Implementing Big Data Analytics Projects in Business

Françoise Fogelman-Soulié and Wenhuan Lu

Abstract Big Data analytics present both opportunities and challenges for companies. It is important that, before embarking on a Big Data project, companies understand the value offered by Big Data and the processes needed to extract it. This chapter discusses why companies should progressively increase their data volumes and the process to follow for implementing a Big Data project. We present a variety of architectures, from in-memory servers to *Hadoop*, to handle Big Data. We introduce the concept of *Data Lake* and discuss its benefits for companies and the research still required to fully deploy it. We illustrate some of the points discussed in the chapter through the presentation of various architectures available for running Big Data initiatives, and discuss the expected evolution of hardware and software tools in the near future.

1 Introduction

Big Data has become somewhat of a buzz word in recent years with countless news items and scientific articles appearing in the general or scientific press. In 2007, the IDC (International Data Corporation) analysts [13] reported the explosion of the *Digital Universe*, creating the necessity to use new *measurement units* for big data: the usual Megabyte and Terabyte would soon have to be replaced by Exabyte or Zettabyte (1,000,000 or 1,000,000,000 Terabytes), with the Digital Universe estimated at 161 Exabytes in 2006. This trend was made possible by the development of Internet, digital devices (phones, cameras, sensors in the Internet of Things) and the sharp decrease in prices for storage, computing power, memory, and network bandwidth.

In 2011, Mc Kinsey [26] said that a 40 % growth was to be expected in the amount of data generated each year. In the same report, Mc Kinsey showed that all economic

F. Fogelman-Soulié (✉) · W. Lu
School of Computer Software, Tianjin University, Beiyangyuan Campus, 135 Ya Guan Road, Jinan District, Tianjin 300350, China
e-mail: soulie.fr@gmail.com

W. Lu
e-mail: wenhuan@tju.edu.cn

N. Japkowicz and J. Stefanowski (eds.), *Big Data Analysis: New Algorithms for a New Society*, Studies in Big Data 16, DOI 10.1007/978-3-319-26989-4_6

sectors could profit from big data: for example, the US health care sector could expect $300 billion potential annual value and the US retail sector could expect 60 % increase in operating margins. Reaping such additional value requires new tools and talents, and this has created the new field of *data science*.

Following these reports and other publications, companies have been embarking on Big Data initiatives, but finding many daunting issues on their way.

In this chapter, we want to describe, in as simple and pragmatic a way as possible, what the difficulties are for companies wanting to run Big Data projects. In Sect. 2, we define Big Data; in Sect. 3, we describe the various stages in a Big Data project process and illustrate these in an example from credit-card fraud on Internet; in Sect. 4, we show how companies should store their Big Data in a Data Lake if they want to implement many Big Data projects; in Sect. 5, we introduce the various elements in a Big Data platform and some of the most widely used analytics packages.

2 Big Data Value for Companies

In 2001, Doug Laney (from Meta Group, now Gartner) published a report [24] in which he showed how the rise of e-commerce, in particular, was producing an explosion in data volumes resulting in growing data management challenges. He introduced three important dimensions: volume, velocity and variety (which have come to be known since as the *3Vs*) and discussed possible solutions to handle them.

- **Volume:** in e-commerce, at the time, lower costs of e-commerce channels started to allow collecting increasing data volumes while, at the same time, enterprises were realizing that such data represented an asset and thus wanted to keep it. The costs of storage, however, would soon come to offset the marginal data value gain, so Laney recommended *sampling and limiting data collected*.
- **Velocity:** the increased speed of interactions on e-commerce sites produced a growing constraint on the speed at which data should be ingested and analyzed. The proposed solution to this issue was to develop *architectures* with more bandwidth, caches, and lower latency.
- **Variety:** the most challenging problem that was identified was the large variety of heterogeneous data sources, incompatible data formats, non-integrated data structures and inconsistent data semantics. Various solutions were proposed by Laney, including metadata management solutions and indexing techniques. At that time, data warehousing was deployed more and more widely, so that the solution to variety was viewed in that framework.

Since then, Big Data has become a major news item and a big market for industry: according to Vasanth [34], the market is expected to grow to $53 billion by 2017, with hundreds of billions of dollars potential values in many domains according to McKinsey report [26]. This shows that despite the risks and problems identified in 2001, Big Data has somehow emerged as a big value opportunity.

What allowed this is two-fold: on the hardware side, the "attack of the exponentials" as Driscoll [9] calls it, has seen the costs of storage, CPU and bandwidth over the last decades exponentially dropping, and network access exponentially increasing. What was not cost-effective, or at all possible in 2001, now is, so that all data can actually be collected, stored, and accessed, at low cost [20]. On the software side, tools to handle, store, distribute and mine Big Data have also rapidly developed producing a very dense and complete set of tools [32].

Nowadays, almost 15 years after his original report, Doug Laney, now at Gartner, considers that "adoption of Big Data is simply inevitable" [23]. Today, many organizations are embarking on Big Data projects, but they find many questions on their way. In the next section, we discuss a process for the implementation of projects to derive value from big data.

3 The Process for Big Data Projects

3.1 Machine Learning for Value

Even though Big Data involves many different techniques, Machine Learning is the *major key success factor for delivering value*, and this is because Big Data allows producing better models.

To produce a Machine Learning model, one needs to assemble, from a variety of data sources, a dataset containing a set of observations (for example: customers in marketing applications, patients in health problems …) for which a certain number of variables are available (see Fig. 1). This dataset, in a Big Data setting, will be *deep* with applications involving millions of observations, and *wide* with potentially

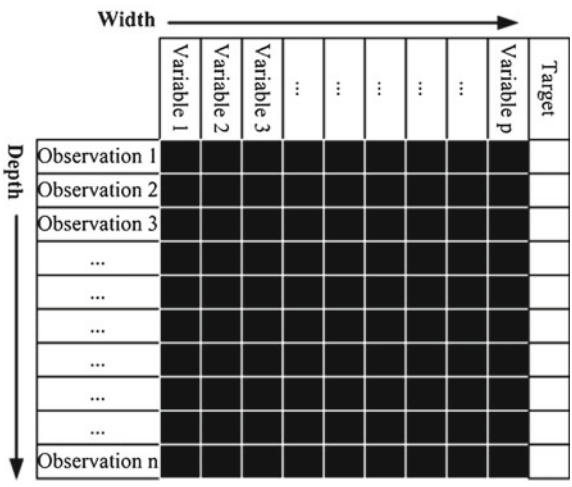

Fig. 1 Dataset for machine learning

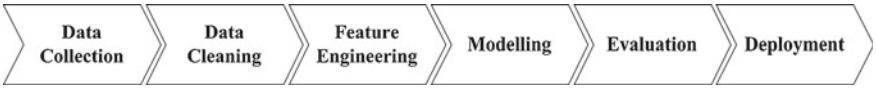

Fig. 2 Big Data process

hundreds of thousands of variables (we will indifferently use the words features or variables). The *depth* might be large, for example, if observations are transactions, but not always: for example, if observations are customers of a Bank, the depth will be limited by the total number of customers, which cannot increase *ad libitum*. A large *width* of course comes from the large variety of data sources included in the dataset and we will see in the following section that width can be further increased through various procedures. Finally, the dataset volume can be seen as the product *depth x width*. In many applications, there is a *target* which we want to predict (for example, whether this customer is going to defect next month to a competitor) and the dataset includes the observations' target values.

3.2 The Process

The process for a Big Data project involves a succession of stages (see Fig. 2):

Data collection: first, data needs to be collected from available data sources. The more data there is at this stage, the best it will be for the final model. Increasingly, open data is available, which can be integrated as very useful data sources.[1]

Data cleaning: data needs to be cleaned to improve quality and consistency. At this stage, the various features must be checked for mistakes (for example, misspellings), deduplicated, reconciled and integrated to produce a unique record (a line in the table of Fig. 1) associated to each observation (see [29]).

Feature engineering aims at producing from existing variables additional computed variables, which could be meaningful for the business domain but hard for a model to learn ("hard" meaning: requiring lots of data, large computation time and a more complex model). Such features could be aggregates on sliding windows (for example, the number of claims for an insurance subscriber in the last 6 months, in the last week …), on geographical areas (for example, the number of accidents on a road segment, a town, a region …) or any variable which makes sense for the particular business. Feature engineering is recognized as the most important success factor for the performance of a Machine Learning model [8]. It usually helps producing models, which are simpler, easier and faster to train, while also providing increased performances for a given algorithm as we will show in an example below. In 2007, at a time when analytics Big Data was starting to pick-up, [10] indicated that less than 5 % of analytics projects were using more

[1]For example: https://www.data.gouv.fr/en/, http://open-data.europa.eu/en/data/, http://publicdata.eu/, http://www.data.go.jp/, http://dataportals.org/.

than 1,000 features in their model, while about 50 % used less than 40 features. Since then, things have changed a lot. In the various projects we have seen, generating an additional 1,000 features is common, but some projects generate a lot more (a few tens of thousands). Out of these features, about 80 % will be standard (time or space aggregates, ratios…), and 20 % will be domain-specific. However, since this stage is extremely time-consuming [3], it is interesting to invest in some systematic way to engineer features. Most recent Machine Learning packages are investing in that area (see section Architectures for Big Data below). Note that additional features can also be obtained from outside sources, such as open data sources or private data obtained from partners or data providers. Data from very different sources and semantics will bring more additional value: this is what is reflected in the Variety of data. Of course, increasing the number of features also increases the Variety, and thus the Volume of the dataset.

Modelling: At this stage, we have assembled a dataset with many features (as shown in Fig. 1). To generate a model, we will use one of the many existing Machine Learning algorithms (see for example the book [16]). Choosing from this very large collection of algorithms might seem hard. However, the problem is easier with Big Data: all recent developments in Big Data, [8, 15], have shown that simple models with lots of data are always better than complex models on less data. Hence, one strategy is to choose one relatively simple algorithm, for example logistic regression, and work at increasing data volumes: engineering features is the simple way for that. Simpler algorithms are also easier to explain than more complex ones, so that sometimes one will prefer a simple logistic regression model to a more performant algorithm, such as, for example, random forests because interpretability is much better for the former.

Note that feature engineering produces features which are usually correlated. Hence the algorithm selected should not be sensitive to correlated variables.

Evaluation: Producing a model from data is an iterative process: datasets will be progressively enriched to produce increasingly wider feature sets. Each time, a machine learning algorithm will be trained to produce a model. Usually not all the available dataset will be used, but only a representative sample drawn from it. Then the sample is separated into two parts: the learning sample is used to produce the model and the validation sample (the rest of the sample) is used to validate the model. In the learning phase, one tries to produce the model which best fits the data in the learning sample. In the validation phase, one verifies that the model properly generalizes to new data it has not seen during training: if it does not, it is because it overfits (see Fig. 3).

Producing a model which fits the data is (relatively) easy; producing one which generalizes is much harder. However, generalization is actually what is required if one wants to further use the model [8].

Vapnik's Statistical Learning Theory [33], for example, provides a framework to monitor the learning process so as to achieve generalization. The framework can be summarized as follows:

- Models are restricted to a given family, for example polynomials of degree d, trees of a certain depth, ridge regression, regularized regression etc. The only constraint on the models' family is that it has finite capacity (or Vapnik—Chervonenkis dimension: this is a measure of the complexity of the family).
- The models in the family are explored so as to select the best compromise between learning error and generalization error (errors respectively on the learning and validation samples).
- In the Structural Risk Minimization (SRM) framework, embedded model families are explored, starting with low capacity and progressively increasing capacity, until generalization error starts increasing.

The capacity of the final model is just large enough to produce a good compromise between learning and generalization errors, but not too large, when it would produce overfitting and a larger generalization error (Fig. 3).

In the course of the learning process, the number of features is increased until no further performance increase can be obtained: Fig. 4 shows the typical behavior expected when increasing the number of features. It could take thousands of features before the performance flattening-out appears, depending on the problem. However, not all the features will be retained in the final model: one will try and select the features most significant for the model, sometimes trading a marginal performance increase with a smaller number of features (i.e. a simpler model with lower capacity). In [10], it was reported that while only about 25 % projects used less than 20 features in the original model, about 50 % did in the final version. Explaining a model with more than 20 features is hard and not very intuitive, which is why most final models these days only incorporate a few dozens features.

It is important to note that one set of optimal features at one time might not be optimal anymore later on, when the situation has changed. When retraining a model in later periods, one should thus always start anew from the full set of features.

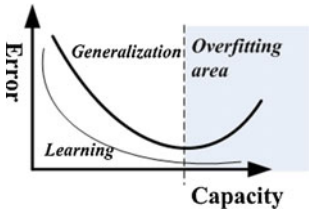

Fig. 3 Learning and generalization error as capacity increases

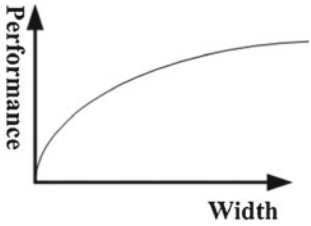

Fig. 4 Performance increases with more features

One should also note that the choice of the optimal features set is not done a priori, or through "expert" knowledge: with big data, it is not reasonable to expect that experts would look at thousands of features and manually evaluate their significance; we need an automatic process for selection. The feature selection process is thus only driven by the data and the algorithm used for producing the model. One should expect different models to produce—usually slightly—different features sets.

A very large number of features can be argued against on the basis of the *curse of dimensionality*: with a fixed-size dataset, the observations occupy an increasingly smaller portion of the input space and the space where the model will apply is increasingly larger when the number of features increases. Generalization thus becomes harder [8]. This issue is lessened in the Big Data situation, since, then, we can increase the size of the sample dataset used for producing the model, while, as always, controlling the capacity of the models' family. However there is generally still a trade-off between the number of additional features we want to use and the increase in sample size necessary for producing the model. At one point, this trade-off will require to stop increasing the number of features, either because the maximum size of the dataset has been reached, or because the performance gain will be too small.

Note that, for many model families, capacity is directly related to the number of features: for example, the family of polynomials of degree 1 in p variables has capacity $p + 1$, while for polynomials of degree 2, capacity is in p^2. So, in that case, increasing capacity through SRM amounts to progressively increasing the number of variables. If one wants to control capacity without restricting the number of features, one would need a family of models for which capacity can be controlled otherwise.

For example, the following family F_Θ:

$$F_\Theta = \{f(x, W) = sign\left(\sum_{j=1}^{p}(W_j \times x_j) + 1\right) / ||x|| \leq R, ||W||^2 = \sum_{j=1}^{p} W_j^2 \leq \Theta\}$$

has capacity $C_\Theta : C_\Theta \leq \min(int(R^2\Theta^2); p) + 1$, which can be controlled through a parameter Θ independently of the number of features p.

Thus, in Big Data situations, one can increase the number of features, provided the size of the training sample is large enough and the model capacity is systematically controlled to avoid overfitting.

Deployment: When the final model is ready, with its reduced number of features, it can be deployed to produce predictions on new data. For deployment, one will always prefer simple models, especially when real-time is required. The example of Netflix is famous: after paying 1M $ to the winners of a challenge[2] which required to improve Netflix algorithm's performance by 10 %, Netflix finally decided not to put in production the solution delivered to them by the winners: "the additional accuracy gains that we measured did not seem to justify the engineering effort needed to bring

[2] http://www.netflixprize.com/.

them into a production environment" [1]. Yet it was 10 % additional accuracy! In conclusion, at deployment time, simplicity should always be the goal.

3.3 An Example

Let us present an example from a concrete application to illustrate how feature engineering, especially with different semantics, can increase performance. This was a collaborative project [11] on credit-card fraud detection on Internet, funded by the ANR (the French National Research Agency). Credit card fraud on Internet is a massive, fast-growing phenomenon, in large part at the hands of organized crime. Merchants and banks are faced with the need to implement solutions to detect it as rapidly as possible. In France, GIE Cartes Bancaires is a Group which has been commissioned by French banks to handle the payment process for all transactions made by credit card holders (Visa, MasterCard) from French banks. In 2013 GIE CB managed over 10 billion transactions (up 3.4 % compared to 2012), with 584.5 million over the Internet, made by 61.7 million CB cards, totaling 524.3 billion euros [14].

The available transaction data is not very rich: in particular, we do not have information about the card holder (name, address, age, gender ...) or the purchased product (product type, number of products purchased ...), information that only the bank (card data) or merchant (product data) would possess. In contrast, we have all the transactions made by the card-holders on Internet, which would not be the case for a bank (which would only "see" the transactions made by the holders having a card in that bank) or a merchant (who would only "see" the transactions made by the card-holders buying at that merchant). This represented about 50 M transactions per month, made with the card not present (internet or phone). For each transaction we have:

- Information on the card: card number, expiration date, issuing bank ...
- Merchant information: username, SIRET, country, merchant business, merchant bank (acquirer) and country of the bank, terminal used ...
- Transaction information: date of the transaction (local and GMT), amount (in local currency and in euro).

Once a card is blocked, we obtain the blocking date and the reason for the blocking (note that this label can come several months after the fraud occurred). The objective of a model is then to classify a transaction as fraudulent or not fraudulent: this is called *fraud detection*. Transactions classified as fraudulent will be transferred to an investigation team, which will investigate all transactions recently made by the corresponding cards.

From these transactions, we will compute various features, which are aggregates characterizing the history of each card or merchant:

- *Cards Aggregates* at date T: over a sliding window (of various lengths: day, week, month) ending at T, features are computed for each card such as the number and average number of transactions, total and average transaction amount, the difference between the number of transactions at time T, respectively the amount, and the average number of transactions in the window, respectively the average amount etc.
- *Merchants Aggregates* at date T: over a sliding window (of various lengths: day, week, month) ending at T, features are computed for each merchant such as the number and average number of transactions, total and average transaction amount, number of fraudulent transactions and total amount of fraud, difference between the number of fraudulent transactions at time T, respectively the amount, and the average number of fraudulent transactions in the window, respectively the average amount.

We thus obtain 666 aggregates as shown in the first lines of Table 1. We then compute *social features* in the following way:

- We first compute the bipartite network made from cards and merchants nodes, linked when there is a transaction by the card at the merchant (using all the transactions in one month). Through the usual technique to project a bipartite network into two unipartite networks [4, 35], we derive a Cards network and a Merchants network: two cards (resp. merchants) are linked in the Cards (resp. Merchants) network if they have purchased from at least k same merchants (resp. have been visited by the same k' cards), where k and k' are some fixed parameters.
- Then, we compute, for each node (card or merchant), a number of variables in the Cards or Merchants networks, such as the degree of the node, the index and size of its community in the unipartite network (see [4]). For merchants, we also compute the average amount and number of cases of fraud successful in his community/his first circle, the average number of distinct fraudulent cards in his community/his first circle, the average number of transactions accepted/rejected in his community/his first circle etc. This gives 195 *social variables* for cards and 99 for merchants, in addition to the already defined variables (as shown in Table 1).

We now compute three models with an increasing number of features: the baseline model uses the 37 original variables, the second model uses in addition the 666

Table 1 Number of features used for the fraud detection model

Variables	Number
Original GIE variables	37
Card aggregates	300
Merchant aggregates	366
Card social variables	195
Merchant social variables	99
Total	997

Table 2 Performance for fraud detection with an increasing number of variables

Model	Recall (%)	Precision (%)	No variables
Baseline	1.40	8.18	37
Baseline, agg.	9.13	19.00	703
Baseline, agg., social var.	9.09	40.58	997
19 Seg. baseline	5.09	28.21	37
19 Seg. baseline, agg.	7.38	28.82	703
19 Seg. baseline, agg., social var.	16.46	60.89	997

aggregates and the final model uses all 997 variables. We evaluate the performance obtained by these models using Recall and Precision, which have been extensively used in the literature for evaluating classification, information retrieval or recommender systems [17]. *Recall* represents here the portion of fraud captured by the model at a certain threshold on score and *Precision* the proportion of truly fraudulent transactions among those classified as such by the model. The threshold on the score produced by the model is chosen so as to generate a number of cards declared fraudulent compatible with the staff available to further investigate them. This threshold cannot be disclosed for confidentiality reasons. The performances in terms of Recall and Precision at that threshold on score are shown in Table 2 (first three lines). The algorithms used were ridge regressions, regularized through a Vapnik's scheme [12].

As can be seen from the first three line of Table 2, the increase in variety (produced by the increase in the number of features) has a very large impact on Precision, which is multiplied by a factor of about 2 with the addition of aggregates, and again with the social variables. Recall is increased by a factor of about 6 with the addition of aggregates, but slightly decreased with the social variables. So, depending upon the objective of the user, the model with all features will be preferred to the model with aggregates only if Precision is more important than Recall.

One can further increase performance by using a segmented model: when datasets are very large, and the analyzed phenomenon is heterogeneous, one can often improve performance by first identifying homogeneous regions, and then performing a local analysis on each. This will often be the case in Big Data problems. Note that segmentation also has the consequence that each segment will be smaller than the original dataset, which might shorten model computing time, while potentially preventing using too many features (if the depth is too small).

- First we implement a segmentation of cards, supervised by the fraud label (using a supervised k-means): we produced 19 segments.
- For each segment, we compute a model for the three feature sets described in Table 1. When a new transaction needs to be analyzed, we pass it through its card segment model.

As can be shown in Table 2 (last three lines), performances are significantly improved. This time, aggregates and social variables increase Recall, while only

social variables increase Precision. This is an illustration of a common finding: it is not possible to tell a priori which feature is going to be significant and bring value. The only way to know is to build a model and look at its performance.

As can be seen from this example, feature engineering can lead to very significant performance increases. This is why, in any Big Data project, the time spent on feature engineering is by far the most important, while the time spent actually producing the model is short: [8] indeed noted that "very little time in a machine learning project is spent actually doing machine learning".

3.4 Big Data Skills

Another discussion branches out from this last remark, related to the importance of statistics skills in Big Data. While it is obvious that Machine Learning relies heavily on statistics, Big Data projects require a set of varied skills, which have come to be known as *data science* [5]. These skills are three-fold: statistics indeed for building, evaluating, analyzing the models; IT to collect data, produce features and deploy models; and obviously business knowledge to correctly frame the problem, identify business-critical features, evaluate the models' performance, and, finally decide on the value of putting the model in production.

As we have seen, the time spent in a project on feature engineering is, by far, the most critical success factor for final performance, which is why IT skills are so important for the success of a Big Data project. But as Fig. 5 shows, all three sorts of skills are required and a total lack of statistics skills may put the IT specialist in a danger zone of producing models which are not valid or which exhibit spurious correlations: the discussion on the dangers of data mining in [21], for example, suggests useful caveats on the uncareful use of data mining.

It should also be clear at this point that Machine Learning used for data science is not the same as pure statistics. Some of the crucial elements necessary for Big Data are just not part of the usual Statistics corpus: these include, for example, learn-

Fig. 5 Big Data skills (from D. Conway's Data Science Venn diagram)

ing and generalization, feature engineering, explainability and capacity for scalable deployment. Another characteristic important in Statistics is absolutely missing in Machine Learning: there is no reasonable assumption which can be made on the data distribution and all the techniques implemented must be agnostic with respect to such assumptions.

Because data science requires a new sort of assorted skills, data scientists are in strong demand. Davenport et al. [6] said that data scientist would be "the sexiest job of the 21st century" and Mc Kinsey [26] predicted that, in the US, demand for data scientists would be 50 to 60 % larger than supply in 2018.

4 The Data Lake

In this section, we will see how companies usually proceed when they want to implement Big Data projects. Obviously, with large companies, that endeavor is global: you do not want to do one project, but many. Companies want to become an analytics competitor [7]: best data and best tools make for the best decisions. Companies thus need to turn their data into an asset.

Of course, all large companies already have multiple data sources in-house to begin with. These data are in silos, application and domain-dependent, and usually contain duplicate, inconsistent versions of the same information. One of the main reasons for this state of affairs is that many large corporations have grown through multiple mergers and acquisitions where each organization merging-in brought its own data system. When starting a Big Data initiative, a company thus has to face a *big data integration problem*, starting with its internal data, continuing with external data (Open Data, partners' data, acquired data …), before finally getting to real Big Data and the ability to exploit it.

Traditional enterprise integration techniques, such as data warehousing or, more generally, *Master Data Management* (MDM) aim at linking all data files into one—*the master file*—providing a *unique version of the truth*. This ensures consistency throughout the various system architectures and applications enterprise-wide and allows sharing data between the various entities in the corporation. However, obtaining a master file requires *Extracting, Transforming and Loading* (ETL) data from the various heterogeneous data sources; developing metadata to describe the data and producing a general data model. Many ETL tools exist on the market (for example Talend Master Data Management, Informatica MDM, IBM InfoSphere, DataStage …). But ETL processes are known to be hard to implement, harder still to maintain and altogether very expensive. Some authors [2] have claimed, for example, that data warehousing projects fail in as much as 50–75 % of the cases. One of the main reasons for such failure is the inability to maintain a manageable scope for the data warehouse, even more so when data sources dynamically change. With Big Data, the problem is going to be even harder than for "simple" warehousing. So we could expect failure rates of the same magnitude at best.

However, MDM might be chasing the wrong rabbit: the master file and data model are efficient for the activities existing or *already defined* in the organization at the time: relational databases and data warehouses are structured in hierarchies and dimensions adapted for the analysis planned at the time of their development. When new needs come up, it is necessary to modify and sometimes completely rebuild the structure to fit the new requirements. With the advent of Big Data, this will be the typical situation: a very wide scope, no known-in-advance goals or analyses, and very dynamic, constantly emerging data sources. A *fixed static data structure cannot* do and data integration on a large scale will still need to be done.

It is to cope with this issue that the concept of *Data Lake* has appeared in the recent years [31]. A Data Lake is a repository of all data collected by an organization, where the data is stored in its *original raw form*. Because no a priori structure or data model is imposed at collection time, all further usage should be possible without having to modify a pre-existing model.

Of course, for any given project, the usual process (described in Sect. 3) will be executed: data will be collected, primarily from the Data Lake and also from other data sources if significant (open data for example). At the cleaning stage, the various data returned by the Data Lake or other sources will need to be deduplicated and reconciled. So this work on data sources reconciliation is not eliminated by the Data Lake: it is postponed to project time when it is needed for a particular objective and restricted to a limited set of data sources. Results of the reconciliation of two data sources can be stored in the Data Lake, as metadata, for further projects which use the same data sources. In this way, data models will emerge progressively from projects developed over time, instead of being imposed at collection time. Hence, data integration is no longer an issue. However, data access is still to be handled: careful tagging and metadata management are required to allow the users to retrieve data interesting for their project, maybe years after the data was collected.

However, more research will certainly be needed to fully exploit Data Lakes: as described in [19], Data Lakes at the present time can only be used by data scientists with significant programming skills. Retrieving data from a potentially enormous (and growing) Data Lake will require adequate skills until adapted tools for navigating the Data Lake are made available. At the present time, it is not possible to simply query a Data Lake, as usual when using SQL for example, since it contains data sources in all possible formats.

As a conclusion, we think that Data Lakes will be increasingly implemented within organizations wishing to implement a succession of Big Data projects, but that additional research will need to take place to help users really, and simply, navigate the Data Lake.

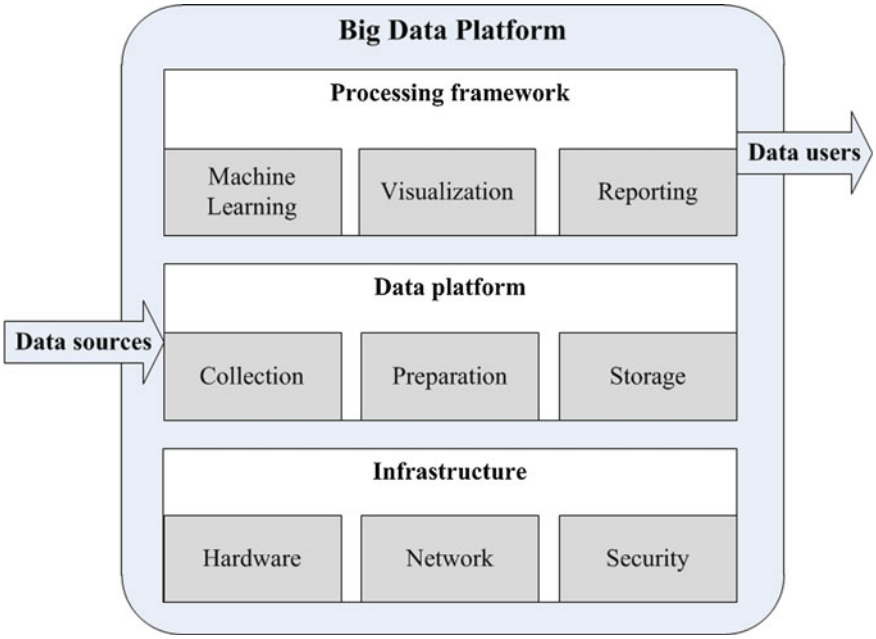

Fig. 6 Big Data platform

5 Architectures for Big Data

5.1 The Big Data Platform

We now turn in this section to a brief description of the possible architectures for Big Data, on both the hardware and software sides. A Big Data platform has the following layers (Fig. 6):

Infrastructure: this includes the hardware, the network, and security components. Obviously, the major requirement for this layer is to allow handling Big Data volumes. Historically, the solution has been to use bigger and bigger servers to handle the increase in data and required processing speed: this is called the *scaling-up* mechanism. However, big servers are not cost-effective when compared to standard PC or low-end commodity machines: a machine with four times the power of a PC costs more than four times the cost of one PC. This is one of the reasons why Google, Yahoo, Amazon, Microsoft and others started to hook together clusters of PC, distributing their data on the machines. With the introduction of Hadoop [27], distributed file system (HDFS) and MapReduce, it became possible to store data in a distributed fashion and run large-scale distributed data processing applications with code distributed where the data is [22]. A cluster of low-end servers with HDFS files is thus very common for storing data, such as, for example a Data Lake. The added

benefit of such an architecture is the scaling-out (or horizontal scaling) property: one can elastically grow the cluster by adding new low-end servers, as data volumes grow. This is a lot cheaper than buying up-front a big server, which will, at one point anyway, become too small and will need to be replaced.

In-memory servers, with very large RAMs, have recently appeared on the market: with 512 GB to a few TB RAM (for example, the Bullion machine[3] has up to 24 TB RAM), it becomes possible to collect data, load it in memory and execute all the modelling in-memory, thus very significantly reducing computing times. Apache Spark is a new distributed computing framework which offers in-memory primitives through its Resilient Distributed Datasets (RDDs). In a way, Spark brings a solution with the better of two worlds: Hadoop and in-memory. Performances are much faster than with Hadoop MapReduce.

Data Platform: this layer includes the various tools necessary to access and collect data, to clean prepare and store it. As we have described in the previous section, data is heterogeneous, comes from many sources and in multiple formats. Tools are thus needed to collect data: the ETL family, as well as various data scraping mechanisms, can be used to extract the data. As we have seen before, it is better not to transform data at collection time, but maybe to only retrieve metadata and tags associated to it. Data will be stored in a platform which allows scaling-out: when new data comes in, we need to be able to elastically add to the storing solution to make room for it. This is why, most of the time, data will be stored on a Hadoop cluster.

Very often, data will be stored in NoSQL data bases (Not Only SQL) which offer different structures than relational databases, such as key-value, document, or graph.[4] These databases offer horizontal scaling and may lead to much better performance than relational databases depending on the domain.

Processing framework: this layer includes the tools necessary for visualizing the data, developing machine learning models, and building reports. We will see that there exist a wide range of software packages for running analytics projects on big data. The main challenge, of course, is to be able to scale when handling the data volumes: visualizing small data sets is relatively easy and one can play with many different visualization styles. However, with increasing volume and variety, the requirement for scalability will strongly restrict the visualization effects to the simplest cases.

Note that the Hadoop/MapReduce environment is more adapted to batch processes, and not so much to the iterative development of machine learning algorithms. With the release of Hadoop 2, new tools have been made available, more adapted to machine learning. In particular, Spark allows building specialized Big Data tools such as MLib for machine learning and GraphX for graph processing. Hadoop 1 and 2 are supported commercially by companies such as HortonWorks and Cloudera.

[3] http://www.bull.com/download/bullion/B-bullion-2014-enWeb.pdf.

[4] https://datafloq.com/big-data-open-source-tools/os-home/.

5.2 Analytics Software Packages for Big Data

There exist many analytics software packages for big data, a lot being open-source [25]. However, surveys [28, 30] show that most data scientists use but a limited set of tools: R is by far the most wide-spread tool and language, while Python for programming is picking up.

The first generation of tools, SAS and IBM-SPSS, tended to offer a wide range of algorithms, in a proprietary framework. The second generation, such as, for example SAP-KXEN, focused on helping automate the data mining process and opening it to business users, through just one algorithm, a regularized regression [12]. As discussed in Gartner's report [18], SAS and IBM-SPSS still dominate the market with a large installed base. Their position, however, will presumably erode as a new generation of tools gets access to the market.

This last generation focuses on Big Data and attempts at covering the entire Big Data project process described in Sect. 3.2. For example DataRobot[5] focuses on the modeling and deployment stages, automatically generating and comparing thousands of models from various open source libraries (R, Spark MLlib, Python-based scikit learn[6]); Dataiku (see footnote 5) offers through its Data Science Studio tools to load and enrich data, then allows to run models from scikit learn, returning the best performing one; Palantir (see footnote 5) offers tools to "*Integrate, manage, secure, and analyze all of the enterprise data*", in particular, Palantir has a strong feature engineering tool which helps automate the generation of standard features.

As can be seen, the new generation tools do not try to develop their own machine learning algorithms (as did SAS and SPSS) but, instead, call upon open-source libraries (such as MLlib and scikit learn[6]) which are very actively enriched by active communities.[7] Notice that these two libraries run on different architectures: scikit learn, being based upon Python, runs best on an in-memory server; while MLlib runs with Apache Spark, and thus can be executed on any Hadoop 2 cluster. With the recent development of Spark, MLlib has developed very strongly, taking over the Mahout[8] library which was running on Hadoop MapReduce. As a consequence, there is an effort by the Mahout community to build future implementations on Spark.

It should be expected that more tools will appear in the near future to build upon these healthily competing libraries.

[5] http://www.datarobot.com/, http://www.dataiku.com/, https://www.palantir.com/.

[6] https://spark.apache.org/mllib/ is Apache Spark's machine learning library; http://scikit-learn.org/ is a machine learning library in Python.

[7] https://github.com/apache/spark; https://github.com/scikit-learn/scikit-learn.

[8] http://mahout.apache.org/.

6 Conclusion

We have described in this chapter why and how companies implement Big Data projects. The field calls upon a wide variety of techniques, tools and skills and is very dynamically developing. Even though we tried to cover most of the practical issues faced by companies, many topics are still missing here: most notably the privacy and security issues, which would deserve a full chapter of their own.

It is our belief that, in the near future, companies will continue investing in Big Data and the results will bring productivity growths in all sectors of the economy.

References

1. Amatriain, X., Basilico, J.: Netflix Recommendations: Beyond the 5 stars. Netflix Techblog. (6 April 6 2012)
2. Amin, R., Arefin, T.: The empirical study on the factors affecting datawarehousing success. Int. J. Latest Trends Comput. **1**(2), 138–142 (Dec 2010)
3. Anderson, M., Antenucci, D., Bittorf, V., Burgess, M., Cafarella, M. J., Kumar, A., Niu, F., Park, Y., Ré, C. & Zhang, C.: Brainwash: A Data System for Feature Engineering. CIDR'13 (2013)
4. Chapus, B., Fogelman Soulié, F., Marcadé, E., Sauvage, J.: Mining on social networks. In: Gettler Summa, M., Bottou, L., Goldfarb, B., F. Murtagh (eds.) Statistical Learning and Data Science, Computer Science and Data Analysis Series. CRC Press, Chapman & Hall (2011)
5. Conway, D.: The Data Science Venn Diagram (2013). Blog. http://drewconway.com/zia/2013/3/26/the-data-science-venn-diagram
6. Davenport, T.H., Patil, D.J.: Data Scientist: The Sexiest Job of the 21st Century. Harvard Bus. Rev. 70–76 (Oct 2012)
7. Davenport, T.H.: Competing on analytics. Harvard Bus. Rev. **84**, 98–107 (2006)
8. Domingos, P.: A few useful things to know about machine learning. Commun. ACM **55**(10), 78–87 (2012)
9. Driscoll, M.: Building data startups: Fast, big, and focused. Low costs and cloud tools are empowering new data startups. O'Reilly Radar (August 9, 2011)
10. Eckerson, W.W.: Predictive Analytics. Extending the Value of Your Data Warehousing Investment. TDWI Best Practices. Report. **Q1**, 2007 (2007)
11. Fogelman-Soulié, F., Mekki, A., Sean, S., & Stepniewski, P.: Utilisation des réseaux sociaux dans la lutte contre la fraude à la carte bancaire sur Internet. In: Bennani, Y., Viennet, E. (eds.) Apprentissage Artificiel & Fouille de Données. Revue des Nouvelles Technologies de l'Information, RNTI-A-6. Hermann, pp. 99–119 (2012) (in French)
12. Fogelman Soulié, F., Marcadé, E.: Industrial Mining of Massive Data Sets. Mining massive Data Sets for Security. In: Fogelman-Soulié, F., Perrotta, D., Pikorski, J., Steinberger, R. (eds.) Advances in data mining, search, social networks and text mining and their applications to security, pp. 44-61. IOS Press. NATO ASI Series (2008)
13. Gantz, J.F.: The Expanding Digital Universe. IDC White Paper (March 2007)
14. Groupement des Cartes Bancaires CB: Activity, Report (2013)
15. Halevy, A., Norvig, P., Pereira, F.: The unreasonable effectiveness of data. IEEE Intell. Syst. **24**(2), 8–12 (2009)
16. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning (vol. 2, no. 1). Springer, New York (2009)
17. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. (TOIS) **22**(1), 5–53 (2004)

18. Herschel, G., Linden, A., Kart, L.: Magic Quadrant for Advanced Analytics Platforms. Gartner Report G00270612 (2015)
19. Heudecker, N., White, A.: The Data Lake Fallacy: All Water and Little Substance. Gartner Report G00264950 (2014)
20. Hilbert, M., López, P.: The world's technological capacity to store, communicate, and compute information. Science **332**(6025), 60–65 (2011)
21. Leinweber, D.J.: Stupid data miner tricks: overfitting the S & P 500. J. Investing **16**(1), 15–22 (2007)
22. Lam, C.: Hadoop in action. Manning Publications Co (2010)
23. Laney, D.: Big Data's 10 Biggest Vision and Strategy Questions. Gartner Blog (2015)
24. Laney, D.: 3D Data Management: Controlling Data Volume, Velocity, and Variety. Application Delivery Strategies, Meta Group (2001)
25. Machlis, S.: Chart and image gallery: 30+ free tools for data visualization and analysis. Computerworld (2013)
26. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A.: Big data: The next frontier for innovation, competition, and productivity. Report, McKinsey Global Institute (2011)
27. Olson, M.: Hadoop: Scalable, flexible data storage and analysis. IQT Quart **1**(3), 14–18. (Spring 2010)
28. Piatetsky, G.: KDnuggets 15th Annual Analytics, Data Mining, Data Science Software Poll. KDnuggets (2014)
29. Rahm, E., Do, H.H.: Data cleaning: problems and current approaches. IEEE Data Eng. Bull. **23**(4), 3–13 (2000)
30. Rexer, K.: 2013 Data Miner Survey. Rexer Analytics (2013)
31. Stein, B., Morrison, A.: The enterprise data lake: Better integration and deeper analytics. PwC Technology Forecast: Rethinking integration. Issue 1 (2014)
32. Turck, M.: The state of big data in 2014 (chart). VB News (2014)
33. Vapnik, V.: Estimation of dependences based on empirical data. Springer. Information sciences and Statistics. Reprint of 1982 Edition with afterword (2006)
34. Vasanth, R.: The Rise Of Big Data Industry: A Market Worth 53.4 Billion By 2017 ! Dazeinfo (2014)
35. Zhou, T., Ren, J., Medo, M., Zhang, Y.-C.: Bipartite network projection and personal recommendation. Phys. Rev. E **76**(4), 046115 (2007)