Data Science End-to-End Project

(Part 2: the Data Analysis Backbone)

1. Motivation

With this second part we aim at developing **one** data analysis backbone for a certain data analysis task on top of the data management backbone you built in the first part of the project: Part 1: the Data Management Backbone.

2. The Data Analysis Backbone

Remember the data management backbone is common for all the organisation. However, the data analysis backbone typically depends on a project or a certain type of analytical task. That is, an organisation may generate several data analysis backbones.

In this second part, aligned with your project context, you first need to explicitly state your analysis question. You cannot start this backbone without a clear analytical objective. Example of analysis questions for an insurance company follow:

- What are the most relevant features from the customer-related data available in order to identify customer churn?
- Predict, via the contact channels used by customers and their last actions in the last months, the degree of satisfaction of a customer.
- Classify the type of costumers we have according to the type of insurances got and the engagement of their relatives (i.e., how many relatives also have insurances with us).
- Etc.

Pay attention to the data you have and devise a clear analysis question that fits your project. Be sure to discuss it with your project supervisor.

Once the analysis question is clear, let us proceed with DataOps-related aspects.

Create yet another different database with the **analytical sandbox** required to perform your data analysis. When creating the analytical sandbox you will need to decide what is the appropriate data model to store the data in order to facilitate its further analytical processing. Typically, you want dataframes or matrixes / tensors here. The analytical sandbox is then stored in a so-called analytical database (e.g., DuckDB or a vector database).

Besides changing the data model, the scripts to generate the analytical sandbox select a subset of the columns and files available in the exploitation zone artefacts. Here, only data to be used for the analysis is considered.

Next, you must generate, at least, one training and validation dataset in the **feature generation** zone. To do so, first, you must decide what features you are going to include in these datasets. Generating features is time-consuming and one of the most relevant tasks in the data analysis backbone. Nevertheless, since ADSDB is not meant to cover data analysis content, it is fine if you conduct simple mappings between variables to features (e.g., birthdate to age). In any case, you will need to prepare such variables to meet the model requirements (e.g., discretize, encoding, normalize, etc.). Labelling is also part of this zone (for supervised solutions). Thus, most

probably, you will need to create four scripts in this zone: feature generation (from attributes to features), data preparation (to prepare the features for the algorithm chosen), labelling (if required) and generate the training and validation datasets. Realise that the order matters between them! So properly orchestrate them to facilitate maintenance.

Model **training**, **validation** and **visualization** of results can be conducted in R, Python libraries or third-party software like SAS. Python and R are recommended for this project, but up to you to decide what libraries to use. Again, ADSDB does not mean to cover the analytical part and we will not focus on the correctness of the algorithms chosen to tackle the data analysis chosen for the project (yet, we expect a minimum level of rigor as of learnt at this stage of the master).

Constraint 8. The scripts to generate the analytical sandbox, feature generation, model training and validation zone must be implemented in notebooks and organized in subfolders (e.g., featureGeneration / dataPreparation).

Constraint 9. Following the idea of the first part of the project, create notebooks to scrutinize each zone. In these notebooks you must include the schema and profiles of the analytical sandboxes created, and the training and validation datasets. Of course, depending on your project you may need further profiling.

Importantly, pay special attention when profiling the training and validation datasets to showcase your posterior data analysis will not be biased/useless (this is a starting point for *explainability*, a key aspect in DataOps). For this reason, in the document, you must include a discussion about the training/validation sets profiling and their validity for the analysis at hand.

?

Constraint 10. Create a notebook to train and generate models. This notebook must contain information about the algorithm hyperparametrization and the metrics used to validate the model (e.g., accuracy). In the document, discuss the algorithm chosen to create the model and its adequacy for the problem at hand. In this project it is not important to obtain good results in the evaluation metrics (we are not evaluating the value of the model). Importantly, this notebook must trace models: i.e., store the model and relate it to the training and validation datasets used to create it.

2.1 Operations

You should now include the data analysis backbone in your operations. To do so, follow the same instructions as per the first part of the project and include the new artefacts created for the data analysis backbone in your orchestration.

Additionally, you must create a run-time app in operations. Typically, you need to create a simple app that reads new data, prepares it to be input into the model created and prints the output of the model.

3. Advanced Topic

The data management and analysis backbones generate the skeleton of a data science project. However, further iterations and needs (that for sure will come) will complicate, bit by bit, your DataOps solution.

To exemplify it, in this second part of the project, we want you to dive in some relevant yet, typically overlooked aspects specific of Data Science projects and include it in your project.

For this part of the project you must choose one of the following topics:

- Data Discovery,
- Data Quality,
- Entity Resolution,
- Feature Selection

For the chosen topic, you must learn the basics about it and implement a basic solution tackling it in your project. Let us discuss first the topics and choose one.

Note: you may propose a topic to the lecturer additionally to these ones. But be sure to get green light from the lecturer before going for it.

The Topics

Find below a description of the topics proposed.

Data Discovery

A key aspect of Data Science is to cross as many data features (roughly speaking, variables) as needed to learn powerful mathematical models that predict or classify events of interest (e.g., purchasing / trading stocks, determine if a person is at risk of suffering a certain disease, etc.).

Thus, an essential aspect of Data Scientists is to gather relevant data sets for the problem at hand. Indeed, the more data sets and variables you gather, the more informative data features you will likely identify to learn robust, generic models not falling into overfitting. A model overfitting occurs when the training data used to learn the model is not representative of general cases and therefore, it is too narrow and specific as to learn a robust model out of it.

The only solution to avoid overfitting is to gather as much data as possible. The more instances and variables you gather, the more likely to build a training data set from which to learn a good model. To accomplish this idea, in Data Science we rely on Data Lakes. A data lake is a huge repository of data where companies or organizations organize their data (their own or obtained from third parties) so that Data Scientists have access to a wealth of data to facilitate building better models.

Data Discovery is a key concept for Data Scientists when exploring huge repositories storing highly heterogeneous data sets from different data sources. We call a data source any source of digital information. For example, social networks (e.g., data obtained through the Twitter API), open data (e.g., Barcelona Open Data platform), internal databases (e.g., the stock database), sensors (e.g., a temperature sensor), etc. Data Discovery is the task allowing Data Scientists to identify complementary data sets leading to Data Augmentation. We say an instance of data (i.e., an instance of a data set) can be augmented with data from another data set if there are common attributes from which to cross such data. For example, consider a data set of *customer(id, name, surname, address, city)*. We might be willing to augment this available data with statistics of the neighbourhood where this person lives as well as data related to its leisure habits coming from Instagram. To do so, we might augment this data set by crossing it with Idescat (the Catalan Office of Statistics) and Instagram. But to do so, realize you need to identify the neighbourhood of that customer from her address and then access the data related to that neighbourhood in Idescat (e.g., average age, average income, or any other variable of interest). Similarly, by combining her name and surname we may identify the Instagram account of that customer and extract information from there about cities visited, etc.

Unfortunately, though, discovering related data in a Data Lake is not easy. Data sets coming from different data sources have syntactic (e.g., different formats such as JSON, CSV, relational, key-value, etc.) and semantic (e.g., customer vs. client) heterogeneities that sometimes make it hard to realize these datasets have complementary data that may lead to Data Augmentation.

Thus, we may summarize this discussion with the following statement: "*Data Discovery is the automated set of tasks that facilitate identifying relevant and complementary data sets for a given analytical task from a huge repository of data sets*".

Seminal bibliography:

- R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker. *Aurum: A data discovery system.* 2018 IEEE 34th International Conference on Data Engineering (ICDE 2018).
- Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, Nikolaos Konstantinou: *Dataset Discovery in Data Lakes*. 2020 IEEE 36th International Conference on Data Engineering (ICDE 2020).
- Javier Flores, Sergi Nadal, Oscar Romero: *Towards Scalable Data Discovery*. 24th International Conference on Extending Database Technology (EDBT 2021).

Data Quality

Data is the main asset of a Data Science project. However, different data sets may attain different data quality levels. For example, data from a source generating too many errors (e.g., a faulty sensor, manually introduced data by humans with default values, etc.) might be problematic if not filtered and cleaned before being used. In databases, a well-known motto is that of Garbage In-Garbage Out (GIGO), which means that if you do not care about the quality of your data, then your models and analysis drawn from them will be indeed garbage too.

With the arrival of Big Data and the availability of a deluge of digital information, data quality has become even more crucial and more difficult to deal with.

In the past, we talked about data curators, data wranglers or the like, which were people manually taking care of the data quality or creating semi-automatic processes that helped to identify errors and process them. To help this people, different data quality perspectives were introduced, which provided hints of what to look at to assess data quality (e.g., accuracy, freshness, completeness, uniqueness, etc.). But now the focus is on automatic data quality processes, or at least, as automatic as possible.

A huge effort was recently invested on the area of supporting the data curator. Indeed, some of the companies raising more funds in the last year in Silicon Valley are related to this topic. But unfortunately, data quality has some artisan aspects, very ad-hoc to the project at hand, that make this problem huge and far away from being solved.

Nevertheless, checking the bright side of this history, there have been recent and interesting advances that are allowing to move forward in advanced data quality processes for Data Science. The most crucial of them all is the introduction of the concept of denial constraint (DC). A denial constraint is a formalism able to express most of the integrity constraints we are used to (e.g., key constraints, functional dependencies, etc.) but in a compact yet very expressive manner. Denial constraints are grounded in logics, which in addition allow to automatically infer interesting consequences (such as clashing DCs, subsumed DCs, etc.). Out of this formalism, a wealth of research appeared as well as very popular tools already in the market. In short, DCs

have allowed to automate most of the Data Quality lifecycle: elicit DCs from data sets, propose automatic reparation rules and apply them (always supervised by humans in what is called human-in-the-loop tools).

This topic can be reformulated in different manners, but essentially address the same problem. For example, "*how to assess the quality of a data source or how to repair dirty data"*.

Seminal bibliography:

- Redyuk, S., Kaoudi, Z., Markl, V., Schelter, S.: *Automating data quality validation for dynamic data ingestion*. 24th International Conference on Extending Database Technology (EDBT 2021).
- X. Chu, I. F. Ilyas, and P. Papotti. Discovering denial constraints. 39th International Conference on Very Large Databases (VLDB 2013).
- El Kindi Rezig*: Data Cleaning in the Era of Data Science: Challenges and Opportunities.* 11th Conference on Innovative Data Systems Research, (CIDR 2021).
- El Kindi Rezig, Lei Cao, Michael Stonebraker, Giovanni Simonini, Wenbo Tao, Samuel Madden, Mourad Ouzzani, Nan Tang, Ahmed K. Elmagarmid: *Data Civilizer 2.0: A Holistic Framework for Data Preparation and Analytics*. 45th International Conference on Very Large Databases (VLDB 2019). Note: Data Civilizer is the backbone of the popular tool *Trifacta (https://www.trifacta.com/)*.

Entity Resolution

Once data is integrated and crossed from different sources, we have the problem to identify if two instances from two different data sets (or even the same data set) might be the same. For example, suppose two sessions browsing an e-commerce platform. It might be interesting to identify all sessions of the same person even if this person is not logged in and, depending on the information logged, it might not be straightforward to identify such sessions. Entity resolution (aka entity matching) indeed aims at identifying different descriptions (e.g., two instances) that refer to the same real-world entity appearing either within or across data sources, when unique entity identifiers are not available.

The relevance of entity resolution for Data Science is huge. We require processes that are able to provide hints whether two different instances from two different data sets refer to the same real-world entity. Realize the impact of this when performing data analysis: a data model might be biased if a certain event appears many times as input or you may lose the big picture related to a real-world event if you fail to detect all data pieces related to it. For this reason, entity resolution nowadays transcends its relevance beyond the analysis and it is also essential for organizing data in Data Lakes into categories.

In the past, entity resolution was tackled as duplicate detection within a database. But with the arrival of Big Data and Data Science, current approaches were challenged and felt obsolete. Similar to the previous cases, the current wealth of data available forces us to look for automatable approaches reducing the impact of this problem in Data Science projects.

By definition, entity resolution has been tackled as a probabilistic problem. Further, given the inherent computational complexity (comparing all instances against all other instances is quadratic but in front of very large databases this cost is not attainable) many techniques trying to reduce the search space were introduced. The most popular, most probably, is blocking, which blocks similar instances (e.g., according to a function) and performs a deep comparison only among similar instances.

Besides blocking, the other usual assumption that entity resolution techniques make is the definition of a similarity score. This is basically the core of the approach and it is a function (or even a complex learned model) that tells us if two descriptions resemble each other. Usually, the output is a confidence score.

Of course, entity resolution might be even more difficult with the presence of dirty data and, in a way, finding duplicates is a cleaning task. For that reason, this is typically a process carried out after (or iteratively intertwined with) cleaning data and, at the same time, considered to be part of Data Quality. Nevertheless, this specific problem is huge enough as to be researched on its own.

Last, but not least, a usual question you may raise is the difference between entity resolution and data discovery. Intuitively, you can think of entity resolution as looking for duplicates, while data discovery aims at identifying *unionable* and *joinable* datasets.

All in all, entity resolution is "*the problem to identify, from one or several data sets, the instances referring to the same real-world entity*".

Seminal bibliography:

- G. Papadakis, J. Svirsky, A. Gal, and T. Palpanas. *Comparative analysis of approximate blocking techniques for entity resolution*. 42th International Conference on Very Large Databases (VLDB 2016).
- T. Papenbrock, A. Heise and F. Naumann, *Progressive Duplicate Detection.* IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 5 (2015).
- V. Rastogi, N. N. Dalvi, and M. N. Garofalakis. *Large-scale collective entity matching*. 37th International Conference on Very Large Databases (VLDB 2011).

Feature Selection

Big Data plays a significant role for building efficient models improving decision making, yet it comes with the problems of storage (and large processing times), challenging data understanding and visualization, poor generalization by models due to overfitting, and lengthy model building times. As the dimensionality of a dataset increases, the required number of samples needed to build a model that generalizes it properly grows more than exponentially. The state of having insufficient samples compared to the dimensionality of a dataset is referred to as sparseness and it generally increases as the dimensionality increases. Sparseness negatively affects the quality of a model's performance metrics, for example, its accuracy.

Besides the problem of sparseness, high-dimensional datasets are difficult to understand and visualize. Models built with such datasets are thus difficult to explain, especially in terms of the many features; the time required to build such models could also be very long. Associated with high dimensional data is also the need for relatively more storage space which signifies increased expenses whether on-site or in the cloud. These resultant challenges from the high dimensionality of datasets have been termed the curse of dimensionality. To fight the curse of dimensionality, effective techniques to reduce the dimensionality of datasets by selecting the most informative features of a dataset are necessary.

Many of such techniques have been long proposed and are called Feature Selection (FS) methods. Most FS methods were designed for small-medium sized datasets with polynomial theoretical runtime and so in the face of large-sized datasets, their effectiveness becomes

jeopardized and they fall into the curse of dimensionality. The aim of this topic is thus to study the state-of-the-art on "*feature selection methods, focusing especially on the ones that scale well over Big datasets; highlighting the current issues of feature selection in Big Data*".

Seminal bibliography:

- Kewei Cheng, Jundong Li, and Huan Liu. 2016. FeatureMiner: a tool for interactive feature selection. In CIKM. 2445–2448
- C. Reggiani, Y.-A. Le Borgne, and G. Bontempi, "Feature selection in high-dimensional dataset using mapreduce," in Benelux Conference on Artificial Intelligence, pp. 101–115, Springer, 2017.
- D. Peteiro-Barral, V. Bolon-Canedo, A. Alonso-Betanzos, B. Guijarro-Berdinas, and N. Sanchez-Maroono, "Scalability analysis of filter-based methods for feature selection," Advances in Smart Systems Research, vol. 2, no. 1, p. 21, 2012.

Implement your solution

Once you have read and learnt the basics on the topic you prefer, you must **implement** an extension on top of your project to include the studied topic. The objective is to practice how to incrementally add complex to the baseline we created while learning more about hot topics in Data Science.

First, identify the zone either in the data management or analysis backbone where your solution must be implemented. After reading the topic, you must decide in which zone you should include new scripting / data repositories to include it in the project. You must justify in the document your decision. Importantly, we want you to realise that the zones facilitate the maintenance (adding, removing or modifying modules to tackle new needs) of a data science project.

Once identified the zone, follow the same pattern as for the data management and analysis backbones: use notebooks in the prototyping phase to generate a solution. Once you are satisfied with it, you should move it to operations and generate the required operations artefacts and orchestrate them with the other modules of your project.

Constraint 11. Identify the zone where to add your advanced topic prototype. Justify your choice. Next, propose a new notebook (or set of notebooks) to add in this zone and discuss how they relate to the existing ones. Maintainability and separation of concerns must be guaranteed. To decide if one or many notebooks are needed, we advise you to follow the same principles as in the rest of the project; i.e., be sure to separate different concerns in different notebooks.

Constraint 12. Your operations layer must be able to execute the complete set of artefacts needed to cover the solution devised for your advanced topic. It is important you properly orchestrate your solution for the advanced topic with the other elements of the zone where you included it.

4. Deliverables

The outcome of the second part of the project is twofold:

- An explanatory document (max. 10 pages long),
- The project repository (it must be the same as in the first part, but now also including the data analysis backbone and the advanced topic)

The document must contain the following sections:

1. Your project supervisor should have access to your **development** (e.g., Drive) and **operations** (e.g., Github) **platforms** where you developed your project. Instructions how to access to these platforms must be in a mandatory section to be included in the document (first section of the document, in a separated page, without numeration) providing the link to these platforms. This section does not count for the maximum number of pages to be used in the document.
2. **Context**. If there is any change you want to include in the context delivered for the first part, do it here. Further, precisely and concisely state your analysis question.
3. **The data analysis backbone**. Using the figure from the slides, instantiate it (i.e., add on top) the tools you used for each element from the architecture.
4. **Advanced topic**. Identify the advanced topic chosen. Explain the solution you implemented and the different notebooks used in the prototyping phase.
5. **Operations**. Add an explanation of how you have organized your operations for the data analysis backbone and the artefacts needed to implement the advanced topic chosen. Specifically, pay special attention to justify the new code added to orchestrate the pieces added.

For items 3 and 4, it is important you discuss the pros and cons of your chosen solution. We will positively assess if you identify the limitations of your current approach (do not worry to state them, it is a baseline, simple solution, it will have plenty). Do not repeat here the information in the notebooks / code. The document should be an overall description to understand your development and operations platform and facilitate the supervisor to explore it.

5. Evaluation

The evaluation of this part of the project is follows the same criteria as for the first part:

Dynamicity. How easy is to add a new feature to be considered in the analytical backbone? And add a complete set of new features? How easy is to change the transformations executed between two zones?

Reusability. Code and data are not duplicated and they are well organized.

Openness. Your system could be easily extended and evolved with new / advanced aspects. This is evaluated via the solution provided for the advanced topic.

Single source of truth. Analysts can access a single source of truth from where to start the analysis.

Reproducibility. The executables in operations can be easily executed and are properly documented to do so. Similarly, for the notebooks in development.

Soundness. The choice of tools / solutions is adequate for each zone.

Completeness. All constraints in the statement are met.

Rigorous thinking. In the document, there is a detailed discussion about pros and cons of each solution and the students identify room for improvement for advanced solutions.

Important Note:

Nowadays there is a boost of tools that cover the whole data analysis backbone (including some governance). For example: MLOps, Weight & Biases, Hugging Face, Hopsworks among many others. For this project, we advise you to avoid tools that automate this part. For sure, they will be interesting in your future professional career, but for the sake of learning, it is better you develop them on your own during the project and understand pros and cons by yourself. If you plan to include some tool of this kind, please, discuss it first with the project advisor.