

# AI<sup>2</sup>: The next leap toward native language based and explainable machine learning framework

Jean-Sébastien Dessureault, Daniel Massicotte

January 16, 2023

## ABSTRACT

The machine learning frameworks flourished in the last decades, allowing artificial intelligence to get out of academic circles to be applied to enterprise domains. This field has significantly advanced, but there is still some meaningful improvement to reach the subsequent expectations. The proposed framework, named AI<sup>2</sup>, uses a natural language interface that allows a non-specialist to benefit from machine learning algorithms without necessarily knowing how to program with a programming language. The primary contribution of the AI<sup>2</sup> framework allows a user to call the machine learning algorithms in English, making its interface usage easier. The second contribution is greenhouse gas (GHG) awareness. It has some strategies to evaluate the GHG generated by the algorithm to be called and to propose alternatives to find a solution without executing the energy-intensive algorithm. Another contribution is a preprocessing module that helps to describe and to load data properly. Using an English text-based chatbot, this module guides the user to define every dataset so that it can be described, normalized, loaded and divided appropriately. The last contribution of this paper is about explainability. For decades, the scientific community has known that machine learning algorithms imply the famous black-box problem. Traditional machine learning methods convert an input into an output without being able to justify this result. The proposed framework explains the algorithm's process with the proper texts, graphics and tables. The results, declined in five cases, present usage applications from the user's English command to the explained output. Ultimately, the AI<sup>2</sup> framework represents the next leap toward native language-based, human-oriented concerns about machine learning framework.

Keywords: *machine learning; framework; NLP; AI ethics; explainability*

## 1 Introduction

Two decades ago, some popular algorithms existed and were well documented in scientific literacy, but there was still no easy way to use them. Scientists had to read the equations and the algorithm before implementing it in the desired programming language. Every matrix had to be multiplied, and every derivative had to be computed by the scientist's code. In the last two decades, machine learning has finally flourished. One of the most meaningful frameworks was certainly TensorFlow [1]. This powerful tool helped the community accelerate development and democratize the machine

learning field. It helped this field of knowledge reach a more comprehensive range of applicative projects instead of being restricted to academics.

A few years after the first version of Tensorflow, many others came to the machine learning community. Among the most popular: Scikit-Learn, CNTK, Torch, Matlab, and Keras [39]. In the last few years, a user-friendly framework with a graphical interface named Orange [8] became available, aiming to be even more accessible for the community, especially for the non-expert. While consistently more accessible over time, requiring less mathematics and fewer programming skills, none of those frameworks has made the ultimate step: the ability to communicate in the native human language.

Some recent studies compare the most popular machine learning software framework. For instance, framework performances have been recently analysed in [39]. For this same purpose of performance analysis, [38] divides frameworks into some topics (computational distribution, Tensor Processing Units and Field-Programmable Gate Array (FPGAs)). [42] compares machine learning frameworks on different hardware platforms, such as Raspberry Pi 3 B+, NVIDIA Jetson, MacBook Pro, Huawei Nexus 6P and Intel FogNode.

Nguyen et al. in [27] have an essential paper regarding this current research. Their work establishes evaluation criteria for supervised, unsupervised, and reinforcement learning, which are the three prominent families of machine learning. [27] presents an overview of machine learning frameworks and gives the advantages and disadvantages of each. Frameworks are applied in different domains. For instance, [30] applies it to the Automated Detection of Arrhythmias in ECG Segments, while [26] is a framework application in the health domain for smart patient monitoring and recommendation. The work of [25][3] present and compares explainable and interpretable frameworks.

This framework, called AI<sup>2</sup>, proposes a natural language interface. To the authors' best knowledge, there is no machine learning framework offering an Natural language Processing (NLP) interface using a chatbot. This first AI<sup>2</sup> version proposes an English chatbot, but some other native languages might be proposed later. The NLP domain has flourished recently, especially when using the Transformers technology [33] [37]. This recent NLP breakthrough created the opportunity to fill the last gap between humans and machine learning frameworks: the ability to communicate in the native human language. This last step has just been done with this proposed AI<sup>2</sup> framework.

A state-of-the-art, Transformer-based NLP agent can now correctly interpret users' English requests. Outperforming older methods like *Recurrent Neural Networks* (RNN) [18] [35] and *Artificial Intelligence Markup Language* (AIML) [24], Transformer technology [19] delivers better results. Transformer-based applications exist in multiple domains. For instance, [23] uses it for sentiment analysis. [16] evaluates a Transformer's ability to learn Italian syntax. Finally, [6] proposes a chatbot that helps detect and classify fraud in a finance context.

Bidirectional Encoder Representation from Transformers (BERT) [14][32] [2] is a widely used NLP model. It performs exceptionally well when evaluating the context and understanding the intent behind the user's query [28].

Using the BERT NLP model, two pre-trained datasets have been used to build the AI<sup>2</sup> framework. The first one, BERT (BERT-large), is helpful to answer common questions like "Which dataset has been used?". The second one is RoBERTa (roberta-large) [21]. It is only used to answer Yes/No questions like "Is it a clustering problem?". Besides launching the requests, a minor contribution of AI<sup>2</sup> is its ability to preprocess the datasets using its NLP chatbot.

Even if the NLP interface is the main contribution of this paper, other contributions are also proposed. For instance, another contribution of the AI<sup>2</sup> framework is the awareness of greenhouse

gases (GHG). *CodeCarbon* [22] recently proposed a library of functions about GHG awareness and AI<sup>2</sup> integrates some of those functions and enhances it with machine learning methods. Based on [29], explainability is an essential contribution of this proposed framework. It aims to include ethics principles from the Institute for Ethical AI & Machine Learning [20]. This UK-based research centre develops frameworks that support the responsible development, deployment and operation of machine learning systems.

*Explainability* is a concept intending to eliminate the "black box" problem. Yoshua Bengio has addressed it, and Judea Perl [15], two Turing awards winners. Over the last decade, ML has reached a certain level of maturity. One of the differences is our expectations of machine learning. There is a need to democratize the methods to non-expert users. Until recently, the scientific community was concerned about lowering the error when using ML algorithms. They were concerned about the performance. Now the expectation is higher. The community still wants good results, but those results have to be found in an explainable, interpretable and ethical context. Human well-being must be the main interest of the ML systems. The results must be explainable. For decades, the "black box" problem was neglected. Now, there are some methods to explain the results and make them understandable to a human. The expectations are also higher regarding the accessibility to the ML methods, GHG awareness and preprocessing. Now, the expectations are higher at different levels. Disposing of the previously presented technologies and based on [13], the contributions of this framework aim to reach expectations with the following targets: 1. democratizing ML frameworks using NLP methods, 2. being GHG aware with a built-in structure to monitor it, 3. being more ethics with a built-in structure systematically explain the results, and 4. having the preprocessing of the data more accessible with an automated NLP based chatbot.

The following sections of this paper are organized with the following structure: Section 2 describes the proposed methodology. Section 3 presents the results. Section 4 discusses the results and their meaning, and Section 5 concludes this research.

## 2 Methodology of the AI<sup>2</sup> framework

### 2.1 Architecture

Fig. 1 presents the architecture of the AI<sup>2</sup> framework. The NLP method, through a chatbot, allows communication with the framework methods and the data using the English language. The kernel of the AI<sup>2</sup> framework includes four types of methods: 1. Preprocessing methods, 2. Machine learning methods, 3. GHG methods, and 4. Explainability methods.

The preprocessing interface method is done systematically once for each dataset when used for the first time. The chatbot guides this user throughout the process. It consists of a series of questions to the user about the dataset and each feature/class. The chatbot asks about the type of each field and its normalization method. The machine learning methods are the classic supervised and unsupervised learning methods: classifiers, regressors, clustering, dimensionality reduction and a method to evaluate the importance of the features. There are also some new methods like *Decision Process for Dimensionality Reduction* (DPDR) [10], *Decision Process for Dimensionality Reduction before Clustering* DPDRC [12], and CK-Means [11]. There are also some functions assuring the GHG awareness of the framework. Based on the CodeCarbon library [22], those functions compute the generated GHG for each request. Before launching a request, the GHG functions will predict the GHG generated for this request. They will try to find equivalent requests using clustering methods to save the execution of the subsequent request, thus, saving the generation of GHG.

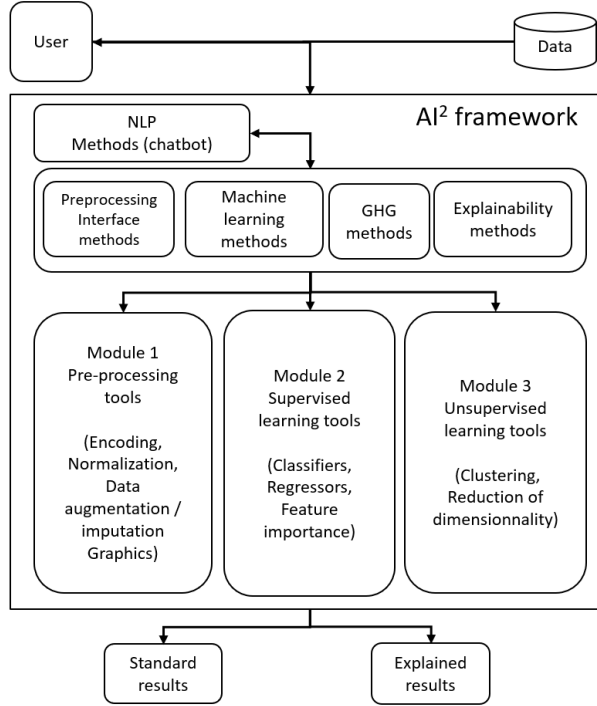


Figure 1: Architecture of the AI<sup>2</sup> framework.

The explainability methods offer a complement to the standard machine learning results. The user gets more than the expected results for his request. He gets a well-documented explanation for every result. The form of the explanation varies according to the used algorithm and data. Some examples (like learning curves and the importance of features graphic) are described in the use cases. The different machine learning methods are divided into three modules. Module 1 includes the preprocessing tools (Encoding, normalization, data augmentation/imputation, graphics). Module 2 consists of the supervised learning tools (classifiers, regressors and the computation of the feature's importance). Finally, module 3 exploits the unsupervised learning tools (clustering and reduction of dimensionality methods). At last, all the results are given in 2 forms: the expected and the explained results. AI<sup>2</sup>'s functions can be called without using its NLP interface. Calling the Python function directly without using the English chatbot is very straightforward. The user is responsible for obtaining his own datasets. No sample dataset is included in this first version of AI<sup>2</sup>.

## 2.2 NLP methods (chatbot)

This machine learning framework is its ability to communicate with a user, exploiting a chatbot based on NLP. The chatbot used by the AI<sup>2</sup> user interface is made with the Transformers technology, thus, being a state-of-the-art NLP model. In the AI<sup>2</sup> context, the Transformer technology is used with the "BERT" technology. Fig. 2 presents the NLP architecture of the AI<sup>2</sup> framework.

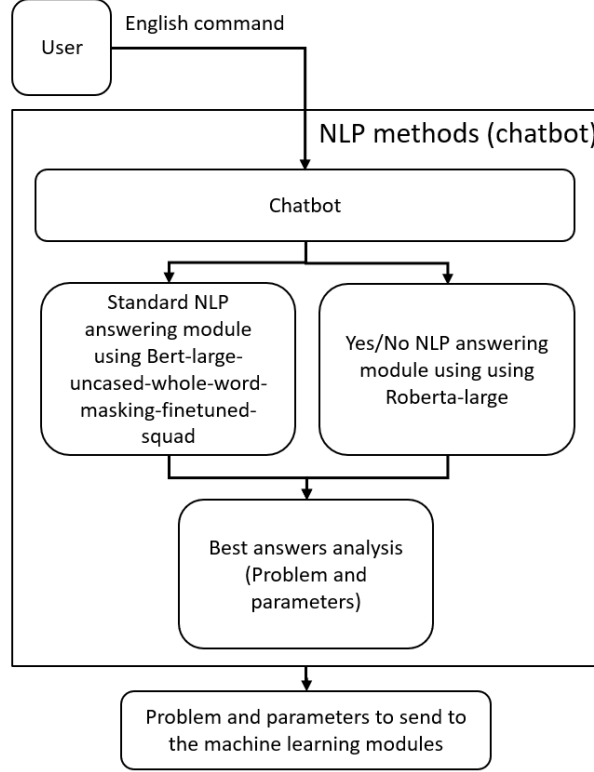


Figure 2: Architecture of the NLP interface in the AI<sup>2</sup> framework.

The chatbot uses two types of questions, requiring two different types of NLP pre-trained data. It is essential to note the difference between the datasets that can be processed by the AI’s methods and those NLP-based pre-trained datasets used by the chatbot. *The Standard NLP answering module using Bert-large-uncased-whole-word-masking-finetuned-squad* can help in responding to open questions like: *What is the dataset?*. As displayed in Table 1, this question is associated with the *DATASET* key. The chatbot will try every question having this key to filling out the dataset information. A typical answer to this request can be *iris*, for the *iris* dataset. The pre-trained dataset *Bert-large-uncased-whole-word-masking-finetuned-squad* [14] is used to answer this type of question. It is a pretrained model on English language using a Masked Language Modeling (MLM) objective.

Using the *Roberta-large* [21] pre-trained dataset, the second type of question is the Yes/No question. A typical question would be *Is this a clustering problem?*. The two possible answers are Yes and No, both associated with a certain level of confidence. As presented in Table 1, this question is associated with the *PROBLEM* key and the *CLUSTERING* return value. If the answer to this question is Yes, it will return *CLUSTERING* as an answer to fill out the information.

As mentioned earlier, every question related to the key is asked in these two types. The NLP system returns an answer for each question and confidence level. The answer related to the best level of confidence is kept. The methodology used to train both pretrained datasets, including the

level of confidence formulas are documented in [14] and [21]. It consists of applying a softmax function on the logits values. The logits variable is known to be the output of a BERT-based Transformer. It is a list of the most probable answers.

The following describes how the chatbot works. The chatbot first asks **"Please, enter your English command to the framework"**. The system specifies writing the English command to avoid confounding with a specific programming language-based command used in other frameworks. The expected command is the English instruction to the AI<sup>2</sup> framework. A typical command could be **"I want to perform a clustering using 3 clusters on the iris dataset."**. From this first answer from AI<sup>2</sup>, the chatbot will read a Parameters.csv file storing the structure of the required keys, the returned values and the questions to send to the chatbot to access the information. There is no specific order for the keys in this file. The system will request the keys to get the related information. For now, there are 73 rows defined in this file. Those rows designate 19 keys and the questions to access them. Many questions may retrieve each key. It is essential to understand that the framework uses those questions to extract pieces of information from the user command. Those questions are entirely transparent for the users. This file will grow following the new releases of the AI<sup>2</sup> framework. Table 1 presents a sample of this file. *Key* field identifies the information to retrieve. For instance, if AI<sup>2</sup> seeks the type of problem in the user's command, it will find all the *PROBLEM* rows. It will then interrogate the user's command with all the corresponding *Questions* field. It will keep the answer having to higher level of confidence according to the Transformer. The answer to the question will be returned, except if it is a Yes/No question. In this case, the *Return value* field will be used. For instance, if the AI<sup>2</sup> system replies *Yes* to the question *Is this a clustering problem?* then the returned value will be *CLUSTERING*. The *Type* field indicates *Y/N* for *Yes/No* questions and *Std.* for *standard* questions.

Table 1: Sample of the Parameters.csv file. Only data used for this example is presented (14 rows on a total of 73).

Key	Type	Return value	Questions to the command
PROBLEM	Y/N	DIMENSIONALITY	Is this about dimensionality?
PROBLEM	Y/N	DIMENSIONALITY	Is this about dimensionality reduction?
PROBLEM	Y/N	CLASSIFICATION	Is this about classification?
PROBLEM	Y/N	CLASSIFICATION	Is this a classification problem?
...	...	...	...
PROBLEM	Y/N	CLUSTERING	Is this clustering?
PROBLEM	Y/N	CLUSTERING	Is this a clustering problem?
PROBLEM	Y/N	CLUSTERING	Is this regrouping?
PROBLEM	Y/N	CLUSTERING	Is this a regrouping problem?
PROBLEM	Y/N	CLUSTERING	Do you want to regroup data?
PROBLEM	Y/N	CLUSTERING	Do you want to cluster data?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?
NB_CLST	Std.		How many groups?
NB_CLST	Std.		How many clusters?
...	...	...	...

Systematically, the chatbot will try to fill the *PROBLEM* key. It must know what kind of

problem it is. To find it out, a question list corresponding to the *PROBLEM* key, is processed by the chatbot. If the chatbot can return the answer, the *problem information* (corresponding to *PROBLEM* in the *Key* field, Table 1) will be filled. If the first question can return no answer, other questions (corresponding to the key) will be tried to extract information. If no answer can be found after having tried all the questions, the chatbot will prompt to directly ask the user: **problem to resolve has been found in your text. Please clearly identify the type of problem to solve.** Then, the algorithm will go to the second and the third required keys: the *DATASET* key and the *NB\_CLST* (number of clusters) key. The interface will ask for every crucial information. When the parameter is not mandatory, its default value will be assumed. The same principle is repeated for every required parameter. An example of a complete sequence is illustrated in Table 2. Remember that the questions are not directly addressed to the user but to his command, aiming to extract meaningful information to execute his request.

Table 2: Example of a typical command and the question sequence used to extract the information of the command: *I want to perform a clustering using 3 clusters on the iris dataset.*

Questions	Answer	Ret. value
(To extract the type of the problem)		
Is this about dimensionality?	No	None
Is this about dimensionality reduction?	No	None
Is this about classification?	No	None
Is this a classification problem?	No	None
Is this clustering?	Yes	CLUSTERING
(To extract the name of the dataset)		
What is the dataset?	Iris	Iris
(To extract the number of clusters)		
How many groups?	(No suitable answer)	None
How many clusters?	3	3

In this example, the answer is *No* for the first four questions, since the command is not about reduction of dimensionality nor classification. Since the command is about a clustering problem, the answer will be *Yes* to the question *Is this clustering?*. Since the value *Yes* would not mean anything, the corresponding return value (Table 1) *CLUSTERING* is returned. After having extracted the problem type, the dataset name is required. The question *What is the dataset?* answer the question. The answer is *Iris*, and the returned value is also *Iris*. The last required information is about the number of clusters needed for the clustering algorithm. There are at least two ways of asking this question since groups and clusters are synonyms. The question *How many groups?* is tried to extract the information. Since the command uses the term *clusters*, no suitable answer is found for this question. The second question will be: *How many clusters?*. The answer and the returned value will be *3*. From this point, AI<sup>2</sup> has all the required information to launch a clustering algorithm using the Iris dataset and 3 clusters. Some more complete examples are shown in Section 3.

## 2.3 Preprocess module

The preprocessing method is done systematically once for each dataset when used for the first time. AI<sup>2</sup> detects when no dataset configuration has been done and stored in a JSON file. The chatbot

then asks for the correct configuration for every field, like their name, role in the dataset, and normalization methods. In the end, the dataset's configuration is stored in a JSON file, and the dataset is preprocessed and stored using the same file name, added with a `_preprocessed` suffix. The chatbot finally asks the user if he wants to process a data imputation of the missing data and a data augmentation.

Fig. 3 presents the functionalities of the preprocess modules. First, a dataset name is given to the module. If a preprocessed version of the dataset already exists, the module will open it, dividing it into train and test data. If the preprocessed files do not exist, the system will try to find the corresponding JSON file. If the JSON file exists, the system will use it to build the preprocessed file. If it does not exist, the AI<sup>2</sup> chatbot will guide the user through some questions about the field and create the final JSON file containing the structure of the dataset, and it will create the preprocessed dataset from this JSON file. Ultimately, it will also split the data into train and test data.

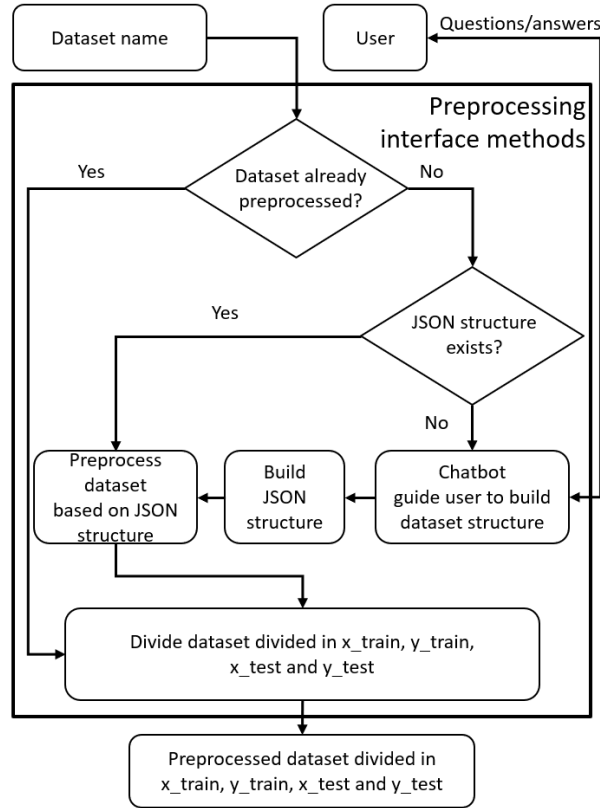


Figure 3: Preprocessing architecture

Fig. 4 shows an example of a structure configuration JSON file. The included fields in JSON format are the following: *dataset\_name* is the name of the dataset. *dataset\_description* is a description of the dataset. *feat\_no* is the number of the feature. *feat\_label* is the label given to this feature. The type of the feature is given by *feat\_type*. Possible values are 1. Feature field, 2.



```

iris.json
{
  "dataset_name": "iris",
  "dataset_description": "iris dataset",
  "feat_no": [
    0,
    ...
  ],
  "feat_label": [
    "Petal length in cm",
    ...
  ],
  "feat_type": [
    "1",
    ...
  ],
  "feat_normalization": [
    "1",
    ...
  ]
}

```

Figure 4: iris.json structure file

Regression value field, 3. Class field, and 4. Class for neural network field (to be one-hot encoded). The last field is *feat\_normalization*. Possible values are 1. No normalization, and 2. MinMax normalization.

Fig. 5 shows an example of an exchange between the chatbot and the user, aiming to preprocess the data.

## 2.4 Machine learning methods

Any framework requires a tremendous amount of development hours. This framework is still in development, yet it has some contributions to bring to the scientific community. Some known algorithms are included, resolving most machine learning problems (prediction, classification, and others). Table 3 shows algorithms included in AI<sup>2</sup>.

```

Let us preprocess the iris dataset. Please, answer the
following questions:
What is the description of the iris dataset (ENTER to skip)?
>This dataset describes the features and the class of the iris
dataset.

What is the name of the field 0? (Value example: 5.1)
>Sepal length in cm

What is the type of field Sepal length in cm? (1. Feature 2.
Predicted value 3. Class 4. Class (to be converted ONE-HOT
for neural network)
>1

What is the normalization applied to Sepal length in cm?
(1. None 2. MinMax)
>1

(... And so on for each feature and class.)

Saving dataset configuration...
The configuration is saved to iris.json
Processing to the file conversion...
The configuration is saved to iris_preprocessed.csv

```

Figure 5: An example of the exchange between the chatbot and the user for the data preprocessing.

Table 3: Machine learning algorithms are included in the AI<sup>2</sup> framework.

No.	Modules	Algorithms
1.	Pre-processing	IQR
2.		SMOTE
3.		KNNImputer
4.		xGEWFI metric
5.	Supervised learning	Neural network regressor
6.		Neural network classifier
7.		Random Forest
8.	Unsupervised learning	K-Means
9.		CK-Means
10.		Silhouette metric
11.		PCA
12.		DPDRC
13.		DPDR
14.		FRSD

The pre-processing methods (Module 1) are regrouped into one callable function. This function can do the whole process of finding the outliers, augmenting the data and imputing the missing data. The recent explainable metric named xGEWFI [9] is used to evaluate the performance of the data generation (imputation and augmentation). It considers the importance of the feature and each feature error to evaluate the global error of the data generation process. Inter Quartile Range (IQR) algorithm is used to find the outliers. Data generation (augmentation and imputation of missing data) are made with a SMOTE algorithm [7] and a KNNImputer [36], respectively.

Some neural networks (multilayer perceptron doing regressions and classifications) [31] are available for supervised learning functions (Module 2). A Random Forest (RF) algorithm [5] is used as a classifier and regressor. It is also used to evaluate the importance of the features.

Some unsupervised learning methods (Module 3) are also available. The K-means algorithm [4] can be executed for clustering problems. The CK-Means algorithm [11] can be called to extract data from the cluster's intersection. The metric to evaluate the cluster consistencies of those first two algorithms is the Silhouette Index (SI) [34]. Concerning the dimensionality reduction, the Principal Component Analysis (PCA) algorithm [17] is included in the AI<sup>2</sup> framework. Two new decision processes are also included to help with the dimensionality reduction problems. 1. Decision Process for Dimensionality Reduction before Clustering (DPDRC) [12] and 2. Decision Process for Dimensionality Reduction (DPDR) [10]. Those two are used in unsupervised learning and supervised learning contexts, respectively. In an unsupervised learning context, Feature Ranking Process Based on Silhouette Decomposition (FRSD) [40] helps evaluate the importance of the features.

## 2.5 GHG Methods - CodeCarbon integration in AI<sup>2</sup>

Climate change is an essential issue for humanity. It is our responsibility to be aware of it and to do everything that can be done to contribute to lower GHG. We know that computer sciences, particularly machine learning, can significantly generate GHG while executing on CPU and GPU. The CodeCarbon library is an important initiative available to data scientists, so they can be aware of their impact on GHG. The following quote can be found on the CodeCarbon website (at [pypi.org/project/codecarbon/](https://pypi.org/project/codecarbon/)) based on [22]: *While computing currently represents roughly 0.5% of the world's energy consumption, that percentage is projected to grow beyond 2% in the coming years, which will entail a significant rise in global CO2 emissions if not done properly. Given this increase, it is important to quantify and track the extent and origin of this energy usage, and to minimize the emissions incurred as much as possible. For this purpose, we created CodeCarbon, a Python package for tracking the carbon emissions produced by various kinds of computer programs, from straightforward algorithms to deep neural networks. By taking into account your computing infrastructure, location, usage and running time, CodeCarbon can provide an estimate of how much CO2 you produced, and give you some comparisons with common modes of transportation to give you an order of magnitude.* The contribution of this paper is to embed this library's features in a machine learning framework, add some machine learning-based functions to predict the subsequent request amount of GHG, and try to spare its execution by proposing some alternatives. Fig. 6 explains those embedded GHG functionalities.

First, every GHG statistic (request name, machine learning algorithm used, dataset, number of data, fields, elapsed time, GHG emissions) is stored in a file. When a user is about to launch a new request, from this stored historic, AI<sup>2</sup> framework will try to predict the amount of GHG this subsequent request will generate. A multilayer perceptron (MLP) is used to evaluate this GHG

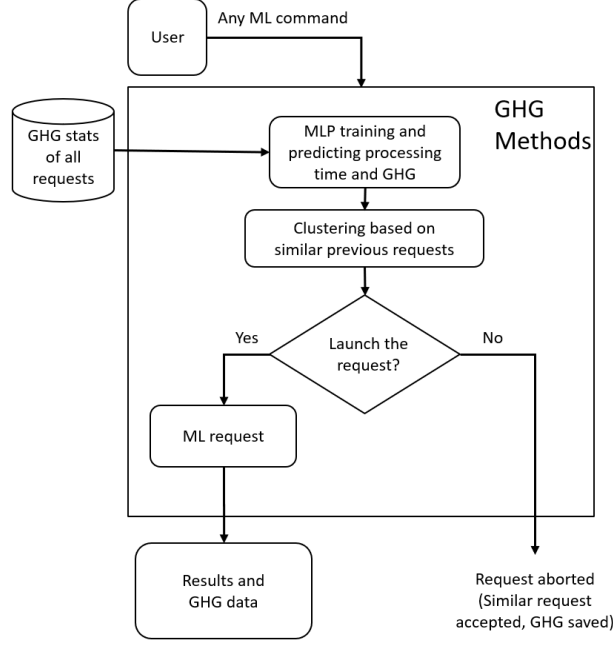


Figure 6: GHG module architecture

amount. This MLP have 5 hidden layers of 25 neurons. It uses a *relu* activation function and an *adam* solver. Then, a k-means clustering algorithm is used to regroup every similar request to the current request. The list is proposed to the user so he can spare his execution, with some similar results available from the historic. Knowing how much GHG will be generated and knowing the similar results of the past, the user will finally decide if yes or no he wants to execute his new request. Fig. 10 presents an example of the information from the chatbot concerning the GHG before launching a new request.

## 2.6 Explainability methods

The goal of this part is to get rid of the famous "black box" problem in machine learning. When most frameworks usually display the results for every executed algorithm, AI<sup>2</sup> will systematically display the ad-hoc graphics, tables and texts that will ensure a better explainability for a particular algorithm. It could be some learning curves, some scalability curves, and some confusion matrices. For instance, for a clustering process, some stacked radar graphics (one per cluster) are produced, plus a Silhouette index graphic that shows the cluster's consistency. A cluster table and a text (in LaTeX format) are also created to complete the explainability of the process. For each machine learning algorithm, the totality of the graphics, tables and texts are generated using the *explain()* method.

```

Predicted execution time (in sec): 4.498
Predicted generated GHG: 4.899e-05 kg CO2

Here are the most similar requests in case launching another
request can be avoided.
    Request _2022-11-21_21-23-43 using dataset make_blob
    Request _2022-11-22_13-54-45 using dataset make_blob
    Request _2022-11-22_14-29-32 using dataset make_blob
Launch the request (y/n)?

```

Figure 7: Information from the chatbot concerning the GHG before launching a new request.

### 3 Results

The following presents five functional use cases. They emphasise the singularity of the AI<sup>2</sup> framework. It shows how a user can execute some requests to this framework and what type of results are presented as output. The output graphics, tables, and texts are not presented in this paper for two reasons: 1. It is not what this paper intends to demonstrate. For instance, there is no need to show result for a simple clustering K-mean process. 2. There would have needed too many graphics, tables and texts to present in this paper. Case 1 to case 5 present a clustering, a reduction of dimensionality, a classification, a prediction, and an evaluation of the feature’s importance.

#### 3.1 Case 1: Clustering

The first case is about a clustering process. As mentioned earlier, the user must write his query in English in the chatbot. For this first case, the following command has been entered: *I want to perform a clustering using iris dataset and having 3 clusters.*

From the Parameters.csv file where a sample is presented in Table 4, the following questions (Table 4) will be generated by the chatbot to fill the required information about a clustering process :

Table 4: Required information and questions to access it.

Key	Type	Return value	Questions
PROBLEM	Y/N	CLUSTERING	Is this clustering?
PROBLEM	Y/N	CLUSTERING	Is this a clustering problem?
PROBLEM	Y/N	CLUSTERING	Is this regrouping?
PROBLEM	Y/N	CLUSTERING	Is this a regrouping problem?
PROBLEM	Y/N	CLUSTERING	Do you want to regroup data?
PROBLEM	Y/N	CLUSTERING	Do you want to cluster data?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?
NB_CLST	Std.		How many clusters?
NB_CLST	Std.		How many groups?

At this first step, AI<sup>2</sup> transparently tries to find the answers in the command entered by the user. After this first step, if AI<sup>2</sup> misses some information, the chatbot will ask for it until every critical information is defined. From this example, the iris dataset is loaded, a *k-means* algorithm is launched with the parameter  $n_{clusters} = 3$  and using the default parameters  $random_{state} = 1$  and  $init = "k - means + +"$ .

The primary results are displayed, presenting a data table along with their clusters, that what most of the frameworks would do. Using AI<sup>2</sup>, each graphic, table and text can be called using the *explain()* method. In this first case, stacked radar graphics are generated for each cluster, allowing to visualize the profile of every cluster. It also generates a graphic of the Silhouette Index, showing and measuring the consistency of every cluster, and finding the mean of the whole clustering process. For each table and graphic, a short text describing it is generated in LaTeX format.

### 3.2 Case 2: Reduction of dimensionnality

The second case is about the reduction of dimensionality. The entered command was: *reduction of dimensionality with iris dataset and having 3 components*. The only required parameter is the targeted number of components that should be used to downsize the dataset. If this parameter is not specified in the command, the chatbot will directly ask to specify it. Since it is defined in this case command, AI<sup>2</sup> will extract three components of the dataset using the PCA algorithm. Always from the Parameters.csv file, the questions shown in Table 5 will be generated by the chatbot to fill the required information about a reduction of dimensionality process :

Table 5: Required information and questions to access it.

Key	Type	Return value	Questions
PROBLEM	Y/N	DIMENSIONALITY	Is this about dimensionality?
PROBLEM	Y/N	DIMENSIONALITY	Is this about dimensionality reduction?
PROBLEM	Y/N	DIMENSIONALITY	Is this about reduction of dimensionality?
PROBLEM	Y/N	DIMENSIONALITY	Is this a regrouping problem?
PROBLEM	Y/N	DIMENSIONALITY	Is this a dimensionality problem?
PROBLEM	Y/N	DIMENSIONALITY	Is this a dimensionality reduction problem?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?
NB_CMPS	Std.		How many components?

The result is a dataset having three principal components (reduced with the PCA algorithm). The *explain()* method generated two graphics: 1. the covariance heatmap of the initial features. 2. a bar graph of the three extracted features' importance (explained variance ratio). For both graphics, a short LaTeX explaining it is generated.

### 3.3 Case 3: Classification

The following case is about the typical problem of classification. For this case, a multiple sentences English is given: *Perform a classification of the iris dataset. I want this request to be reproducible.*

Test [4.8,3.0,1.4,0.2] value. The first sentence of the command is straightforward. Those two sentences are written in a single command. It calls a classification of the iris dataset. To do so, it will call a multilayer perceptron (MLPClassifier from the Scikit-learn framework). The second sentence mention that it requires reproducible results. This will set the seed of the *random\_state* parameter to the "1" integer value, assuring the request gives the same result every time. The opposite would have been a "random request". The seed would have been set to *None*, allowing the request to give slightly different results due to some random synaptic connection initialization. If it is not specified, the request is reproducible. The final sentence commands to try some values. In other words, it aims to classify the specified values [4.8,3.0,1.4,0.2]. The questions in Table 6 will be extracted from the text command.

Table 6: Required information and questions to access it.

Key	Type	Return value	Questions
PROBLEM	Y/N	CLASSIFICATION	Is this about classification?
PROBLEM	Y/N	CLASSIFICATION	Is this a classification problem?
PROBLEM	Y/N	CLASSIFICATION	Do you want to classify data?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?
RANDOM	Y/N	RANDOM	Is this a random request?
RANDOM	Y/N	REPRODUCIBLE	Is this a reproducible request?
TEST	Std.		What are the test values?
TEST	Std.		What values do you want to be tested?

The classification result will then be shown. The training is done with cross-validation having the parameter  $k = 10$ . The whole dataset is split  $k$  times, and the subsets are used to validate to process. The training and validation scores are returned for each step of the cross-validation. While both scores are increasing, the training may continue the learning process. When the training score is still increasing while the validation score starts to decrease, it is precisely the right time to stop the training process. Stopping before that moment creates under-fitted training, and stopping after that point results in overfitted training. Calling the *explain()* method, a learning curve is generated of both the training score and validation score based on the cross-validation.

A state-of-the-art method executes the neural network to classify the data. Earlier in the process, the train and the test data were split, allowing the algorithm to train and evaluate the performances. Performance graphics is also created, showing the performance of the training. Scalability graphics show the ratio of the number of processed data/processing time. Like the other cases, LaTeX texts are generated to explain every graphic.

### 3.4 Case 4: Prediction

This case aims to demonstrate the prediction feature of the AI<sup>2</sup> framework, using the *MLPRegressor* from *Scikit-learn*. It also shows how to preprocess a dataset before calling an algorithm. This preprocessing can be called in the chatbot. In this case, the following English command is given: *Do the preprocess of the iris2 dataset*. Note that the iris2 dataset is identical to the iris dataset, except that the class field is not included. Selecting the columns of a dataset is not included in

this first version of AI<sup>2</sup>, but it will be in a different version. The iris2 dataset remains with four features: Sepal length, sepal width, petal length and petal width. The value of the petal width must be predicted. When responding to the chatbot’s questions, the user must specify that the first three fields are non-normalized features and the fourth is a regression value. After responding to the questions in the chatbot, the iris2.json file is created, containing the information about the configuration. The iris2\_preprocessed.csv data file is also created containing the preprocessed data. A second command can be sent to AI<sup>2</sup> using the chatbot: *I want to make a prediction using the iris dataset. Test [4.5,3.1,1.2]*. The questions in Table 7 will be extracted from the text command.

Table 7: Required information and questions to access it.

Key	Type	Return value	Questions
PROBLEM	Y/N	PREDICTION	Do you want to make a prediction?
PROBLEM	Y/N	PREDICTION	Is this a prediction problem?
PROBLEM	Y/N	PREDICTION	Do you want to predict something?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?
TEST	Std.		What are the test values?
TEST	Std.		What values do you want to be tested?

Three graphics are generated to explain the results as in 3.3. A learning curve is displayed to ensure no training underfitting or overfitting. A second graphic shows the performance of the training process. Moreover, a third graphic shows the scalability of the training. As always, LaTeX texts are created to explain the figures, ready to be cut and pasted in a LaTeX document.

### 3.5 Case 5: Feature’s importance

This next case shows how to evaluate the feature importance in the AI<sup>2</sup> framework. The following command has been typed in AI<sup>2</sup>’s chatbot: *Find the importance of the features with the iris dataset*. This command calls a Random Forest algorithm. More precisely, the *RandomForestClassifier* and the *RandomForestRegressor* from the *Scikit-learn* framework. According to the configuration file’s content (iris.json in this case), it will detect whether it is a dataset made for regression or classification. In this case, the iris is a dataset made for classification, so the *RandomForestClassifier* algorithm will be used. From the Parameters.csv file, the questions shown in Table 8 are asked by the chatbot to fill in the information about the feature importance algorithm:



Table 8: Required information and questions to access it.

Key	Type	Return value	Questions
PROBLEM	Y/N	FEAT_IMP	Is this about feature importance?
PROBLEM	Y/N	FEAT_IMP	Is this about the importance of the features?
PROBLEM	Y/N	FEAT_IMP	Is this a feature importance problem?
PROBLEM	Y/N	FEAT_IMP	Do you want to know the feature importance?
DATASET	Std.		What is the dataset?
DATASET	Std.		Which data are used?

The *explain()* method gives a graphic where the X axe represents the index of the features, and the Y axe shows each feature’s normalized level of importance. A LaTeX explanation text is generated as usual.

### 3.6 GHG algorithms validation

As stated in 2.5, the AI<sup>2</sup> framework predicts GHG for each algorithm to be executed. Execution time is also predicted before calling the machine learning algorithm. To validate those predictions, a clustering algorithm has been called within 50 iterations loops. For each execution, a randomized dataset of 10,000 to 50,000 rows and 5 to 20 features have been used. Those datasets were generated by the *make\_blob()* function of the *scikit-learn* framework. Fig. 8 shows the validation of the predicted and real values of the generated GHG. X axe displays the 50 iterations, and the Y axe shows the level of GHG (in  $kgCO_2$  unit). The regression algorithm was trained from a dataset containing 1382 rows containing the request’s historical.

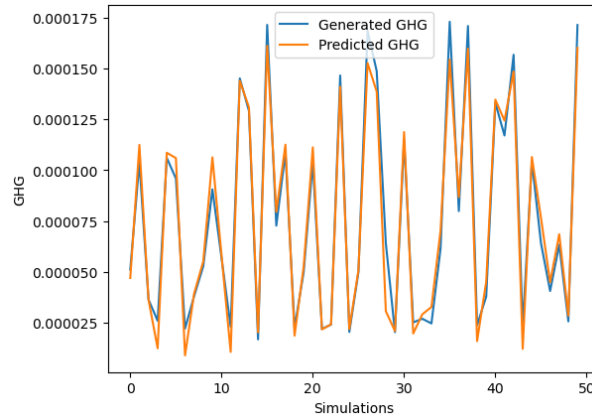


Figure 8: Validation of the predicted and real GHG

Fig. 9 displays the validation of the predicted and actual values of the execution time for every iteration of the loop. X axe shows the 50 iterations, and the Y axe shows the execution time (in sec.).

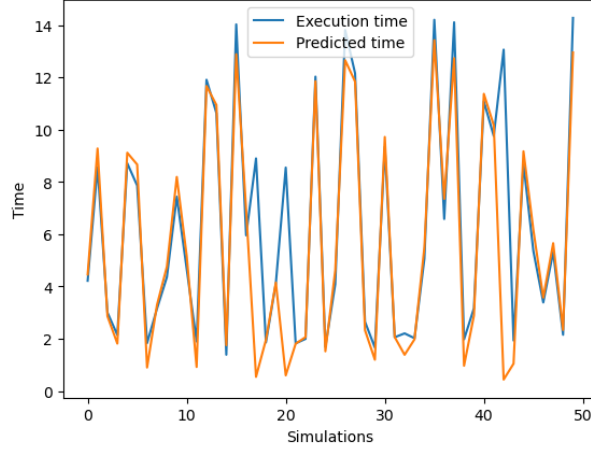


Figure 9: Validation of the predicted and real execution time

```
Here are the most similar requests in case launching another request can be avoided.
Request _2022-11-21_21-23-43 using dataset make_blob
Request _2022-11-22_13-54-45 using dataset make_blob
Request _2022-11-22_14-29-32 using dataset make_blob
```

Figure 10: iris.json structure file

Concerning the predicted and real GHG and execution time, it can be seen that the signal is reasonably reconstructed.

Finally, before launching each request, AI<sup>2</sup> proposes similar requests from the request's historic after extracting this information using a clustering process. Fig. 10 presents an example of the AI<sup>2</sup> propositions of the similar requests.

## 4 Discussions

The first contribution of this paper is to present an accessible framework. With its state-of-the-art NLP methods, this machine learning framework is a pioneer in communicating with a non-expert user in English. The new Transformers technology allows the AI<sup>2</sup> framework to receive native language commands extracted, parsed and executed. When there is an essential missing parameter, AI<sup>2</sup> will use its chatbot to communicate with the user, asking him to enter the missing information. With this NLP interface, a user can exploit the AI<sup>2</sup> framework without knowing how to code with a programming language like Python or others.

The AI<sup>2</sup> framework is GHG-aware, and this is the second contribution of this paper. The CodeCarbon library is encapsulated in each of its ML functions, allowing the calculation of the GHG for each algorithm executed. Those GHG records are kept in a register and used to predict, based on ML, the GHG generated before the execution. AI<sup>2</sup> also propose some similar registered

requests, also based on ML, to save this execution and save GHG.

The opposite of most other frameworks, AI<sup>2</sup> systematically encapsulates the most important format of explanations about the data and the results. This aspect of the framework is crucial to solving the famous black-box problem. This is the third contribution of this paper. Most of the machine learning framework is not systematically offering some explainability with the results. AI<sup>2</sup> does. It generates, for each request, some graphics, some tables, and some texts explaining the results and the data, thus, making this framework more ethical than others.

The final contribution of this paper is data preprocessing. It usually takes time to code a suitable preprocessing of the data. The AI<sup>2</sup> framework proposes a method based on communication with the chatbot to automatize this process. Guided by the AI<sup>2</sup> chatbot, the user may do some basic preprocessing of its datasets by establishing the dataset’s structures. Having a structure stored in a JSON file, the preprocessing module can generate a new preprocessed dataset.

Comparing AI<sup>2</sup> with other machine learning frameworks, what is the advantage of using it? For now, there are frameworks more complete and more sophisticated. The AI<sup>2</sup> framework targets non-expert users who need a machine-learning algorithm to process their data. Typical AI<sup>2</sup> users would be, for instance, researchers, engineers, teachers and students in natural science, and so on. A significant part of the scientific community cannot program complex algorithms using a programming language. An NLP interface is the best solution since it requires no programming skills.

Table 9 shows a comparison between AI<sup>2</sup> and the other popular machine learning framework, according to 3 criteria: 1. NLP interface, 2. GHG awareness, 3. Explainability, and 4. NLP Preprocessing.

Table 9: Comparison of the popular machine learning frameworks, specialized frameworks, and AI<sup>2</sup>

Framework	NLP	GES Aware	Explain.	Prepro.	Code req.	Ref.
AIX360	NO	NO	<b>YES</b>	NO	YES	[3]
ELI5	NO	NO	<b>YES</b>	NO	YES	[3]
Gluon	NO	NO	NO	NO	YES	[27]
Keras	NO	NO	NO	NO	YES	[27]
LIME	NO	NO	<b>YES</b>	NO	YES	[25]
Matlab	NO	NO	NO	NO	YES	[27]
MXNet	NO	NO	NO	NO	YES	[41]
Orange	NO	NO	NO	NO	<b>NO</b>	[8]
PyTorch	NO	NO	NO	NO	YES	[27]
Scikit-learn	NO	NO	NO	NO	YES	[27]
SHAP	NO	NO	<b>YES</b>	NO	YES	[25]
Skater	NO	NO	<b>YES</b>	NO	YES	[3]
Tensorflow	NO	NO	NO	NO	YES	[41]
What-if Tool	NO	NO	<b>YES</b>	NO	YES	[25]
XAI	NO	NO	<b>YES</b>	NO	YES	[25]
CodeCarbon	NO	<b>YES</b>	NO	NO	NO	[22]
<b>AI<sup>2</sup></b>	<b>YES</b>	<b>YES</b>	<b>YES</b>	<b>YES</b>	<b>NO</b>	[13]

Note that some well-known frameworks may seem absent from the list: CNTK and Theano are no longer supported. Caffe2 is merged with PyTorch. According to Table 9, we can regroup

the frameworks into three categories: 1. The general, multi-purpose frameworks (Gluons, Keras, MXNet, Tensorflow, PyTorch, Matlab, Orange and Scikit-learn) 2. The Explainability frameworks (AIX360, ELI5, LIME, SHAP, Skater and XAI), and 3. The GHG-aware framework (CodeCarbon).

This table shows AI<sup>2</sup>'s novelty. It is the only framework that combines all the studied criteria (NLP interface, GES awareness, Explainability, Preprocessing, and Coding required). It is the first framework to have an NLP interface to send the instructions to the framework. Several frameworks integrate the explainability of the data and the models, but no general and multi-purpose framework includes it. AI<sup>2</sup>: The next leap toward native language-based, GHG-aware and explainable ML framework.

## 5 Conclusion

This framework proposes a tool for the non-expert to use machine learning methods. It offers an NLP interface so the user can communicate with the framework using a chatbot. It encapsulates some very concrete functions to provide ecological awareness. It includes the principle of explainability, proposing expanded results explications for different algorithms. It finally allows preprocessing of data using an English chatbot.

This framework could be the first draft of a long series of improvements. There are many future works to do for each of its contributions. Regarding its NLP interface, this framework can be improved by training the pre-trained Transformer on a specific machine learning-oriented text corpus. Likely, the NLP's performance will significantly improve. The chatbot method can also be optimized to minimize errors and recognize the user's intentions. Questions used to extract command information can be improved by increasing the quality and the number of questions. GHG awareness can be improved. Better methods can be found to minimize wasted energy, maximize the GHG estimation before calling an algorithm, and cluster similar requests. There is a lot to do, but this framework has the merit of being aware of the climate change problem and proposing a modest solution. Explanations available for each data and machine learning algorithm can also be optimized in quantity and quality. Some essential explanations are included in this framework, but those need to be systematically included. Regarding the preprocessing module, there are many things to add. For instance, some normalization methods can be added. The rows and columns selection can be added to this module, also. Some graphics can be added to plot data at the preprocessing stage. Finally, this framework contains a limited number of ML algorithms. Some more ML algorithms can be easily added to the AI<sup>2</sup> framework.

## 6 Acknowledment

This work has been supported by the "Cellule d'expertise en robotique et intelligence artificielle" of the Cégep de Trois-Rivières and the Natural Sciences and Engineering Research Council.

## References

- [1] Martín Abadi et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

- [2] Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121, 2020.
- [3] Namita Agarwal and Saikat Das. Interpretable machine learning tools: A survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1528–1534.
- [4] Mohiuddin Ahmed, Raihan Seraj, and Syed Mohammed Shamsul Islam. The k-means algorithm: A comprehensive survey and performance evaluation. *Electronics*, 9(8):1295, 2020. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.
- [5] Gérard Biau and Erwan Scornet. A random forest guided tour. *TEST*, 25(2):197–227, 2016.
- [6] Jia-Wei Chang, Neil Yen, and Jason C. Hung. Design of a NLP-empowered finance fraud awareness model: The anti-fraud chatbot for fraud detection and fraud classification as an instance. 13(10):4663–4679.
- [7] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1):321–357, 2002.
- [8] Janez Demšar et al. Orange: data mining toolbox in python. *the Journal of machine Learning research*, 14(1):2349–2353, 2013.
- [9] Jean-Sébastien Dessureault and Daniel Massicotte. [2206.08980] explainable global error weighted on feature importance: The xGEWFI metric to evaluate the error of data imputation and data augmentation.
- [10] Jean-Sébastien Dessureault and Daniel Massicotte. [2206.08974] DPDR: A novel machine learning method for the decision process for dimensionality reduction. 2022.
- [11] Jean-Sébastien Dessureault and Daniel Massicotte. [2206.08982] ck-means, a novel unsupervised learning method that combines fuzzy and crispy clustering methods to extract intersecting data. 2022.
- [12] Jean-Sébastien Dessureault and Daniel Massicotte. DPDRC, a novel machine learning method about the decision process for dimensionality reduction before clustering. *AI*, 3(1):1–21, 2022. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.
- [13] Jean-Sebastien Dessureault and Daniel Massicotte. Ai2: a novel explainable machine learning framework using an nlp interface.
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding, 2019.
- [15] Lisa R. Goldberg. *The Book of Why: The New Science of Cause and Effect*, volume 19. 2019. Publisher: Routledge \_eprint: <https://doi.org/10.1080/14697688.2019.1655928>.
- [16] Raffaele Guarasci, Stefano Silvestri, Giuseppe De Pietro, Hamido Fujita, and Massimo Esposito. Assessing BERT’s ability to learn Italian syntax: A study on null-subject and agreement phenomena.

- [17] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 2016. Publisher: The Royal Society Publishing.
- [18] M. I. Jordan. Serial order: A parallel distributed processing approach. Technical report, June 1985-March 1986. (AD-A-173989/5/XAB; ICS-8604).
- [19] Shigeki Karita et al. A comparative study on transformer vs RNN in speech applications. *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 449–456, 2019.
- [20] The Institute for Ethical Ai \& Machine Learning. The institute for ethical AI & machine learning.
- [21] Yinhan Liu et al. RoBERTa: A Robustly Optimized BERT Pretraining Approach.
- [22] Kadan Lottick, Silvia Susai, Sorelle A. Friedler, and Jonathan P. Wilson. Energy Usage Reports: Environmental awareness as part of algorithmic accountability.
- [23] Shivani Malhotra, Vinay Kumar, and Alpna Agarwal. Bidirectional transfer learning model for sentiment analysis of natural language. 12(11):10267–10287.
- [24] Maria das Graças Bruno Marietto et al. Artificial Intelligence MARKup Language: A Brief Tutorial.
- [25] Sina Mohseni, Niloofar Zarei, and Eric D. Ragan. A Multidisciplinary Survey and Framework for Design and Evaluation of Explainable AI Systems.
- [26] Anand Motwani, Piyush Kumar Shukla, and Mahesh Pawar. Novel framework based on deep learning and cloud analytics for smart patient monitoring and recommendation (SPMR).
- [27] Giang Nguyen et al. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, 52(1):77–124, 2019.
- [28] Long Ouyang et al. Training language models to follow instructions with human feedback.
- [29] Sebastian Palacio, Adriano Lucieri, Mohsin Munir, Sheraz Ahmed, Jörn Hees, and Andreas Dengel. XAI handbook: Towards a unified framework for explainable AI, 2021.
- [30] The-Hanh Pham, Vinitha Sree, John Mapes, Sumeet Dua, Oh Shu Lih, Joel E. W. Koh, Edward J. Ciaccio, and U. Rajendra Acharya. A novel machine learning framework for automated detection of arrhythmias in ECG segments. 12(11):10145–10162.
- [31] Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. Multilayer perceptron: Architecture optimization and training. 2016. Accepted: 2021-07-07T10:37:59Z Publisher: International Journal of Interactive Multimedia and Artificial Intelligence (IJIMAI).
- [32] Denis Rothman. *Transformers for Natural Language Processing: Build Innovative Deep Neural Network Architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and More*. Packt Publishing Ltd.

- [33] Denis Rothman. *Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more*. Packt Publishing Ltd, 2021.
- [34] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation.
- [36] Olga G. Troyanskaya, David Botstein, and Russ B. Altman. Missing value estimation. In Daniel P. Berrar, Werner Dubitzky, and Martin Granzow, editors, *A Practical Approach to Microarray Data Analysis*, pages 65–75. Springer US, 2003.
- [37] Ashish Vaswani et al. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [38] Joost Verbraeken et al. A survey on distributed machine learning. *ACM Computing Surveys*, 53(2):30:1–30:33, 2020.
- [39] Zhaobin Wang, Ke Liu, Jian Li, Ying Zhu, and Yaonan Zhang. Various frameworks and libraries of machine learning and deep learning: A survey. *Archives of Computational Methods in Engineering*, 2019.
- [40] Jaehong Yu, Hua Zhong, and Seoung Bum Kim. An ensemble feature ranking algorithm for clustering analysis. *Journal of Classification*, 37(2):462–489, 2020.
- [41] Kuo Zhang, Salem Alqahtani, and Murat Demirbas. A comparison of distributed machine learning platforms. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9, 2017.
- [42] Xingzhou Zhang, Yifan Wang, and Weisong Shi. pcamp: Performance comparison of machine learning packages on the edges. 2018.