An exhaustive list of the Data Analytics algorithms provided in the I-BiDaaS platform

When a company wants to start understanding the data of their business, the term Data Analytics rapidly comes into play, and a difficult task arises for a user since the number of existing algorithms associated with data analysis is very big. Therefore, the common question is: **"What is the best algorithm I should use to process and understand my data?"**

One of the objectives that were defined for the I-BiDaaS project (and specifically for WP3 – Batch processing innovative technologies for rapidly increasing historical data) was to provide what we call Innovative Distributed Solvers (IDS), capable of solving optimization problems by using innovative and distributed big data analytics architectures. These algorithms have to leverage existing tools/frameworks/libraries, such as COMPSs, dislib, scikit-learn or PyTorch, while offering scalable and reliable distributed data management, with an easy-to-use interface that avoids vendor lock-in with the underlying storage system.

In this blog article, we provide an exhaustive list of all the algorithms that have been implemented and used during the whole lifetime of the I-BiDaaS project. Our intention with this is to show that we have a big portfolio of algorithms ready to help for many different real life data analysis cases. In fact, all the algorithms listed here have been used in the I-BiDaaS use cases from the Manufacturing, Telecom and Financial sectors), either alone or with a compatible combination of them.

The list also includes where the algorithms can be located, thus downloaded and used, since in the majority of the cases, the algorithms have been published as open source so they can be shared with different communities. It is also important to mention that we never wanted to "reinvent the wheel", thus, we have reused existing algorithms/frameworks when the case in question allowed it. For the rest of cases, either improved versions of existing algorithms (e.g., in terms of scalability/parallelism, or underlying frameworks that run them) or new algorithms have been provided. Some algorithms have not been published due to the fact that the code modifications are tied to the proprietary data sets of our use cases, thus making them impossible to be shared without revealing sensitive information from our data providers.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

Finally, even when the I-BiDaaS platform has this big list of algorithms available (as shown in this article), we also provide in our user interface a questionnaire for end-users that provides guidance on which should be the best choice when selecting an algorithm to analyse data. The interface asks the user a number of chained questions, and depending on their answer, it is able to recommend a particular algorithm for the type of data or analysis that the user wants to conduct. As we stated before, it is not only enough for us to provide a rich set of algorithms, but it is even more important to guide users to what is more convenient for them to get the most of their analyses, and the questionnaire has certainly achieved that.

List of algorithms developed and publicly available:

- ADMM based K-means
    - Brief description: A convex version of the renowned K-means clustering algorithm. Unlike the standard K-means algorithm, which requires the user to provide the desired number of clusters, the convex version of K-means algorithmically determines the optimal number of clusters.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_ADMM_Kmeans.py
- ADMM based logistic regression
    - Brief description: ADMM is a general optimization framework, well suited to training a wide variety of machine learning models, and has a distributed implementation. Logistic regression is a binary classification algorithm, known for its good performance in practice.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_Logistic_ADMM.py
- Stochastic gradient-based logistic regression
    - Brief description: Stochastic gradient-based logistic regression offers the advantage of a communication efficient solver, alleviating the communication cost of training a distributed logistic regression model.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_Logistic_StochGrad.py
- ADMM based Ridge and ElasticNet regression

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

    - Brief description: Ridge and ElasticNet algorithms are part of the family of regularized regression algorithms. Both are very similar to the Lasso algorithm, with the main difference being the regularization: while Lasso uses a L1 regularizer to induce sparsity, Ridge regression uses a L2 regularizer, while ElasticNet combines both L1 and L2 regularization.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_Ridge_ADMM.py

      https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_ElNet_ADMM.py

- Differentially private K-means
    - Brief description: Represents a version of K-means that provides some additional privacy guarantees with respect to the data used for training, i.e., the algorithm makes it impossible to perform reverse engineer the model and exploit the data.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/dpkmeans.py
- Differentially private Random Forest
    - Brief description: Differentially private Random Forest is an extension of the basic RF algorithm that provides privacy guarantees with respect to the data used for training, i.e., it makes it impossible to perform reverse engineering on the model and exploit the data.
    - Location: https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/dpRandomForest.py
- ADMM Lasso:
    - Brief description: Lasso (**L**east **A**bsolute **S**hrinkage and **S**election **O**perator) represents an algorithm for training sparse linear regression that uses shrinkage. It performs L1 regularization.
    - Location:

      https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_Lasso_ADMM.py

      https://github.com/ibidaas/knowledge_repository/blob/master/tools_technologies/sources/batch_processing/unspmf/distributed_Lasso_ADMM_warm_start.py

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

https://dislib.bsc.es/en/stable/dislib.regression.lasso.html

List of algorithms developed, not publicly available due to use case restrictions:

- Elbow method
    - Brief description: The Elbow method is a heuristic to determine the number of clusters in a data set. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve as the number of clusters to use.
    - Location: Not publicly available
- Weighted Random Forest (scikit-learn based)
    - Brief description: Weighted Random Forest represents an extension of basic Random Forest. When a problem is imbalanced, i.e., there are more samples belonging to class "a" than to class "b", the basic random forest algorithm is susceptible to discriminate and favour the larger class. Weighted Random Forest manages to avoid this scenario, by associating different weights (or rewards) for different classes in the objective function.
    - Location: Not publicly available. Based on scikit-learn: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- Rolling analytics for Sentiment Score computation
    - Brief description: Approximate windows-based analytics for counting the sentiment score over huge amounts of streaming text. The approach overcomes the large memory consumptions of typical rolling windows systems, by using the notion of time-decayed counters (in which a value automatically decays over time using a mathematical formula). This allows to provide different types of analytics (such as sentiment score, word frequencies, and most frequent words) for different and longtime intervals (e.g., last minute, last hour, and last day), with minimum allocated resources (i.e., only a simple counter for each feature is needed).
    - Location: Not publicly available. The description of our approach is available at Deliverable D4.3.

List of algorithms used from public frameworks:

- K-means (dislib)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

    - Brief description: K-means clustering is a method that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centres or cluster centroid). It is a well-established clustering algorithm, known to scale well to a large number of samples.
    - Location: https://github.com/bsc-wdc/dislib/tree/master/dislib/cluster/kmeans

- DBSCAN (dislib)
    - Brief description: Another well-known clustering algorithm. Unlike K-means, which assumes convex-shaped clusters, DBSCAN can find clusters of any shape. Additionally, DBSCAN does not require a user provided estimate of the number of clusters.
    - Location: https://github.com/bsc-wdc/dislib/tree/master/dislib/cluster/dbscan
- Random Forest (dislib)
    - Brief description: It is a well-established classification algorithm. It represents an ensemble algorithm, as each "forest" consists of a number of decision trees. Works well for supervised learning classification problems, and has a good interpretability and practical performance.
    - Location: https://github.com/bsc-wdc/dislib/tree/master/dislib/classification/rf
- t-SNE (scikit-learn)
    - Brief description: An algorithm used mostly for visualization of high-dimensional data. It projects the data in 2D or 3D, with the aim of keeping a neighbouring and clustering structure of the data as close to original as possible.
    - Location: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
- PCA (scikit-learn)
    - Brief description: Principal component analysis (PCA) is the process of computing the principal components of data and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible.
    - Location: https://scikit-learn.org/stable/modules/decomposition.html#principal-component-analysis-pca

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

- Deep Neural Network (PyTorch framework)
  - Brief description: A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions. These components functioning similar to the human brains and can be trained like any other ML algorithm. They contain a large number of layers that enable them to learn relationships between the input data and the target values, whether that relationship is linear or highly non-linear.
  - Location: https://pytorch.org/
- DenseNet-201 (PyTorch framework)
  - Brief description: DenseNet-201 is a convolutional neural network that is 201 layers deep. It is used mostly in image classification.
  - Location: Torchvision https://pytorch.org/docs/stable/torchvision/models.html
- K-modes (K-modes library)
  - Brief description: K-modes is a clustering algorithm similar to K-means, but that allows to work with categorical data.
  - Location: https://pypi.org/project/kmodes/
- Time-series based on Prophet (Facebook)
  - Brief description: The adopted approach based on Prophet, unlike conventional time series models, does not require the values to be regularly spaced, nor there is need to interpolate missing values, which was suitable for our considered data set with a high degree of missing values.
  - Location: https://facebook.github.io/prophet/
- XGBoost (XGBoost library)
  - Brief description: Distributed variant of gradient boost used to train ensemble algorithms, like Random forest. Known to be a highly efficient, flexible and portable library for machine learning algorithms under the gradient boosting framework.
  - Location: https://xgboost.readthedocs.io/en/latest/
- CATBoost (CATBoost library)
  - Brief description: CATBoost represents a variant of gradient boost used to train ensemble algorithms, like Random forest. Its main advantages are an innovative, fully automated algorithm for processing categorical features, order boosting, a permutation driven

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.

alternative to the classical boosting algorithm as well as fast and easy to use training.

    - Location: https://catboost.ai/

- Interquartile Range Test (IQR – SciPy)
    - Brief description: IQR (interquartile range) test is used for outlier detection. Outliers represent values that fall outside of the overall pattern of the data. IQR effectively checks how the data is spread about its median.
    - Location: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.iqr.html

Find & Follow us

Website | **Twitter** | **LinkedIn**

Zenodo | **OpenAIRE** | **GitHub**

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 780787.