# An exploratory study on machine learning frameworks

**Caio Flexa**[1,2,*], **Walisson Gomes**[1], **Sérgio Viademonte**[1]

[1]Instituto Tecnológico Vale,
Rua Boaventura da Silva 955, Nazaré, Belém, 66055-090 Pará, Brazil.

[2]Applied Electromagnetism Laboratory, Federal University of Pará,
R. Augusto Corrêa, Guamá 01, Belém, 66075-110 Pará, Brazil

{caio.rodrigues,walisson.gomes}@icen.ufpa.br,

sergio.viademonte@itv.org

***Abstract.*** *This document describes a preliminary study on computing frameworks and technologies, for the purpose of developing machine learning (ML) system applications. Several frameworks, application programming interfaces and programming libraries for ML algorithms have been developed in the last few years, in a relatively short period of time, making difficult a decision on which one to chose in a particular application. This study reviews some criteria and performs a preliminary evaluation of some of the most used ML technologies for developing system applications, with the purpose to guide and facilitate the decision on which of them to apply, given a particular application.*

## 1. Introduction

In the last few years several computing technologies have emerged for the processing of large data set, such as computing frameworks, application program interfaces (APIs) and programming libraries. Processing large amount of data is a mean to acquire knowledge implicitly distributed on such data volumes, with the purpose to developed smart computing applications. Examples of such smart applications in industry are predictive systems for operational failures, applications for preventive maintenance of industrial assets, tailoring energy consumption according to optimal use, among others.

The research described in this document is part of a major investigation on technologies for processing large volumes of data, focusing on artificial intelligence, machine learning and data analytic capabilities. This investigation aims to guide and facilitate the decision on which technologies for processing large volumes of data to chose for the development of smart computing applications. This document describes the motivations of this investigation, and presents some preliminary results. This research project has been developed as part of Vale S.A (Vale) initiatives in the Industry 4.0 [MICS 2019]. Vale is a multinational mining organization, headquartered in Brazil, with operations that spread over various countries.

An industrial problem has been selected for our investigation. We have used a data set about railways incidents obtained from Vale's operations to evaluate the predictive performance of ML algorithms on computing the likelihood of an incident occurrence. The

[2]Supported by Instituto Tecnológico Vale and Brazilian National Council for Research and Development (CNPq), under the research grant $402748/2018-2$.

data set covers a period of 9 years, with 59 features, and approximately 25 thousand instances. To illustrate some of the features used in our investigation were: the time, date, municipality and the position along the railway (given in kilometers) where an incident occurred, the existence of level crossings, the occurrence of mechanical or electric failure, among others. For the reason of data confidentiality, we have labeled the incidents in our study as I-A, I-B, I-C and I-D. Therefore, the analysis becomes a classification problem, with 4 classes. A series of ML algorithms were applied to build classifiers, as we are concerned in identifying classes of incidents, and later to evaluate the predictive performance of the obtained classifiers. Readers interested in detailed information about this initiative of railways incidents analysis, should refer to specific literature, [Viademonte et al. 2018].

The focus of this paper is in the APIs and respective ML algorithms and their suitability for the choosen industrial problem. It is important to note that the main purpose of this research is to investigate the suitability of technologies to specific problems, not a direct comparison among the selected technologies. This document describes the current stage of the investigation on technologies for processing large volumes of data, and presents some preliminary results. Section 1, Introduction, describes the context of our work. Section 2 describes the frameworks and APIs and briefly comments some of their particularities, in Section 3, they are applied to the railway incidents problem. Finally, Section 4 concludes the paper and gives the some final remarks.

## 2. Technology descriptions

This section briefly describes and highlights some of the key aspects of the technologies under study in this research project. Initially, four technologies were chosen for both theoretical and practical analysis: Apache Spark, TensorFlow, Scikit-learn and PyTorch. These technologies are subject of study in a number of recent papers, including [Shi et al. 2016, Giannini et al. 2017, Mavridis and Karatza 2017].

### 2.1. Apache Spark

Originally developed in AMPLab of the University of California, Apache Spark[3] is an open source framework for distributed computing, later passed on to the Apache Software Foundation that has maintained it ever since. Spark is a Big Data framework that provides an interface for computing cluster parallelism and fault tolerance, whose main purpose is to process large volumes of data. Another important aspect is that Spark supports different programming languages: Java, Python and Scala.

One of the strongest points in Spark is that it provides several components, working in an integrated fashion. The main components are: *Spark Streamming*: enables the processing of flows in real time; *GraphX*: provides a graphs processing capability; *Spark-SQL*: SQL language for data queries and processing; *MLlib*: machine learning library with several algorithms for classification, regression and clustering.

### 2.2. TensorFlow

TensorFlow[4] is an open source library for numerical computation developed by Google Brain Team. It's mostly applied for deep neural networks applications, but can also be

[3]Online source: `https://spark.apache.org/documentation.html`.

[4]Online source: `https://www.tensorflow.org/api_docs`.

used for others machine learning system applications. TensorFlow supports different programming languages, such as Python, Javascript, C++, Java, Go, and Swift.

The library is called TensorFlow due its fundamental processing structure, Tensors, which are graph structures that implement a distributed data flow. Tensors flow along the graph, being submitted to several computations implemented by nodes. This type of structure is applied in both fast experimentation and large-scale projects, regarding its modular and parallel computing capabilities. TensorFlow provides several capabilities for data processing, regression, classification and clustering. The implementation of neural network models is further facilitated by Keras, a high-level API that encapsulates many of TensorFlow inherent complexities.

### 2.3. Scikit-learn

Scikit-learn[5] is an open source ML library for Python programming language. It provides a large set of ML algorithms for classification, regression, clustering, dimensionality reduction, among others. For example, it includes implementations of following ML algorithms: support vector machine, random forest, gradient boosting and $K$-medians. In addition, it provides a seamlessly integration with Numpy and SciPy libraries.

The scikit-learn project started as *scikits.learn*, as part of *Google Summer of Code* project, coordinated by David Cournapeau. The name results from the idea that it is a "SciKit" (SciPy Toolkit), an extension of the SciPy library (https://scipy.org/scipylib/index.html) developed by third parties and distributed separately. Scikit-learn library is currently under active development, sponsored by INRIA, Telecom ParisTech and Google.

### 2.4. PyTorch

PyTorch[6] is a Torch-based library for Python using GPUs and CPUs. It was released by Facebook's artificial-intelligence research group on October of 2016. PyTorch has been used in applications such as natural language processing and the Pyro software. Pyro is a probabilistic programming language built on Python developed at Uber Research labs. At the end of March 2018, Caffe2 (Convolutional Architecture for Fast Feature Embedding), a deep learning framework originally developed at University of California, Berkeley was merged into PyTorch. However, it is not a simple set of wrappers to support a popular language. PyTorch was rewritten and adapted to be fast and look native, often used as a substitute for Numpy. PyTorch provides two high-level capabilities:

- Tensor computation with strong GPU acceleration;
- Deep Neural Networks (package *torch.autograd*).

## 3. Theoretical Comparison

Each technology has been developed with one or more objectives in mind. For example, Spark is able to deal with huge amounts of data and it is also a cluster manager for commodity hardware. TensorFlow and Pytorch are flexible and focus on deep learning, while Scikit-learn is easier to install and use, and has been developed for machine learning applications, such as classification and clustering, but not necessarily deep learning.

[5]Online source: `https://scikit-learn.org/stable/documentation.html`.
[6]Online source: `https://pytorch.org/docs/stable/index.html`.

Therefore, the selection of these technologies will depend on the task to be addressed and its requirements. We have applied an industrial problem, the predictive analysis of railway incidents, to evaluate those frameworks and APIs. The selected criteria in this research are: capabilities for parallel computation, data ingestion and processing, easy of use and function availability.

**License**: the four technologies are open source and have community support.

- Apache Spark and TensorFlow – Apache License 2.0.
- Scikit-learn and Pytorch – Berkeley Software Distribution (BSD).

**Parallel computation**: all technologies support execution in CPU with multiple cores and in clusters. Tensorflow and Pytorch both have support to GPU platforms. In fact, they try to take full advantage of GPU's numerical processing capabilities to train neural networks. Apache Spark was designed for distributed computing environment and, although some libraries are necessary, it can also include GPUs. Scikit-learn does not have GPU support. The main reason for this is to keep fewer dependencies and avoid issues with specific platforms.

**Preprocessing**: all selected technologies can benefit from Python built-in functionality and science libraries. Scikit-learn presents a simple and direct way to transform data into the desired formats. In fact, the library can also be employed to prepare them beforehand passing to other ML frameworks. Each of the libraries has its own way to implement data preprocessing, such as *one hot encoding*, *normalization*, *filtering*, etc. Tensorflow and Pytorch also provide a nice interface for preprocessing images with, respectively, *image* and *tourchvision* modules. Apache Spark stores the data in-memory during processing to aid operations, enabling for optimized performance. At the core of the Spark *framework* data structure is termed *DataFrame*, which represents a set of data that can be implicitly processed using parallelism and distribution. This level of abstraction allows programmers to focus on the implementation of the ML models without requiring additional knowledge. Its *pipeline*, a high-level abstraction, allows to create a stream of data processing where each stage performs a data preprocessing task.

**Data ingestion**: data ingestion can directly affect the overall time to build a model. With Scikit-learn, data can be passed using well-known libraries such as Numpy and Pandas. Other technologies can commonly use those libraries, but they have their own way of dealing with data ingestion and the creation of data processing pipelines. Tensorflow uses the *tf.data* API mainly to deal with data ingestion when using GPU and TPU; Pytorch allows a certain level of control of data ingestion with pre-defined interfaces for data loading and sampling, keeping parallelism transparent; Apache Spark, was designed to be efficient with distributed computation and large data sets, therefore it presents better capabilities for ingesting and preprocessing (very) large datasets.

**Easy of use**: Scikit-learn is the simplest and easiest to start with, among the four technologies studied in this research. If a more complex architecture is necessary, specially when implementing neural network models, Tensorflow and Pytorch are the way to go. They have a similar and straight forward installation process, with good documentation and active community. Tensorflow supports *Keras* API layer, which offers a good abstraction for the creation of deep neural network models. At the other hand, Pytorch is more dynamic, allowing a simple definition and debugging. Apache Spark is a powerful tool for large-scale data applications, since it encapsulates all the complexity

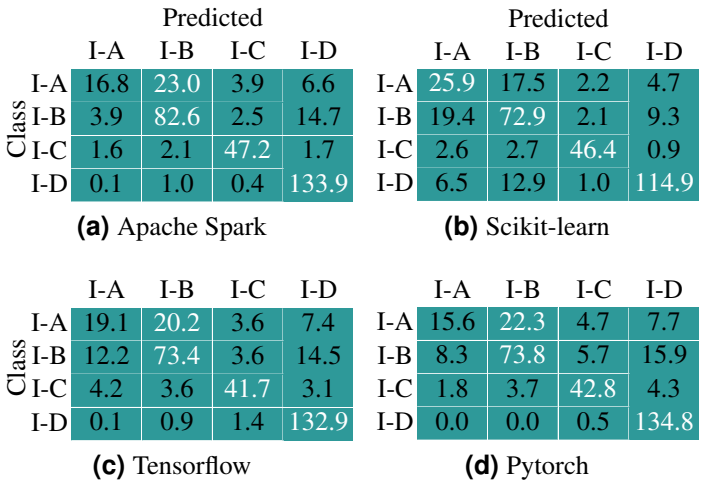


**Figure 1. Confusion matrices for all four technologies. Classes I-A, I-B, I-C, and I-D represent four common incidents.**

of parallelism. However, the use of Apache Spark is not trivial, mainly considering data structures manipulation and programming.

**Function availability**: all the selected technologies implement a considerable number of commonly used ML algorithms for data clustering, regression, classification and dimensionality reduction, specially Scikit-learn and Apache Spark. TensorFlow and Pytorch provide better capabilities, such as high level libraries, for building and processing neural network models, including deep learning.

## 4. Empirical Evaluation

With the purpose of evaluating the applicability of the selected technologies, they were applied to the railway incident prediction problem. Since the technologies have different design purposes, they implement different algorithms and specific functionalities. Apache Spark and Scikit-learn were evaluated through their respective implementation of CART decision tree generator algorithm. Tensorflow and Pytorch were evaluated through of a neural network model with three dense layers, with *RMSprop* optimizer and 500 epochs.

The evaluation has been done numerically and qualitatively, using ML algorithms with a ten fold cross-validation procedure. Table 1 displays the mean accuracy, balanced accuracy and precision, with the respective standard deviations. Figure 1 shows the four confusion matrices on our experiments.

| | **Apache Spark** | **Scikit-learn** | **Tensorflow** | **Pytorch** |
|---|---|---|---|---|
| Accuracy | $0.820 \pm 0.014$ | $0.760 \pm 0.019$ | $0.781 \pm 0.015$ | $0.781 \pm 0.022$ |
| B. accuracy | $0.754 \pm 0.019$ | $0.737 \pm 0.022$ | $0.716 \pm 0.026$ | $0.708 \pm 0.030$ |
| Precision | $0.817 \pm 0.020$ | $0.768 \pm 0.020$ | $0.770 \pm 0.017$ | $0.773 \pm 0.029$ |

**Table 1. Performance measures on a ten fold cross-validation. The results are represented by mean and standard deviation.**

It can be observed that there is a small difference among the four technologies and two algorithms. Taken the accuracy and precision measures, Apache Spark shows the best

performance. The confusion matrices show that the algorithms tend to misclassify I-A. This effect was reduced only in Scikit-learn's. The class I-A refers to important incident.

The ML algorithms were not complex to use, specially with Scikit-learn. Tensorflow and Pytorch have abstraction libraries, helping to define the sequential network, and the use of *Keras* in Tensorflow significantly reduces its complexity. Apache Spark demonstrated good results, but it is less intuitive to use, specially due to its more complex procedures for data processing.

The results discussed above are not conclusive, in terms of which of those technologies performs better, in general. The results illustrated in Figure 1 and Table 1 provide an indication of which, among the selected technologies, would be more appropriate to apply for a problem similar to the incident analysis, introduced in Section 1.

## 5. Conclusions and further work

This document presents some preliminary results, as part of an exploratory study on computing frameworks and technologies for developing ML system applications. Four ML technologies have been selected, Apache Spark, Scikit-learn, Pytorch and TensorFlow. This initial evaluation was taken based on the criteria of parallelism, data processing and ingestion, usability and the availability of ML algorithms. An empirical evaluation of ML algorithm performance, on the railway incidents prediction problem, has been carried out.

All technologies presented similar performance on test data, therefore they can be considered equally efficient on this criteria. Apache Spark shows more suitable for distributed computation of large dataset, although more complex to use, mainly regarding its data processing capabilities. TensorFlow and Pytorch work well for deep learning applications, and the use of Keras significantly reduces the complexity of implementation. Scikit-learn showed the simplest API to use, although not the best alternative for deep learning applications, or for highly distributed computation. The results discussed are preliminary, as they are part of the initial research and investigation on the technologies. Further studies and performance evaluation, including on different applications, will be performed as part of the research project discussed in this document.

## References


Giannini, F., Laveglia, V., Rossi, A., Zanca, D., and Zugarini, A. (2017). Neural networks for beginners. A fast implementation in matlab, torch, tensorflow. *CoRR*, abs/1703.05298.

Mavridis, I. and Karatza, H. (2017). Performance evaluation of cloud-based log file analysis with apache hadoop and apache spark. *Journal of Systems and Software*, 125.

MICS (2019). Ministério da indústria, comércio e serviço. industria 4.0. `http://www.industria40.gov.br/`. [Online; accessed 18-March-2019].

Shi, S., Wang, Q., Xu, P., and Chu, X. (2016). Benchmarking state-of-the-art deep learning software tools. *CoRR*, abs/1608.07249.

Viademonte, S., de Souza, C., Carneiro, N., Junior, J. F., and Lyra, W. (2018). A computational framework for railway incident analysis: from data mining to data visualization. In *AMCIS2018*, New Orleans, Louisiana US.