained with uncompressed data, and different compression techniques were compared. The research extended a previous study by analyzing the use of autoencoders for compression and reducing both instances and dimensionality of the dataset. The accuracy rate of the models when trained with compressed data instead of original data was maintained.

**INDEX TERMS** Autoencoder, Bayesian network, big data, edge computing, machine learning.

## I. INTRODUCTION

Industry 4.0, often known as “The Fourth Industrial Revolution” is transforming how organizations operate with the aim of transforming it into a smart organization capable of achieving superior commercial results [1]. Higher levels of efficiency and productivity, as well as a higher level of automation, are the main targets of Industry 4.0 [2]. Additionally, a novel feature of Industry 4.0 is the ability to have all of the information needed in real time, despite of where in the plant the process is taking place, allowing for faster reactions to market demands [3].

Multiple factors may be crucial in determining whether a machine may fail in the short term, including process timeframes, voltage or current readings, or humidity or pressure levels. A forecasting analytics model is able to identify the common trends that typically lead to machine failures based on the information gathered from these many data sources. This enables the development of a numerical model which can identify these trends prior to the occurrence of faults,

The associate editor coordinating the review of this manuscript and approving it for publication was Wentao Fan .

allowing for the conduct of preventative maintenance on the machine without the production being halted as a result of the failure.

However, what if similar results may be achieved using far smaller number of instances?

For this purpose, this paper addresses the utility and value that reduced datasets could offer when used in conjunction with Machine Learning algorithms: Starting with a huge data collection that contains numerous instances, the principal target is to generate a smaller dataset that retains the same or almost same accuracy outcomes when Machine Learning algorithms are applied. This entails creating a tiny dataset with a sufficient number of representative cases from the larger set. It should be noted that the compression is done with the aim that the reduced set will create Machine Learning models as accurate as those created with the original set, any other use of the reduced dataset is not considered.

To obtain a tiny but representative dataset, new approaches are required to minimize the volume of datasets up to 75%. However, for the model accuracy of the reduced dataset to have the same level of precision as the original one is highly

improbable. Should the accuracy be lower, the lost accuracy is considered to be negligible. For this purpose, Bayesian compression and autoencoders analysis have been employed. Bayesian Networks were applied to the original data sets using the Weka software platform's libraries. Depending on the reduction rate to be applied to the original data set, the total number of outputs of the Bayesian network will vary. On the other hand, as far as autoencoders is concerned, they are composed of two main elements: an encoder to which the original dataset will be given to generate a latent vector, and a decoder, in charge of reconstructing the original data from the previously obtained latent vector. In this case, as we are only interested in compression, the decoding part will not be performed. Both techniques, Bayesian networks and autoencoders, have been used with compressions of 25%, 50% and 75%. The comparison of accuracy results obtained through the use of machine learning algorithms between the original and compressed datasets showed that the main target of keeping the accuracy rate of the model has been reached in most cases.

Therefore, this research is aimed at evaluating the accuracy of Machine Learning models trained using compressed data. For that, the mentioned two compression techniques have been developed. The accuracy of the models trained with compressed data has been compared to those trained with uncompressed data, where an equivalent accuracy has been reached.

It should be noted that this document is structured according to the following. Section II describes the state of the art or background. Section III introduces the experimentation conducted in this research including the proposed methodology, validation of the experiments and the results. Finally, Section IV presents our conclusions. Section V sets our suggestions for future work.

## II. BACKGROUND

Several strategies have been developed over the years to both minimize the number of instances in the models and to lessen the complexity of the network's structure. When it came to both cases, pruning was one of the first approaches that was implemented: As far back as 1997, Wilson and R. Martinez proposed three techniques for reducing the training dataset's number of instances [4] and in the other hand, LeCun, Denker, and Solla proposed "Optimal Brain Damage", a novel technique for shrinking the size of the learning network by gradually deleting weights [5]. Additionally, algorithms employing Bayesian approaches and weight reduction have been developed with success [6], [7], [8]. In any instance, since the array format is ignored when doing Bayesian compression, weight pruning is ineffective at compressing data. To address this issue, the Compressed Sparse Column (CSC) format was developed [9]. These studies compressed the network topologies but not the input data using Bayesian Compression.

Still in 2017 [10], intended to tackle the subject through a Bayesian perspective. It was first used to prune nodes

instead of individual weights, and it was then used to calculate the appropriate fixed point precision to encode the weights, based on the posterior uncertainties of the priors.

Lin et al. [11] and X. Li et al. [12] conducted subsequent research on the Optimal Fixed Point and hashing quantization, respectively, and published their findings. In terms of the Optimal Fixed Point, they developed a method for determining the fixed point's bandwidth using the DCN<sup>1</sup> layers. Contrary to equal bit width settings, the experiment revealed that optimizing bit width in DCNs results in a model size reduction of more than 20% without sacrificing accuracy on the CIFAR-10 test [13]. Nonetheless, Wenlin Chen et al. used a Discrete Cosine Transform (DCT) to convert filter weights to the frequency domain and a low-cost hash function to randomly combine frequency parameters into hash buckets for hashing quantization.

Regarding the compression through autoencoders, after studying variational autoencoders for unsupervised clustering through deep generative model [14], Dilokthanakul et al. observed an over-regularization problem that happens in normal variational autoencoders. To solve it they propose to use a minimum information constraint which improves the unsupervised clustering. In 2021, Fang et al. [15] showed how to use the latent variable modelling to improve controllability without hurting state-of-the-art generation effectiveness. To do so a pre-trained Transformer-based architecture was integrated with latent representation vectors creating a conditional variational autoencoders.

Regarding the literature on data compression in a more general way, that is, without considering the field of Machine Learning, various studies have been carried out on compression techniques. Fowler and Yangel [16] presented a lossless data-compression algorithm, which achieves better reduction performance than the generic compression algorithms that are commonly available on modern computer systems. There are also compression codecs for the compression of multimedia content such as videos for example [17], [18], and [19]. There are also techniques developed for the compression of integer sequences, such as the universal code technique [20], [21].

Following the current state of the art, it is worth noting that a great deal of research has been carried out on the compression of all kinds of items: sequences of numbers, multimedia content, videos, voice, etc. On the contrary and although there are studies focused on reducing the complexity of the algorithms, there is none focused on studying the accuracy of Machine Learning models created with compressed sets.

## III. DATA COMPRESSION TECHNIQUES

This section will present two different compression techniques that have been used in this paper.

<sup>1</sup>DCN: Deep Convolutional Network.

**A. NAIVE BAYESIAN CLASSIFIER**

Classification is the process of associating an item (instance, data) with a certain class (category). A Bayesian classifier is a statistical classification algorithm that is based on Bayes’ theorem. Through employing these classifiers, an instance’s likelihood of belonging to a class can be predicted [22]. For example, a photograph could be classified as landscape, portrait, or urban. Classification is a mathematical method that entails assigning a class,  $c$ , from a set of classes  $C$  to a given instance defined by a vector of characteristics or attributes [23]. These classifiers make the premise that an attribute’s value has no effect on the other attributes’ values inside a particular class. These classifiers make the premise that an attribute’s value has no effect on the other attributes’ values inside a particular class. “Conditional class independence” is the name given to this hypothesis. Its purpose is to make the calculations easier [22]. Classifying everything that is interpreted through the senses is something that comes naturally to human beings; in essence, it allows us to abstract the information, transforming it into a form that is more suitable for decision-making purposes. Classification has a wide range of applications, including, for example [23]:

- In industry, quality control entails classifying a component or finished product as correct or defective.
- Security systems: determine, for example, if a person has access to a particular location.
- Intelligent vehicles: identifying pedestrians on the road, classifying items detected via cameras or a variety of other sensors.
- Spam-filtering email readers: remove spam messages.
- Medical imaging analysis is the process of identifying malignancies on x-rays.
- Biometric systems are those that associate a picture of a fingerprint with a specific individual.

As a result, it is critical to develop classifiers, whether in hardware or software, that can assist in the resolution of such issues.

The naive Bayesian classifier operates in the following manner [22]:

- 1) Assume  $Z$  to be a training set of instances, with each instance labeled with a class. There are  $K$  classes,  $C_1, C_2, \dots, C_k$ . Every instance is represented by an  $n$ -dimensional vector,  $X = X_1, X_2, \dots, X_n$ , that contains the  $n$  measured values of the  $n$  attributes,  $A_1, A_2, \dots, A_n$ , respectively.
- 2) In hindsight, the classifier will predict that sample  $X$  belongs to the class with the highest probability, conditioned on  $X$ . That is, it is assumed that  $X$  will belong to the class  $C_i$  if and only if:

$$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m, j \neq i.$$

This way, could be found the class that maximizes  $P(C_i|X)$ . The class  $C_i$  for which  $P(C_i|X)$  is maximized is called the “Maximum posteriori hypothesis”. By Bayes’ theorem:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

- 3) Because  $P(X)$  is constant for all classes, just  $P(X|C_i)P(C_i)$  must be maximized. If the a priori probabilities of the classes,  $P(C_i)$ , are unknown, it is assumed that they are equally likely:  $P(C_1) = P(C_2) = \dots = P(C_k)$ , and therefore would be  $P(X|C_i)$  maximized. Otherwise  $P(X|C_i)P(C_i)$  is maximized. Class a priori probabilities may be estimated by  $P(C_i) = \text{freq}(C_i, T)/|T|$
- 4) With the goal of predicting the class label of  $X$ ,  $P(X|C_i)P(C_i)$ , is evaluated for each class  $C_i$ . The algorithm assumes that  $X$  belongs to the class  $C_i$  provided and only if it is the class maximizes  $P(X|C_i)P(C_i)$ .

**B. AUTOENCODER**

An autoencoder is a neural network that consists of an encoder and a decoder. The aim of the encoder is to transform high-dimensionality data into low-dimensionality data generating a latent vector, while the aim of the decoder is to get the latent vector  $C_{AE}$  or low-dimensionality data and transform it to high-dimensionality data. The entire network learns the identity function  $X_{out} = x$  by optimizing the weights and bias of each unit [12].

The weight and bias matrixes of the encoder are represented by  $W^T$  and  $b$ . The encoder and decoder functions are represented like the ones below [12]:

$$E(x) : W(x) + b \quad D(x) : W^T + b'$$

where  $W$  is weight matrix and  $b$  bias matrix.

The difference between  $x$  and  $X_{out}$  is identified as loss function. The most used loss functions are Mean Absolute Error (MAE) and Mean Square Error (MSE) [12]. The next equation represents the loss function of autoencoders:

$$\min(f_{loss} : (W^T(W_x + b) + b'), x))$$

Finally, back propagation is the method used in order to train the weights and reduce the difference between  $x$  and  $X_{out}$ .

**IV. EXPERIMENTS**

To employ the Bayes Theorem and Autoencoders in front of a huge dataset in order to minimize it without losing any significant information has been the emphasis of this research. As a result, this could be a significant advancement for computational challenges that may develop while processing massive volumes of data, as it would allow for faster response times. In order to do this, this research is being carried out with the goal of producing machine learning models that are trained with a much smaller quantity of data than the original one while preserving equal accuracy. As a result, the model performs just as well even if it was trained on a much lower amount of data. However, how can those models be obtained?

Two different ways of obtaining those models have been analyzed, the experimentation therefore consisting of two

parts: Data instances reduction using Bayesian Networks and Data Dimensionality reduction using autoencoders.

The hardware used for all of the following experiments is a computer with an i7 processor, 32 GB RAM and a RTX 3090 GPU.

**A. INSTANCES REDUCTION**

In order to obtain a dataset reduction in terms of rows, Bayesian Networks were applied to the original data sets using the Weka<sup>2</sup> platform’s libraries. The number of outputs produced by the Bayesian network changes according to the amount of reduction that will be applied to the original dataset in the model under consideration.

**1) ALGORITHM PERFORMANCE**

The following are the steps carefully taken by the algorithm to conduct Bayesian data compression during this experimentation:

- The original data is gathered and reviewed to ensure that each instance has a unitary class for prediction. This check is carried out in order to determine whether or not the classes can be managed by the Weka libraries.
- In order to compress the data, a Bayesian classifier is created from the available data. As soon as it is obtained, the nodes are correctly sorted for the system, which means that the order chosen is the one that ensures that from the location of a node in the list to the right, there is no parent of that node. To store the data, a dictionary is created.
- The amount of instances to gather is determined by the compression percentage specified, and from the node dictionary, the sorted nodes are passed through in the order in which they were returned by the sorting technique. The probability value of the state of the node is acquired, and the values are sorted from lowest to highest in descending order. The specified states are added to the instances and after that, the full instances are added. These values are returned as a set of instances from the dictionary that has been constructed. The number of instances returned will be equal to the compression % specified.

Figure 1 shows the procedure followed to reduce the instances of a dataset.

**B. DIMENSIONALITY REDUCTION**

On the other hand, to obtain a dataset reduction in terms of columns, the experiment was carried out using an Autoencoder. The Autoencoder has two different parts; first an encoder and second a decoder. To work as expected, the input size of the encoder and the output size of the decoder must be exactly the same. Figure 2 shows the structure of an autoencoder.

<sup>2</sup>Weka: Waikato Environment for Knowledge Analysis is a free software licensed which contains a collection of visualization tools and algorithms for data analysis and predictive modeling.

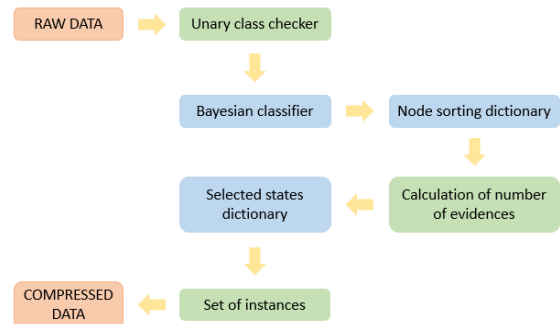


FIGURE 1. Bayesian compression steps diagram.

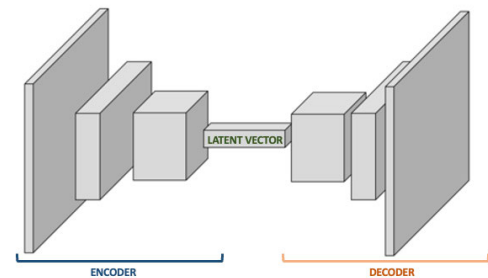


FIGURE 2. Autoencoder’s structure.

To develop the algorithm the different layers of the neural network of the encoder and the decoder must be specify. For this particular case, one dimensionality convolutional layers have been used. This means that, it changes the fully connected layers of the autoencoder structure to convolutional layers [24]. The number of layers added to both parts have been different considering the size of the latent vector, which size depends on the reduction rate. For instance, if the aim is to reduce the dataset by 50%, the number of layers needed will be different than if the aim is to reduce the dataset by 75%.

In this work, as we are only interested in the compression part of the autoencoder (encoder) and not in the reconstruction of the data, the compress data would be the latent vector in Figure 2, and each gray box represents each convolutional layer. To reduce the data in each step, a MaxPooling is added between one convolutional layer and the next.

Following code shows the different configurations for each reduction, the tuple that is specified for each layer between brackets shows the output shape.

**1) 50% DIMENSIONALITY REDUCTION**

Next, the model is shown.

```
Model: ``24to12columns_encoder``
```

Layer (type)	Output shape	Param#
conv1d (Conv1D)	(None, 24, 2)	8
max_pooling1d (MaxPooling1D)	(None, 12, 2)	0
conv1d_1 (Conv1D)	(None, 12, 1)	7

```
=====
Total params: 15
Trainable params: 15
Non-trainable params: 0
```

```
Model: ``24to12columns_decoder``
```

Layer (type)	Output shape	Param#
conv1d_2 (Conv1D)	(None, 12, 2)	8
up_sampling1d (UpSampling1D)	(None, 24, 2)	0
conv1d_3 (Conv1D)	(None, 24, 1)	7

```
=====
Total params: 15
Trainable params: 15
Non-trainable params: 0
```

As we can see on the above code, for the encoder the input data has the size of 24 columns and it's passed through a convolutional 1-dimension layer, then is reduced in size to 12 columns using max pooling and passed to the last convolutional layer. On the other hand. the decoder receives the encoder output and passed it through a convolutional 1-dimension layer, then the dimensionality is increased up to 24 columns again using the up-sampling layer. Finally, the reconstructed data is given by the last convolutional layer which gets data from the previous up sampling layer.

### 2) 75% DIMENSIONALITY REDUCTION

The model is shown below

```
Model: ``24to6columns_encoder``
```

Layer (type)	Output shape	Param#
conv1d (Conv1D)	(None, 24, 1)	8
max_pooling1d (MaxPooling1D)	(None, 12, 2)	0
conv1d_1 (Conv1D)	(None, 12, 2)	14
max_pooling1d_1 (MaxPooling1D)	(None, 6, 2)	7

```
=====
Total params: 29
Trainable params: 29
Non-trainable params: 0
```

```
Model: ``24to6columns_decoder``
```

Layer (type)	Output shape	Param#
--------------	--------------	--------

conv1d_3 (Conv1D)	(None, 6, 2)	8
up_sampling1d (UpSampling1D)	(None, 12, 2)	0
conv1d_4 (Conv1D)	(None, 12, 2)	14
up_sampling1d_1 (UpSampling1D)	(None, 24, 2)	0
conv1d_5 (Conv1D)	(None, 24, 1)	7

```
=====
Total params: 29
Trainable params: 29
Non-trainable params: 0
```

In this case, the initial data is passed through a convolutional 1-dimension layer, then is reduced in size to 12 columns using max pooling, the same process is repeated passing the data to the convolutional 1-dimension layer and reducing the size using max pooling layer to have 6 columns. Finally, the output data from this layer is passed to the last convolutional layer. The decoder receives the encoder output and passed it through a convolutional 1-dimension layer then the dimensionality is increased up to 12 columns using the up-sampling layer. The previous process is repeated again with the same 2 layers to increase dimensionality to 24 columns and the reconstructed data is given by the last convolutional layer which gets data from the previous up sampling layer.

### 3) 90% DIMENSIONALITY REDUCTION

Following is the model.

```
Model: ``24to2columns_encoder``
```

Layer (type)	Output shape	Param#
conv1d (Conv1D)	(None, 24, 2)	8
max_pooling1d (MaxPooling1D)	(None, 12, 2)	0
conv1d_1 (Conv1D)	(None, 12, 2)	14
max_pooling1d_1 (MaxPooling 1D)	(None, 6, 2)	0
conv1d_2 (Conv1D)	(None, 6, 2)	0
max_pooling1d_2 (MaxPooling 1D)	(None, 2, 2)	0
conv1d_3 (Conv1D)	(None, 6, 2)	0

```
=====
Total params: 43
Trainable params: 43
Non-trainable params: 0
```

Model: ``24to2columns\_decoder``

Layer (type)	Output shape	Param#
conv1d_4 (Conv1D)	(None, 2, 2)	8
up_sampling1d (UpSampling1D)	(None, 6, 2)	0
conv1d_5 (Conv1D)	(None, 6, 2)	14
up_sampling1d_1 (UpSampling1D)	(None, 12, 2)	0
conv1d_6 (Conv1D)	(None, 12, 2)	14
up_sampling1d_2 (UpSampling1D)	(None, 24, 2)	0
conv1d_7 (Conv1D)	(None, 24, 1)	7

=====  
 Total params: 43  
 Trainable params: 43  
 Non-trainable params: 0

In this configuration, the 24 column input data is passed through a convolutional 1-dimension layer and then is reduced in size to 12 columns using max pooling. After, the same process is repeated passing the data to the convolutional 1-dimension layer and reducing the size using max pooling layer to have 6 columns. This data is passed through the next convolutional layer and the dimension is reduced to 2 columns using max pooling layer one more time. Finally, the output data from this layer is passed to the last convolutional layer.

In the second part, the decoder receives the encoder output and passed it through a convolutional 1-dimension layer and then the dimensionality is increased up to 6 columns using the up-sampling layer. The previous process is repeated again with the same 2 layers to increase dimensionality to 12 columns and repeated one last time to increase to 24 columns. Finally, the reconstructed data is given by the last convolutional layer which gets data from the previous up sampling layer.

4) ALGORITHM PERFORMANCE

The steps made by the algorithm for dimensionality reduction during the experimentation are the following:

- The original data is loaded and reviewed to ensure that there are not outliers that will compromise the learning of the model. The categorical column of values used for prediction is separated from the rest of the dataset to conserve the original values.
- For dimensionality reduction an autoencoder is created, which will be composed by two separate elements, the

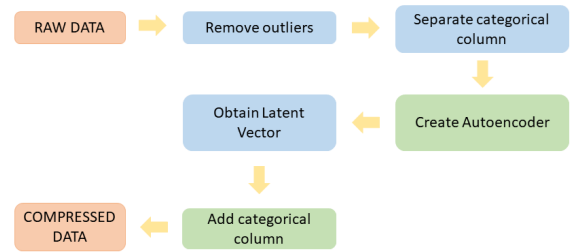


FIGURE 3. Dimensionality reduction steps diagram.

encoder as the input and the decoder as the output of the model. The model must be fitted by training data taken from the original one, to do it as optimal as possible the epochs and the batch size must be specified considering the training data length.

- The data is passed through the encoder to obtain the latent vector with dimensionality reduction. Then the categorical column will be attached to the latent vector in order to be able to make predictions with the new reduced data.

Figure 3 shows the procedure followed to reduce the dimensionality of a dataset.

V. APPLICATION SCENARIO

A. EXPERIMENTATION STEPS

To conduct the experiment, a public dataset from Kaggle was used, which was presented according to the industry paradigm [25]. This data provides information about a flotation plant, a method used to concentrate iron ore. The objective is to forecast the percentage of Silica in the final product, which is iron ore concentrate and its impurity (which is the % of Silica).

Data was collected every 20 seconds from hardware sensors such as temperature, pH, flow, density and all other continuous process variables, with no processing of the data (the dataset shows raw data). Quality indicators such as the percentage of silica in a sample, the percentage of iron ore in a sample, and so on are laboratory-based quality measurements. Approximately every 15 minutes, a sample of the iron ore pulp is taken. Those samples are forwarded to a laboratory for examination. Thus, each two hours, the laboratory provides feedback on the quality analysis; in other words, every two hours, you receive a lab/quality measurement of the product stream (iron ore concentrate), which provides insight into the product’s quality (iron ore pulp concentrate).

The main goal is to use this data to predict how much impurity is in the ore concentrate. As this impurity is measured every hour, if we can predict how much silica (impurity) is in the ore concentrate, we can help engineers, giving them early information to take actions. Hence, they will be able to take corrective actions in advance (reduce impurity, if it is the case) and also help the environment (reducing the amount of ore that goes to tailings as you reduce silica in the ore concentrate).

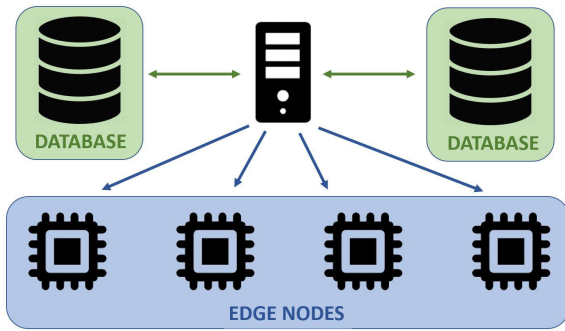


FIGURE 4. Proposed architecture.

Target is to predict the last column, which is the % of silica in the iron ore concentrate.

**B. ARCHITECTURE**

It is highly likely that the variables needed to predict the impurity in the gold concentrate are not all the collected variables. In addition, the rest of the variables could be used to generate other models of interest to the company, so we have considered an architecture that not only enables to predict the impurity, but also allows to use the data to create other different models. It is as follows in Figure 4.

Once the data has been gathered, it is saved in a database and utilized to build the prediction model. As illustrated in Figure 4, the architecture has been designed for the following purposes: On a large-scale computer server, the database’s data is gathered and Bayesian compression is conducted. Once the dataset has been reduced, it is sent to the little nodes spread throughout the plant. As a result, each node, referred to as a ‘Edge node,’ can develop unique models based on the data it wants to predict in a far more efficient and timely manner than if it had all the data in the raw.

**VI. EXPERIMENTAL EVALUATION**

**A. VALIDATION**

To validate the compression efficiency, the dataset was subjected to Bayesian compressions using a variety of different search methods and compression rates:

- Search algorithms: Hill Climbing and Tree Augmented Network (TAN).
- Compression rates: 25%, 50% y 75%.

*a: HILL CLIMBING*

Hill-climbing is a local optimization approach. It takes the steepest ascent/descent direction possible from your current position and incurs minimal computing cost. As it limits us from reverting to a previous choice, it is frequently referred to as an irreversible strategy [26].

*b: TREE AUGMENTED NETWORK (TAN)*

Tree-Augmented Naive Bayes is a semi-autonomous Bayesian learning technique. By utilizing a tree structure in which each attribute is dependent on only the class and

TABLE 1. Results of 25% and 50% compressed.

	Without compression Accuracy	25% compressed		50% compressed	
		HC	TAN	HC	TAN
Bayesnet K2	73.66%	73.46%	73.57%	73.13%	73.21%
BayesNet TAN	93.93%	93.08%	93.98%	93.80%	93.73%
J48	99.41%	97.86%	99.01%	98.59%	98.69%
KNN (5)	96.85%	91.55%	94.83%	93.95%	93.77%
Naive Bayes	52%	52.02%	52.10%	51.95%	52.05%
Random Forest	99.95%	99.46%	99.89%	99.83%	99.78%
Simple Logistic	59.36%	59.55%	59.45%	59.43%	59.37%

another attribute, it removes the naive Bayes attribute’s assumption of independence. Classification is carried out using a maximum amplitude-weighted tree, which optimizes the likelihood of the training data [27].

After designing models with compressed data, the accuracy of the trained models with compressed data was compared to the accuracy of the trained models with original data. Numerous models employing a variety of categorization techniques (Bayesian Network K2 [28], Bayesian Network TAN [29], J48 decision tree [30], K-Nearest Neighbor [31], Naive Bayes [32], Random Forest [33] and Simple Logistic Regression [34]) have been created in order to study the behavior of each.

**B. RESULTS - INSTANCES REDUCTION**

The experiment’s results are summarized in Table 1. On the left there are numerous algorithms for generating various prediction models. The first column shows the accuracy of each algorithm when trained on raw data models. The experiment was then divided into the three compressed sets indicated previously: 25%, 50%, and 75% compression of the raw data, respectively.

Each of these sets has two distinct models for each algorithm, as each one employs a separate search strategy: Hill Climber and Tree Augmented Network (TAN). When model accuracy results are compared, the results from compressed models are almost identical to those from raw data, but with up to 75% less size. The difference in accuracy is small, and in some circumstances, compressed data actually enhances accuracy. For example, when the Bayesnet TAN method was used to construct a model with a 25% compression of the data and the TAN search algorithm, the accuracy of the model was 93.98%, compared to 93.93% when the raw data was used. On either hand, the model constructed using 75% compressed data has a precision of 93.14%. This indicates that even after reducing the amounts of data by 75%, the model’s accuracy only varied by 0.79%.

Additionally, the Random Forest technique fits very well with the data reduction. With an accuracy of 99.95% when using raw data, the accuracy of compressed models is 99.89% at 25% compression, 99.83% at 50% compression, and 99.46% at 75% compression. The gap between the model

TABLE 2. Results of 75% compressed.

	Without compression	75% compressed	
	Accuracy	HC	TAN
Bayesnet K2	73.66%	72.29%	72.28%
BayesNet TAN	93.93%	93.13%	93.14%
J48	99.41%	97.80%	97.78%
KNN (5)	96.85%	91.48%	91.45%
Naive Bayes	52%	51.96%	51.94%
Random Forest	99.95%	99.46%	99.45%
Simple Logistic	59.36%	59.50%	59.45%

with raw data and the model with 75% compressed data in this situation is only 0.49%.

Additionally, the J48, KNN, and Bayesnet TAN algorithms all adapt extremely well to raw data, with 99.41%, 96.85%, and 93.93% accuracy, respectively.

In terms of compression, the method that loses the most accuracy is the KNN with a compression ratio of 25% on the raw data. Nevertheless, in this example, the Bayesnet TAN improves its accuracy by 0.05% when using compressed data.

One could argue that the loss is likewise negligible in these circumstances, given that the loss for the Bayesnet TAN and J48 methods is 0.13 and 0.72 for data compressed at 50% and 0.79 and 1.61 for data compressed at 75%, respectively.

Even while the remaining algorithms (Bayesnet K2, Naive Bayes, Simple Logistic) do not achieve such high accuracy with raw data (73.66%, 52%, 59.36%), the accuracy loss with compressed data is likewise negligible, with a maximum loss of 1.37% when compressed to 75% using Bayesnet K2.

When Bayesian compression is performed, the Hill Climbing or Tree Augmented network search algorithm is used for this purpose. The kind of search algorithm that produces the best results while compressing is determined by the compression percentage utilized. When it comes to 25% compression, the TAN is the search method that, in 85% of circumstances, compresses the data in a more representative manner than other algorithms. Nevertheless, when the compression ratio is 75%, the Hill Climber algorithm performs better in 85% of cases. If raw data is compressed to 50%, it is not always obvious which algorithm to employ, as Hill Climber produces better results in 57% of cases while TAN produces better results in 43% of cases.

C. RESULTS - DIMENSIONALITY REDUCTION

Dimensionality Reduction’s experiment results are shown in Table 3. As in the case of instances reduction, on the left side we have as a reference the accuracy of each algorithm with the raw data. In this case, the experiment was then divided into three compressed sets: 50%, 75% and 90%.

As shown in Table 3, the results are not as promising as those obtained with the reduction of instances 1 2.

Not only does the Naive Bayes algorithm not lose accuracy when compressed to 75%, it even improves accuracy by 0.98%. When the compression is 50%, the accuracy improves by 1.39%. After Naive Bayes, the algorithm that loses the

TABLE 3. Results of 50%, 75% and 90% compressed.

	Without compression	50% compressed	75% compressed	90% compressed
	Bayesnet K2	73.66%	57.62%	54.45%
Bayesnet TAN	93.93%	64.33%	62.91%	41.61%
J48	99.41%	86.66%	78.46%	40.91%
KNN (5)	96.85%	95.10%	83.10%	36.12%
Naive Bayes	52%	53.39%	52.98%	38.46%
Random Forest	99.95%	94.72%	85.00%	NA
Simple Logistic	59.36%	57.75%	57.57%	38.91%

least with a compression of 75% is the simple Logistic (1.79%), followed by KNN (13,75%).

The algorithms that best adapt to 50% compression are KNN, Naive Bayes and Simple Logistic. The KNN algorithm with 50% compression has a loss of only 1.75% accuracy. The Naive Bayes algorithm even improves the accuracy by 1.39% with the data compressed to 50%. Finally, Simple Logistic only have a loss of 1,61%.

As far as the 90% compression is concerned, it can be seen that the loss is not negligible, but to reduce the data set by 90% is quite acceptable.

VII. CONCLUSION

As described at the beginning of the paper, the objective and novelty of the article was to obtain and compare different compression techniques for datasets to be used to create Machine Learning models. In order to achieve the objective, we applied Bayesian compression and Autoencoders to the raw datasets to generate tiny edge computing models. As a result, as demonstrated, we may develop models that can be implemented in nodes with limited computational capacity due to the reduced size of the dataset, while still producing comparable results to the original dataset. Additionally, it was discovered that in some situations, the compressed data set produced more accurate findings than the original dataset when specific machine learning algorithms were applied to it.

It must be highlighted that when Bayesnet with the TAN search algorithm is applied to reduce instances or Naive Bayes is used with autoencoder’s compressed data (50%), even a small amount of compression is performed in the case of Bayesnet, such as 25%, the model’s accuracy can even improve (0,05% accuracy improvement for the Bayesnet compressed data and 1,39% for data compressed with autoencoder).

In the case of Bayesian compression and the use of Bayesnet algorithm, while other algorithms may lose some accuracy when dealing with compressed data, this loss can be regarded insignificant in this case, even when compression is set to 75%.

On the other hand, when using the autoencoder for compression, it has been concluded that it works very well for very specific cases. That is when using the up sampling layer for decoder reconstruction, in order to reduce the dimensionality to the 50%. But it gave bad results when adding zero padding



layers for reconstruction because it adds multiple zeros to the output distorting the result, not letting the granularity as good as Bayesian compression.

In addition, the methodology proposed in this paper allows the datasets to be compressed into the desired percentage, achieving an ad-hoc methodology for each use case.

As future work, we really believe it would be interesting to combine both techniques on the same dataset in order to reduce both the number of instances and the number of dimensions, or at least to find a different technique that combines both reductions.

In analyzing a compression process, it is essential to consider both energy efficiency and time efficiency to identify areas for improvement. Energy efficiency is a key factor in reducing costs and achieving environmental sustainability, while time efficiency is important for maximizing productivity and minimizing downtime. By analyzing both aspects, it is possible to detect potential bottlenecks in the compression process and find opportunities to optimize performance. A thorough analysis of energy and time efficiency in the compression process can be crucial in achieving a more efficient and cost-effective process in the future.

## REFERENCES

- [1] V. Gopalkrishnan, D. Steier, H. Lewis, and J. Guszczka, "Big data, big business: Bridging the gap," in *Proc. 1st Int. Workshop Big Data, Streams Heterogeneous Source Mining, Algorithms, Syst., Program. Models Appl.*, 2012, pp. 7–11.
- [2] L. Thames and D. Schaefer, "Software-defined cloud manufacturing for Industry 4.0," *Proc. CIRP*, vol. 52, pp. 12–17, Jan. 2016.
- [3] H. Lasi, P. Fettke, H. G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," *Bus. Inf. Syst. Eng.*, vol. 6, no. 4, pp. 239–242, 2014.
- [4] D. R. Wilson and T. R. Martinez, "Instance pruning techniques," in *Proc. ICML*, 1997, pp. 400–411.
- [5] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 2, 1989, pp. 1–8.
- [6] E. Nalisnick, A. Anandkumar, and P. Smyth, "A scale mixture perspective of multiplicative noise in neural networks," 2015, *arXiv:1506.03208*.
- [7] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1613–1622.
- [8] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2498–2507.
- [9] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, Jun. 2016, vol. 44, no. 3, pp. 243–254.
- [10] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.
- [11] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2849–2858.
- [12] X. Li, T. Zhang, X. Zhao, and Z. Yi, "Guided autoencoder for dimensionality reduction of pedestrian features," *Int. J. Speech Technol.*, vol. 50, no. 12, pp. 4557–4567, Dec. 2020.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012.
- [14] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with Gaussian mixture variational autoencoders," 2016, *arXiv:1611.02648*.
- [15] L. Fang, T. Zeng, C. Liu, L. Bo, W. Dong, and C. Chen, "Transformer-based conditional variational autoencoder for controllable story generation," 2021, *arXiv:2101.00828*.
- [16] J. E. Fowler and R. Yagel, "Lossless compression of volume data," in *Proc. Symp. Volume Visualizat.*, 1994, pp. 43–50.
- [17] V. V. Pirozhenko and V. D. Grigoriev, "Video stream processing and compression with codec choice ability," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2020, pp. 455–457.
- [18] U. Sarwar, "Real time multiple codecs switching architecture for video conferencing," Ph.D. dissertation, Univ. Sains Malaysia, 2008.
- [19] I. H. Putro, "Performance of various codecs related to jitter buffer variation in VoIP using SIP," *J. Teknik Elektro*, vol. 8, no. 2, pp. 103–108, Apr. 2010.
- [20] L. Allison, A. S. Konagurthu, and D. F. Schmidt, "On universal codes for integers: Wallace tree, Elias omega and beyond," in *Proc. Data Compress. Conf. (DCC)*, Mar. 2021, pp. 313–322.
- [21] P. Fenwick, "Universal codes," in *Lossless Compression Handbook*. 2003, pp. 55–78.
- [22] K. M. Leung, "Naive Bayesian classifier," *Polytech. Univ. Dept. Comput. Sci./Finance Risk Eng.*, vol. 2007, pp. 123–156, Nov. 2007.
- [23] L. Sucar, "Clasificadores bayesianos: De datos a conceptos," in *Proc. Eur. Conf. Mach. Learn. Princ. Pract. Knowl. Discov. Databases*, 2008, pp. 1–18.
- [24] Y. Zhang. (2018). *A Better Autoencoder for Image: Convolutional Autoencoder*. Accessed: Mar. 23, 2017. [Online]. Available: [http://users.cecs.anu.edu.au/Tom.Gedeon/conf/ABCs2018/paper/ABCs2018\\_paper\\_58](http://users.cecs.anu.edu.au/Tom.Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58)
- [25] *Quality Prediction in a Mining Process*. Accessed: Sep. 30, 2010. [Online]. Available: <https://www.kaggle.com/edumagalhaes/quality-prediction-in-a-mining-process>
- [26] T. J. Rezek, R. G. R. Camacho, N. M. Filho, and E. J. Limacher, "Design of a hydrokinetic turbine diffuser based on optimization and computational fluid dynamics," *Appl. Ocean Res.*, vol. 107, Feb. 2021, Art. no. 102484.
- [27] F. Zheng and G. I. Webb, "Tree augmented Naive Bayes," *Tech. Rep.*, 2010.
- [28] B. Lerner and R. Malka, "Investigation of the K2 algorithm in learning Bayesian network classifiers," *Appl. Artif. Intell.*, vol. 25, no. 1, pp. 74–96, Jan. 2011.
- [29] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Mach. Learn.*, vol. 29, no. 2, pp. 131–163, Nov. 1997.
- [30] N. Bhargava, G. Sharma, R. Bhargava, and M. Mathuria, "Decision tree analysis on J48 algorithm for data mining," *Proc. Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 6, 2013.
- [31] L. Kozma, "K nearest neighbors algorithm (KNN)," *Helsinki Univ. Technol.*, vol. 32, Feb. 2008.
- [32] G. Kaur and E. N. Oberai, "A review article on Naive Bayes classifier with various smoothing techniques," *Int. J. Comput. Sci. Mobile Comput.*, vol. 3, no. 10, pp. 864–868, 2014.
- [33] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble machine learning*. Berlin, Germany: Springer, 2012, pp. 157–175.
- [34] T. G. Nick and K. M. Campbell, "Logistic regression," in *Topics in Biostatistics*. 2007, pp. 273–301.



**NEREA GÓMEZ LARRAKOETXEA** received the degree in computer engineering and the master's degree in automation, electronics, and industrial control from Deusto University, in 2016 and 2017, respectively.

Since 2015, she has been involved in research projects with DeustoTech, where she started as an Intern. During this time, she has been working on big data projects, applying artificial intelligence techniques to large datasets. She is currently working on a doctoral thesis of industrial nature and framed within the new paradigm of Industry 4.0, with edge computing being her principal research area.



**JOSEBA ESKUBI ASTOBIZA** received the degree in computer engineering and the Master of Computer Engineering degree from the University of Deusto, in 2018 and 2020, respectively.

In 2015, he received a Junior Researcher Scholarship for which he became part of Deusto eVida in 2018. During this period, he developed IOS applications that served to offer support and help in the education of people with Down Syndrome through games, classified as serious games. During

the master's study, he was a part of the Vidrala I+D department in a project in collaboration with DeustoTech related to artificial intelligence applied to the world of industry. In addition, he worked for a short period of time in a consultancy and advised a start-up in relation to data analysis.



**IKER PASTOR LÓPEZ** received the degree in computer engineering, in 2007, the master's degree in information security, in 2010, and the Ph.D. degree (cum laude) in computer science, in 2013. He participated in the Program in Big Data and Business Intelligence, in 2016.

He is with Deusto University and focuses its scientific interests on the areas of big data analytics, opinion mining, and computer vision. He is the author of several scientific articles reviewed

by peers in conferences and indexed journals. He has participated in the gestation, scientific development, and technical development of numerous competitive projects and contracts with companies, the latter with several successful cases of knowledge transfer actions. In addition, he is a member of the Scientific Committee of several congresses, such as CISIS, SOCO, and ICEUTE. He is a Reviewer of many journals, included in the JCR as the magazine of *Engineering and Industry-DYNA*.



**BORJA SANZ URQUIJO** received the Ph.D. degree (cum laude) in information systems on malware detection in android mobile devices, in 2012.

He has been a Researcher with the DeustoTech-Computing Research Unit, since 2008, where he was the Head Researcher, from 2015 to 2018. He is currently a Lecturer with the Faculty of Engineering, University of Deusto. His primary research interests include machine learning, big

data, knowledge discovery, and information retrieval. He is working on fairness, ethics, accountability, and the impact of artificial intelligence in society. He has a long track record leading national and international projects in the research areas previously mentioned. He has worked closely with different social agents, enterprises, and other research centers. He has been a researcher in more than 50, between H2020, national and private projects and a project manager in several of them. He has published several book chapters and more than 60 articles in specialized national and international impact journals, such as *Logic Journal of IGPL*, *Electronic Commerce Research and Applications*, and *Expert Systems with Applications*.



**JON GARCÍA BARRUETABEÑA** has begun the research activity with the University of Mondragón, where he developed his doctoral research in the dynamic behavior of complex materials, both from the analytical, numerical, and experimental points of view. From 2011 to 2013, he signed a contract with Ikerlan Ik4, where he worked on large research projects and contracts with companies mainly in the noise and vibration area. In 2013, he was offered a permanent position

with the University of Deusto to launch the Applied Mechanics Research Group and to promote research and teaching activity in the field of automation. He was the principal researcher of the Applied Mechanics Research Group for two years. He is accredited by ANECA as a Private University Doctoral Professor in 2013 and the Hired Doctoral Professor of mechanical engineering, in 2013, and has two six-year research periods, from 2008 to 2013 and from 2015 to 2020 tranche respectively, as well as a six-year transfer period, from 2013 to 2018 tranche.



**AGUSTIN ZUBILLAGA REGO** received the degree in computer engineering, the degree in political science and public administration, and the Ph.D. degree (cum laude) in computer and telecommunications engineering, in 2015, in artificial intelligence applied in the political science field.

He was a Researcher with DeustoTech, from 2009 to 2016, promoting and participating in several technological research projects at both national and international levels. Later, he led the Digital Economy Laboratory, Orkestra-Basque Competitiveness Institute, achieving a profound track in the digital economy as well as innovation and transformation at the SME level for policymakers and business leaders. In 2022, he has started to lead Bilbao Office, Vicomtech-Applied Technology Research Center, Basque Country. He is a lecturer in four universities in the north of Spain and advises several governments on digital policies.

...