Big Data to Enable Global Disruption of the Grapevine-powered Industries

D7.3 - Experimental Report on Projected Datasets

DELIVERABLE NUMBER	D7.3
DELIVERABLE TITLE	Experimental report on Projected Datasets
RESPONSIBLE AUTHOR	Nikola Tulechki (SAI)

Co-funded by the Horizon 2020
Framework Programme of the European Union

GRANT AGREEMENT N.	780751
PROJECT ACRONYM	BigDataGrapes
PROJECT FULL NAME	Big Data to Enable Global Disruption of the Grapevine-powered industries
STARTING DATE (DUR.)	01/01/2018 (36 months)
ENDING DATE	31/12/2020
PROJECT WEBSITE	http://www.bigdatagrapes.eu/
COORDINATOR	Nikos Manouselis
ADDRESS	110 Pentelis Str., Marousi, GR15126, Greece
REPLY TO	nikosm@agkroknow.comttttt
PHONE	+30 210 6897 905
EU PROJECT OFFICER	Mrs. Annamaria Nagy
WORKPACKAGE N. | TITLE	WP7 | Cross-sector Rigorous Experimental Testing
WORKPACKAGE LEADER	CNR
DELIVERABLE N. | TITLE	D7.3 | Experimental report on Project Datasets
RESPONSIBLE AUTHOR	Nikola Tulechki
REPLY TO	nikola.tulechki@ontotext.com
DOCUMENT URL	http://www.bigdatagrapes.eu/
DATE OF DELIVERY (CONTRACTUAL)	30 March 2020 (M27), 31 Dec 2020 (M36, Final Version)
DATE OF DELIVERY (SUBMITTED)	31 July 2020 (M31), 29 Jan 2021 (M37, Final Version)
VERSION | STATUS	2.0 | FInal
NATURE	RE (REPORT)
DISSEMINATION LEVEL	PU (Public)
AUTHORS (PARTNER)	Giannis Stoitsis (Agroknow), Ioanna Polychronou (Agroknow), Mihalis Papakonstadinou (Agroknow), Panagiotis Rousis (Agroknow), Timotheos Lanitis (Agroknow), Pnagis Katsivelis (Agroknow), Nikola Tulechki (SAI), Salvatore Trani (CNR), Ida Mele (CNR)
CONTRIBUTORS	Nikola Rusinov (ONTOTEXT)
REVIEWER	Simone Parisi (ABACO)

VERSION	MODIFICATION(S)	DATE	AUTHOR(S)
0.1	Table of Contents	06/04/2020	Nikola Rusinov (ONTOTEXT)
0.2	Initial version	30/06/2020	Nikola Rusinov (ONTOTEXT
0.3	Input from partners	10/07/2020	Ioanna Polychronou (Agroknow), Panagis Katsivelis (Agroknow) Salvatore Trani (CNR)
0.5	Internal review	15/07/2020	Salvatore Trani (CNR), Giannis Stoitsis (Agroknow)
0.6	Table of Contents, revised version	20/7/2020	Ioanna Polychronou (Agroknow), Mihalis Papakonstadinou (Agroknow), Panagis Katsivelis (Agroknow)
0.7	Initial version	30/07/2020	Ioanna Polychronou (Agroknow), Mihalis Papakonstadinou (Agroknow), Giannis Stoitsis(Agroknow), Panagiotis Rousis (Agroknow), Timotheos Lanitis (Agroknow)
0.9	Internal review, final comments	3/8/2020	Ioanna Polychronou (Agroknow), Mihalis Papakonstadinou (Agroknow), Panagiotis Rousis (Agroknow), Timotheos Lanitis (Agroknow)
1.0	Final first version	5/8/2020	Giannis Stoitsis (Agroknow)
1.1	Revision	23/12/2020	Nikola Tulechki (SAI), Salvatore Trani (CNR), Mihalis Papakonstadinou (Agroknow), Giannis Stoitsis(Agroknow), Panagiotis Rousis (Agroknow), Timotheos Lanitis (Agroknow), Nikola Rusinov (SAI)
1.2	Internal Review	17/01/2021	Simone Parisi (Abaco)
2.0	Final version	18/01/2021	Nikola Rusinov (SAI)

PARTICIPANTS		CONTACT
Agroknow IKE (Agroknow, Greece)		Nikos Manouselis Email: nikosm@agroknow.com
SIRMA AI (SAI, Bulgaria)		Todor Primov Email: todor.primov@ontotext.com
ConsiglioNazionaleDelleRicherche (CNR, Italy)		Raffaele Perego Email: raffaele.perego@isti.cnr.it
KatholiekeUniversiteit Leuven (KULeuven, Belgium)		Katrien Verbert Email: katrien.verbert@cs.kuleuven.be
Geocledian GmbH (GEOCLEDIAN Germany)		Stefan Scherer Email: stefan.scherer@geocledian.com
Institut National de la Recherché Agronomique (INRA, France)		Pascal Neveu Email: pascal.neveu@inra.fr
Agricultural University of Athens (AUA, Greece)		Katerina Biniari Email: kbiniari@aua.gr
Abaco SpA (ABACO, Italy)		Simone Parisi Email: s.parisi@abacogroup.eu
SYMBEEOSIS EY ZHN S.A. (Symbeeosis, Greece)		Konstantinos Rodopoulos Email: rodopoulos-k@symbeeosis.com

ACRONYMS LIST

BD	BigDataGrapes
DSPS	Distributed Stream Processing Systems
IoT	Internet of Things
LDBC	Linked Data Benchmark Council
MIPS	Million instructions per second
MFLOPS	Million floating-point operations per second
MUDD	Multi-dimensional data generator
OWL	Ontology Web Language
RDF	Resource Description Framework
SPARQL	Symantec Protocol and RDF Query Language
SNB	Social Network Benchmark
SPB	Semantic Publishing Benchmark
SDPS	Streaming Data Processing Systems
TPC	Transaction Processing Performance Council

EXECUTIVE SUMMARY

The deliverable D7.3 "Experimental Report on Projected Datasets" consists of a report describing the outcomes of the experimentation performed on the projected datasets provided by each pilot, utilizing the Big Data Grapes stack.

This report starts by outlining the methodology and metrics we employ for our experimentation. Moreover, data generators are described emphasizing on State of the Art. We focus our experimentation on a pilot level, initially analysing the provided datasets and evaluating them against the Big Data Vs, as suggested and described in detail in D7.1 and described the data usage patterns.

For three of the pilots, we move on to identify and document the data flows each pilot adheres to throughout the Big Data Grapes stack, highlighting the frameworks and components involved in the process. Moreover, applying data generation techniques create projected datasets per dataset. We then experiment on each of the identified data flow steps by describing 3 usage scenarios for each. During the execution of each scenario for each step, we monitor the chosen performance metrics, showing the respective diagrams and analysing the outcomes.

For the two remaining pilots we perform large scale estimates of the data growth patterns.

This deliverable also presents an end to end report for each of the pilot's data flows, identifying bottlenecks and making suggestions to improve the performance where needed.

TABLE OF CONTENTS

LIST OF FIGURES

1.INTRODUCTION

The Big Data Grapes platform aims at using big data tools and technologies to help the grapevine industry. Within the context of this project a vast amount of data has been incorporated and used to stress-test the platform, as described in D7.2.

In this report we perform a projection over the collected data for the next 5 and 10 years to ensure that the design and implementation of the data platform will show no decrease in performance.

This report is structured as follows:

Initially we describe the methodology we will use and pose the probable evolution scenarios.

In chapter 4 we perform the end-to-end experimentation described in D7.2 for the worst case scenario for three of the pilots. And we describe the generation methods used on the projected datasets for the next years that follow the statistical distribution that was the outcome of the previous step.

In chapter 5 we report on the projections for each dataset in terms of number of records for the two remaining pilots and we perform a statistical analysis on the provided datasets.

We conclude this report with our main findings and experimentation outcomes.

2.METHODOLOGY

In this section we describe the methodology we will use within this report to generate the projected datasets and perform the proposed experimentation on top.

Data generation is a crucial and challenging task which allows testing novel research approaches, new optimization techniques, and different database configurations. Very often, the original datasets cannot be used for the testing due to data sparsity or privacy issues. For these reasons, a lot of effort has been devoted to the generation of synthetic datasets that allow repeatable tests when data is too sparse and to protect user privacy since synthetic data can be safely shared with third parties for testing their approaches (Anderson et al., 2014). The challenges of data generation lie in preserving the properties of the original datasets (realistic data) as well as dealing with any type of data (versatility) and being fast (efficiency).

For small datasets and simple settings, the generation of synthetic data can be done with ad-hoc scripts that are tailored for each use case. On the other hand, for large datasets and complex scenarios (e.g., testing a database consisting of many tables, with several use cases and dependencies), automatic approaches for synthetic-data generation are needed. For this deliverable, simple script-based solutions are satisfactory as the size and complexity of the BDG data is moderate. Anyway, if data scales up and complexity becomes an issue, there are several state-of-the-art approaches for generating synthetic data in a fully automatic way (Rabl et al., 2010 & 2015).

Synthetic data generators use meta-information given by the original data to generate synthetic (but realistic) datasets. Moreover, these generators are versatile and able to adapt to any type of data as well as easy to use and efficient.

As described in D7.1 and D7.2, we focused our experimentation around the 4Vs of Big Data: volume, velocity, variety and veracity. Based on the characteristics of each dataset, we choose to employ techniques and methods described in the bibliography (Deliverable D7.1, see BDGS and BigDataBench) to ensure the presence of every V for each of the datasets provided by the pilots. We focus our experimentation on the performance of the BDG stack by tracking several system indicators as described in section 2.2. Finally, we employ MetricBeat, i.e., the proposed tool for monitoring the scalability of BD platform reviewed in D7.1 and used in D7.2, over our Elastic stack to monitor and report our chosen metrics.

Since all of the BDG stack has been deployed in a microservice architecture, our step by step and end to end experimentation is done over the API endpoints provided by the platform. To make the experimentation easier we developed a python script that simulate these endpoint calls.

To showcase the potential of the deployed stack in terms of scalability we will perform the experimentation using three usage scenarios for each step of the process by gradually increasing the requests made towards the platform in the next version of the deliverable.

In this section we describe the methodology we use for the experimentation over the provided datasets, along with the steps we perform and the respective metrics we use for each of them.

2.1. STATE OF THE ART

Synthetic Generators for Big Data. Parallel Data Generation Framework (PDGF) is a famous suite for synthetic data generation (Rabl et al., 2010). Due to its versatility, it is often used as a standard core of the data generators employed in big data projects, for example, it was used to implement the data generator for BigBench (Ghazal et al., 2013). Moreover, PDGF is fast and efficient as it can run with high speedup in a multi-core and multi-node environment. It is also general and versatile as it writes data in different formats (e.g., CSV, JSON, XML, and SQL) to files, database systems, and big data storage systems (e.g., HDFS). PDGF's generation strategy is based on the determinism of the generators for pseudo-random numbers (i.e., the sequences of random numbers are repeatable). So, PDGF uses XORSHIFT random number generators that behave like hash functions. More in detail, it is based on a seed-generation hierarchy where one seed is chosen for each table, then this seed is used to create one seed per column, which in turn is used to create one seed per field. Lastly, the field seed is used to generate a random number for the generation of the value to use in the synthetic table. Value generators can be simple numerical generators or based on dictionaries to map the random number to any type of value. The users have to simply specify the data model and formatting instructions, making the framework easy to use. Besides, PDFG is very efficient as the seeds can be precomputed and cached.

Rabl et al. (2015) presented a completely automatic approach for data synthetization from existing data sources. Their system, called DBSynth, is an extension of PDGF able to generate large amounts of synthetic data that realistically reflects the complexity of the original data. Compared to PDGF, DBSynth makes the configuration automatic as it enables the extraction of data model information from existing databases.

Other tools for scaling up existing datasets are UpSizeR and Myriad. The former generates a graph of the original schema information and correlation information with the purpose of generating synthetic data accordingly to the patterns observed in the real dataset (Tay et al., 2013). Myriad has a configuration generation tool, called Oligos, which can analyze the schema and statistical information of DB2 databases (Alexandrov et al., 2012).

Synthetic Generators for Realistic Data. Keeping the synthetic data as much realistic as possible is a key challenge in synthetic data generation. This is due to the fact that synthetic datasets might miss to preserve the hidden patterns of the original dataset, so the analyses on synthetic data can compromise the benchmarking results.

Researchers and data scientists have focused on preserving the properties of the original datasets. Traditional techniques are based on domain sampling within a field or on preserving the cardinality relationships among different fields. Despite these techniques help by adding degrees of realism to synthetic data, they often fail to preserve the hidden complex patterns of real-world datasets (Hoag & Thompson, 2007). Indeed, relying on a mapping to transform a real dataset into a synthetic one does not give any guarantee that the original patterns are preserved. Moreover, there is a risk that the applied transformations can be discovered by malicious users and the original dataset can be recovered, putting the user privacy at stake. Also, model-driven generators can only preserve some of the patterns they are able to encode.

Although synthetic datasets are better suited for benchmarking, they have to reflect the characteristics of the original datasets. For this reason, real data is often analyzed for modeling benchmarking datasets. This step is typically manual, but it can be automated. Data mining techniques can be employed to discover patterns in real datasets that must be preserved in the synthetic data. Eno and Thompson (2008) proposed an approach based on decision trees for generating synthetic datasets of any size which show the same patterns of the original data. The drawback of this work is that it takes into account only the one type of pattern (i.e., the decision-tree pattern).

RSGen, presented by Shen and Antova (2013), relies on metadata stored in databases to get information about data distribution and structure. It ensures realistic data since it generates similar datasets by using histograms of the original data, but it is limited to only numerical data. A more versatile tool is DBSynth (Rabl et al., 2015). It leverages built-in dictionaries for mapping numbers to values of any type.

Generation Languages for Synthetic Data. Synthetic Data Definition Language (SDDL) and Predictive Model Markup Language (PMML) are two well-known languages in the field of synthetic-data generation.

SDDL is an XML-format language for describing the synthetic data generated by a software. The advantage of SDDL is that for each element of the original dataset, a pool of possible values can be specified, each one with a weight (probability) representing how much the value is likely to appear in the data.

PMML is an open standard language where each file consists of a data dictionary and one or more data-mining models. The former contains information about field types and data ranges and is independent of any data-mining model which represents information about the distribution of values in a field. A single PMML file may contain multiple models (e.g., decision trees, association rules, cluster models, regressions).

2.2. Steps and metrics

We choose to perform the experimentation on a pilot level. We identified a set of steps we follow for each of them. First, we identify and abstractly describe the provided datasets for each pilot. Then we move on with evaluating them against the Vs of Big Data, generating where applicable synthetic data series to cover any of the 4 Vs that are not covered by the provided datasets. We then identify the data flow each dataset will follow in the BDG stack, denoting the steps of this flow. Finally, using a python script that will simulate bursts of this data flows throughout the BDG stack, we will monitor and report our chosen metrics for this benchmark in real time.

It should be noted that all of the experimentation is done using the actual provided data by each pilot and involves the already integrated and working components of the BDG stack.

As mentioned above, we will perform step by step and end to end experimentation for each pilot. In order for us to track and report the performance of the stack we chose to monitor the following indicators:

- Completion time, both on a step by step level, as well as on the whole end to end data flow,
- CPU (central processing unit) usage, we will track the CPU usage by each of the components as they are triggered by the flow,
- Memory usage of each of the components and technologies,
- Network usage in terms of bytes, we employ this metric, since the whole stack is based on a microservice architecture.

3.EVOLUTION SCENARIOS AND DATA TRENDS

Agriculture and food are major contributors to the global economy. Agriculture and food are major contributors to the global economy, underpinning livelihoods and economic growth in the developed and developing world alike. Overall, the agriculture and food industries account for 6% of GDP in the EU, comprising 15 million businesses and 46 million jobs[1]. Meanwhile, agriculture accounts for 65% of Africa's workforce and 32% of the continent's GDP[2]. In some of Africa's poorest countries, including Chad and Sierra Leone, it accounts for more than 50% of GDP.

Agriculture and the global food system are struggling under the combined pressures of a growing population, climate uncertainty and volatile market forces.

Populations are growing, climates are changing, and markets are often volatile. As the world's population grows to around 10 billion by 2050[3], the global agriculture system is under pressure to provide sufficient nutritious food to meet the demand. Much progress has been made in reducing hunger and poverty and improving food security and nutrition. Gains in productivity and technological advances have contributed to more efficient resource use and improved food safety. But major concerns persist. Based on "The future of food and agriculture, Trends and challenges" (FAO[4]), some 795 million people still suffer from hunger, and more than two billion from micronutrient deficiencies or forms of over nourishment.

Based on FAO "Agricultural productivity and innovation[5]" (5) to meet demand, agriculture in 2050 will need to produce almost 50 percent more food, feed and biofuel than it did in 2012. This FAO estimate takes into account recent United Nations (UN) projections indicating that the world's population would reach 9.73 billion in 2050. In sub-Saharan Africa and South Asia, agricultural output would need to more than double by 2050 to meet increased demand, while in the rest of the world the projected increase would be about one-third above current levels. Accounting only for the revised population projections, global agricultural demand is projected to increase by more than 63 percent between 2005/07 and 2050. Since production expanded by 15 percent between 2005/07 and 2012, the projected increase in agricultural demand from 2013 to 2050 would amount to approximately 49 percent.

For agriculture to respond to future challenges, innovation will not only need to improve the efficiency with which inputs are turned into outputs, but also conserve scarce natural resources and reduce waste (OECD, 2011; Troell et al., 2014). Worldwide, conservation agriculture has been adopted on some 117 million ha, or about 8 percent of total world cropland. Demand for food and other agricultural products is projected to increase by 50 percent between 2012 and 2050.

Alongside these challenges are huge opportunities: a global data infrastructure is emerging, and, with innovative business models, it is time to invest in open data-driven solutions in agriculture and nutrition. One of the major challenges is to make agriculture and food systems sustainable is the increasing of the open data. It is a common belief that investing in data-driven agriculture is expected to increase agricultural production and productivity, help adapt to or mitigate the effects of climate change, bring about more economic and efficient use of natural resources, reduce risk and improve resilience in farming, and make agri-food market chains much more efficient. As a result, agricultural sector can start making more informed decisions, making the sector run more smoothly and contributing more to these challenges. Therefore, the increase in data must be large enough to directly meet the needs that arise. In many cases, the data increases in proportion to the growing trend of needs.

[1] European Commission (2013), The common agricultural policy and agriculture in Europe: Fact sheet http://europa.eu/rapid/press-release_MEMO-13-631_en.htm, accessed 27/04/15

[2] World Bank (2013), World Bank Agriculture and Africa: Fact sheet, http://web.worldbank.org/WBSITE/EXTERNAL/COUNTRIES/AFRICAEXT/0,,contentMDK:21935583~pagePK:146736~piPK:146830~theSitePK:258644,00.html, accessed 18/05/

[3] https://www.unfpa.org/world-population-trends

[4] http://www.fao.org/3/a-i6583e.pdf

[5] http://www.fao.org/3/a-i6583e.pdf(page 46)

Specifically for the farming sector the set up of the IoT installations is increasing significantly as presented in figure 1 that show the trend of the device shipment for the last 5 years.

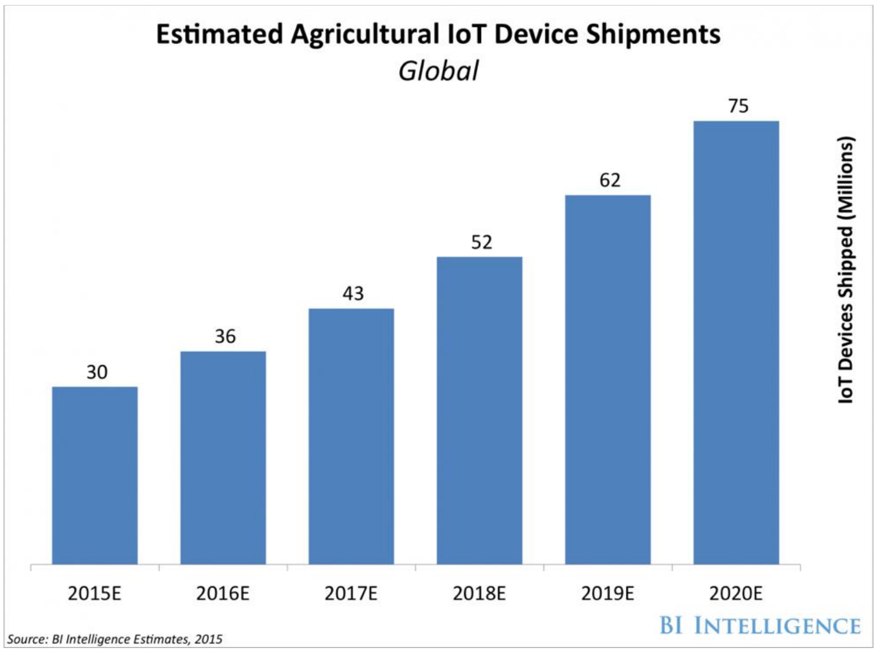


Figure 1: Figure 1: IoT device shipment trend for the last 5 years

The trend of the data generated by a farm per day will be increased by 400% in 2030 as presented in figure 2.

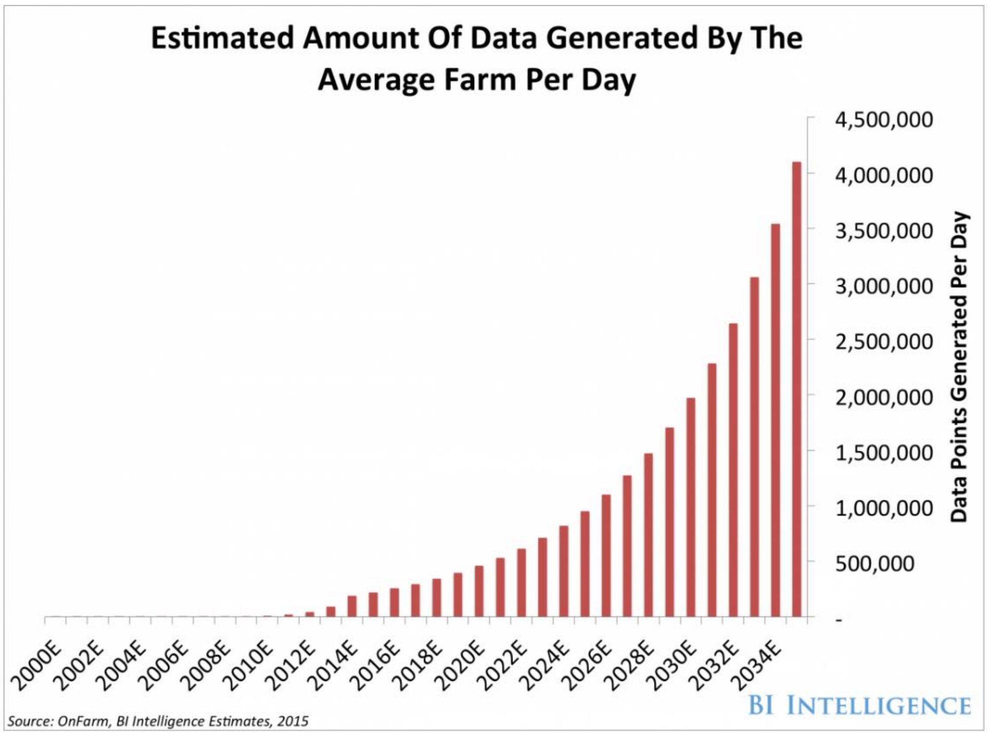


Figure 2: Estimated amount of data generated by farm per day

Using this high level trends for the growth of data in the agrifood sector in combination with the growth rate for specific data types used in each pilot, we will define the data specific growth rate for each pilot in the project.

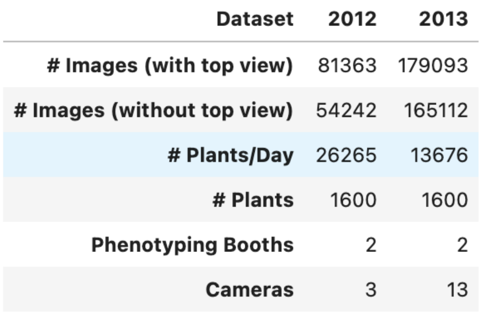
4.RIGOROUS TESTING EXPERIMENTS

4.1. WINE MAKING PILOT

This pilot deals with research in the fields of viticulture and phenology with a focus on the quality of the final product and its association with indicators collected from the field and the laboratory. The provided datasets include genetic data, weather and sensor data as well as laboratory experiments and field management activities. In particular, for the purpose of this experimentation, we focused on the leaf counting task, aimed at developing a machine-learned pipeline for counting leaves from side-view grapevine images taken into the imaging cabin of the PhenoArch platform managed by INRA. This specific application has been chosen because the task is particularly laborious and time-consuming, given each year there are thousands of plants to measure. In the following sections will analyze the data generation approach, the data generation approach for the leaf counting dataset and the result of the data projection. Finally, we will conduct an experimentation on the projected datasets collecting the metrics and adopting the methodology described in the previous section.

4.1.1. Data Usage Patterns

This pilot is characterized by two datasets concerning plant images collected by the PhenoArch phenotyping platform in two consecutive years, one in 2012 and another in 2013. For each of the two experimentations the platform, which consists of two phenotyping booths, tooks pictures of plants every two/three following days for a period of about 40 consecutive days. For each plant and day of imaging, in the 2012 dataset 3 cameras per phenotyping booth were used, while in the 2013 dataset this number raised to 13. Cameras are always fixed during one experiment, adopting the same position and settings, and photographing photos to each phenotyping booths. One camera takes images from the top, while the other takes side images. The plant rotates using a brushless motor (to mimic a turning table), so each side image is taken always in the same directions for each plant. On a regular basis, plants are analyzed by experts that annotate the number of leaves of each plant. The setup is described in detail in D.4.3, while in the following table it is summarized the characteristics of the two datasets. There were 26.265 annotations in the 2012 dataset, and 13.676 in 2013. Each annotation corresponds to a single plant analyzed in a specific day, and for that annotation several images were taken (3 for the 2012 dataset and 13 for the 2013 one).



Dataset	2012	2013
# Images (with top view)	81363	179093
# Images (without top view)	54242	165112
# Plants/Day	26265	13676
# Plants	1600	1600
Phenotyping Booths	2	2
Cameras	3	13

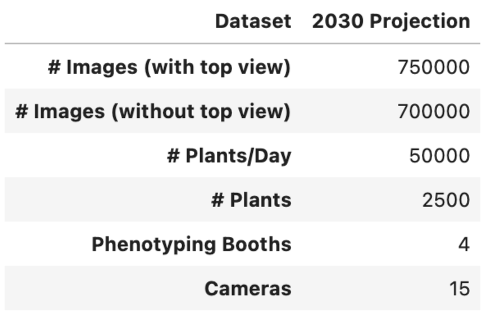
All of these images were accessible through the filesystem, thus they need neither to be injested into the platform nor to be processed or exposed by an ad hoc API. The growth rate of the data for the last 10 years will be used to generate the projected datasets. The experimentation on the projected datasets will be performed similarly to how it has been done on real data in D7.2. We will use the same methodology here in order to test the system over 10 years of projected data.

4.1.2. Data Generation

In this section we will describe how we generate 10 years of projected images for the leaf counting task. In order to have a realistic forecast of the growth rate for this pilot, we started from the following observations, based on the investments made by the INRA partner in the last year on expanding/developing the PhenoArch phenotyping platform:

- the number of phenotyping booths raised from 2 in 2012 to 13 to 3 in 2019
- the number of cameras increased from 3 in 2012 to 13 in 2013 and 15 in 2019
- the number of plants raised from 1600 in 2012/2013 to 2000 in 2019.

Based on these observations, and assuming to observe similar growing trends in the next decade, we can forecast the number of plants will increase from 2000 to 2500 in 2030 and the number of phenotyping booths will increase as well from 3 to 4. The number of cameras instead will remain unchanged, because they are already taking 14 side-view photos of every plant on each phenotyping booth, which is sufficient for the task. Also the time-interval between two consecutive photographing of the same plant will remain unchanged, and set to 2 days (setting used in the 2012 experimentation), and the duration of an experimentation, keeping fixed to 40 days (each plant will be photographed 20 times in the time interval, every two days). The number of annotations (single plant analyzed in a specific day) thus raises to 50.000. These forecasts result in a growth of the number of plant images by more than 400% . In the following table we summarize the characteristics of the projected dataset according to the aforementioned forecasts.



Dataset	2030 Projection
# Images (with top view)	750000
# Images (without top view)	700000
# Plants/Day	50000
# Plants	2500
Phenotyping Booths	4
Cameras	15

In order to generate the projected plant images we started from the real images of the 2012 and 2013 datasets and applied to them a data augmentation technique. This technique is standardly used in machine learning to generate artificial copies of already existing data by slightly modifying the original images by means of several geometric transformations, flipping, color modification, scaling, cropping, rotation, translation, noise injection. By adopting such a synthetic data generation approach we can benefit also for preserving the ground truth leaf counting label of the original images, thus allowing the usage of such artificial dataset for improving the performance of the machine learning model employed for this pilot, making it more robust. Previous research (Simard et al., 2003) to systematically understand the benefits of data augmentation techniques indeed demonstrated that data augmentation can act as a regularizer in preventing overfitting in neural networks and improve performance in imbalanced class problems. All in all, 700.000 plant images were generated by using this technique (plant images taken from top were not generated given they are not used by the machine learning model for the leaf counting task).

4.1.3. Experimentation

In this section we will perform the rigorous testing experimentation for each of the identified steps, using the projected dataset.

4.1.4. Prediction of Leaves

In this section we perform the rigorous testing experimentation for the leaf counting task by using the synthetic data generated according to the strategy defined above. The plant images synthetically generated are stored on the file system of the server where the model is deployed and the API are served. For us to track and report the performance of the stack we chose to monitor the following indicators:

- Completion time, on the whole dataset
- CPU (central processing unit) and GPU (graphical processing unit) usage, we will track the computational usage as it is triggered by the processing
- Memory usage of each of the components and technologies (main and GPU memories)

All the steps described are executed on one of our servers that has the followings specs:

- RAM: 192GB DDR3
- CPU: two Intel(R) Xeon CPU E5-2630 v3 @ 2.40GHz (dual CPU, 8 cores with hyper-threading, 32 total concurrent threads)
- GPU: Nvidia TITAN Xp GPU (12 GB of GDDR5 memory, 3840 Nvidia CUDA cores)

For the purpose of this experimentation we adopted the same experimental setting defined in D7.2, that we can summarize as it follows: we feed the prediction API of the machine learning component with every image of the synthetic dataset, collecting in background the several metric indicators.

4.1.5. End to End Report

In conclusion to our experimentation for the Wine Making pilot and the projected dataset we ran the experiments on the full synthetic dataset.

The completion time for concluding the inference of the 700.000 plant images was 16.132 seconds (i.e., about 22ms per image, in line with the performance of the model as evaluated on real data in D7.2). The CPU resulted to be completely saturated, i.e., used at 100% of its computational power, while the GPU to be under-used (i.e., we are not fully exploiting the computational power provided by the graphic unit**)**. All these findings confirm the results in D7.2, highlighting that the inference time of the model is only marginally affecting the efficiency of the component, while the preprocessing steps, despite fully exploiting the parallelism of the machine, represents a bottleneck in the current architecture.

These results show that the proposed approach scales well, in terms of average inference time per image, even when collecting a decade of data and for our purposes validates the usage of a complex solution based on a Deep Neural Network model within the chosen technological stack.

4.2. NATURAL COSMETICS PILOT

The natural cosmetics pilot focuses on the prediction of the biological efficacy of pharmaceutical plants. Currently this pilot has provided a dataset covering the lab experiments performed on the plants which is linked with satellite image processing of the respective fields in order for the correlation of lab tests and satellite images to be made possible. In the following sections will analyze the data patterns for each dataset based on "Dataset Analysis & Evaluation" from deliverable D7.2. We will describe the data generation approach and the result of the data projection. Finally, we will identify the data flows of this specific use case inside the stack and experiment on each of them, using the metrics and the methodology described in the previous section.

4.2.1. Data Usage Patterns

As mentioned in the introductory section, this pilot has provided 2 datasets. One is the lab tests performed on pharmaceutical plants and the other is the geographical information and related metadata of the fields the tests were performed against. Using the latter another dataset is also employed, that of the satellite image processing for these specific fields. In the next version of the deliverable, we will further describe the data patterns of these datasets in order to find the best data generation approach.

Besides for the related pilot, this dataset and use case were also used to demonstrate the distributed inference mechanisms over Big Data. The particularities of the data, notably one relatively small and static datasets with very few laboratory measurements and one highly dynamic dataset with hourly weather measurements made this use case ideal for experimenting and demonstrating a distributed data storage infrastructure, where a native triplestore (GraphDB) is used for the static data, a document store (Mongo DB) for the dynamic dataset with weather predictions. Query-time federation used to dynamically join the two. The setup and results are described in D.4.2. We will also use the same pipeline here in order to test the system over 10 years of projected data, and notably evaluate the behaviour of distributed components and query-time federation.

4.2.2. Data Generation

We generate 10 years of projected weather data. The initial dataset in the Natural Cosmetics Pilot consists of data from 14 weather stations, each collecting 7 measurements on a 10 minute interval. As only the summer months are relevant, the measurements cover only 4 months from May to July.
The following table shows the first ten measurements from the 2019 dataset of the Argos weather station.

date	time	temp_out	out_hum	bar	rain	rain_rate	wind_speed	wind_dir	hi_speed
01/05/19	0:00	17.3	46	1012.3	0	0	1.6	WNW	6.4
01/05/19	0:10	17.2	49	1012.4	0	0	1.6	SW	6.4
01/05/19	0:20	16.9	52	1012.3	0	0	1.6	SW	6.4
01/05/19	0:30	16.2	54	1012.3	0	0	0	SW	4.8
01/05/19	0:40	15.9	55	1012.3	0	0	1.6	SW	4.8
01/05/19	0:50	15.8	56	1012.3	0	0	0	---	0
01/05/19	1:00	15.6	58	1012.3	0	0	0	---	0

01/05/19	1:10	15.2	60	1012.3	0	0	0	SW	3.2
01/05/19	1:20	14.8	60	1012.1	0	0	3.2	SW	8

In order to generate the project data we programmatically reproduced the overall weather patterns of the current year for a period of a total of 12 years, thus simulating a complete weather dataset until the year 2032, covering the 14 meteorological stations. All in all, 2239029 sets of observations were produced for the whole period.

4.2.3. Experimentation

In this section we will perform the rigorous testing experimentation for each of the identified steps, using the projected dataset.

4.2.4. Dataset Upload and ingestion of Enriched Data

The generated dataset is converted to JsonLD, totalling 2.7GB of data and loaded into MongoDB using the built in tools. The upload results in a 369M hard disk footprint of the MongoDB collection. Note that one of the reasons we chose this distributed approach are the excellent built in compression mechanisms in this software.

4.2.5. Distributed inference and correlation of data

The main part of the experiment is to reproduce the distributed inference step over the simulated data, because this is the potential bottleneck of the approach. The setup consists of dynamically producing and aggregated weather dataset for each of the relevant estates. While kept in the MongoDB are the (very) granular measurements at 10 minutes increments, we will produce from the dataset aggregate on a daily level, with the average of the measurements as value for the given variable and day.

The rationale behind this approach is fully explained in D4.2, but let's summarize it here. Our aim is to ensure an eventual consistency guarantees between the two datastores. MongoDB serves as the source of truth and a master of the data for meteorological measurements. GraphDB stores the derived version of that data aggregated at the desired level. By periodically running the inference query will ensure that MongoDB and GraphDB remain in a consistent state. While introducing some lag between the data stored in Mongo and that within GraphDB, this approach allows us to have a much lighter triplestore footprint For analytical purposes it should be affordable. When the meteorological data accumulates to a considerable size doing the input asynchronously would be preferable. Synchronous inserts into the system would pose performance issues and a possibility for data loss.
Our aim here is to be able to reproduce the results form 4.2 on the much larger synthetic dataset

4.2.6. End to End Result

In conclusion to our experimentation for the Natural Cosmetics pilot and the projected data flows its dataset we ran the inference query (see D4.2 8.5) and measured execution time and inferred statements. We ran the experiments on a regular production server with an Intel I7 CPU, 48 Cores and 500G of RAM.

The inference query produced 87822 statements running for 37 minutes.

These results show that this approach scales well within the desired parameters even when collecting a decade of data and for our purposes validates the distributed inference setup within the chosen technological stack.

4.3. FOOD PROTECTION PILOT

Working with real industry scenarios we define which are the main parameters that need to be predicted. Based on these findings we test several machine learning and deep learning algorithms to predict parameters like the chemical risk, pricing and fraud. The goal is to combine different datasets to achieve the optimum performance for the prediction of the risk in raw materials (e.g. grapes) and finished products (wine). A large number of different data sources and data types has been used. We used textual information that includes mainly announcements about food recalls and border rejections (FDA, RASAFF, FSSA, AUSTRALIAN FOOD SAFETY AUTHORITY). We also use numeric data that is lab test and prices. The main source of our datasets is the open data published by the governments. Moreover, private data that includes ingredients that a company uses for food production, list of suppliers and internal lab test. Risk assessment module, price prediction dashboard and recall prediction have developed.

4.3.1. Introduction

One of the most challenging tasks for companies that produce and process a vast amount of data is performance management. To manage, store and process this overflow of data, a technique called "data scaling" has become necessary for many organizations dealing with exploding datasets. A scalable data platform accommodates rapid changes in the growth of data, either in traffic or volume. For this reason, it is very important to run the right benchmark tests to see how well a data platform will scale with a significantly enhanced number of data. In this paper, we will use data from Agroknow's Big Data Platform to follow a set of tailor-made steps and then monitor the performance of our results.
There is data concerning food recalls and border rejections, data pertaining to a huge amount of raw ingredients and commodities, and data unpacking all the possible hazards behind a food safety incident. All that data is collected from many trusted sources in real time to provide a collection of organized data, all in one platform.

Nowadays, food safety and certification throughout all the supply chain have become very strict requirements, so they are continuously in activity to develop and promote more selective methods of monitoring and control. That has led to a major increase in public data from official sources of food incidents across many countries that focus on a broad range of products and ingredients.

All that data was collected, stored, processed, and analyzed so it was important to have a data platform that was prepared and tested for future growth in its data volume. To make sure that our platform can handle the increasing amount of data, a scaling test was executed. We set a map of steps as follows: we describe the methodology we used as well as the metrics and finally, we conclude this report by analyzing the outcomes of this process, the identification of bottlenecks and heavy-duty tasks.

It should be noted that all the experiments were performed using actual data from Agroknow's food safety Platform dataset.

4.3.2. Goal of the Scalability Assessment

The goal of this scalability assessment is to perform an experiment that will test the scalability of the Big Data Platform for a specific use case such as the Food Protection use case, studying the scalability from collection of food safety records to the delivery of analytics and risk predictions to the user. In this experiment we took into consideration the growth of data for the next 10 years.

4.3.3. Experimentation Steps

4.3.3.1. Generation of simulated data

The goal of this step is to create a synthetic dataset that will simulate the growth of the data in the next 10 years. Overall, the steps we followed from end-to-end can be depicted in the figure below.

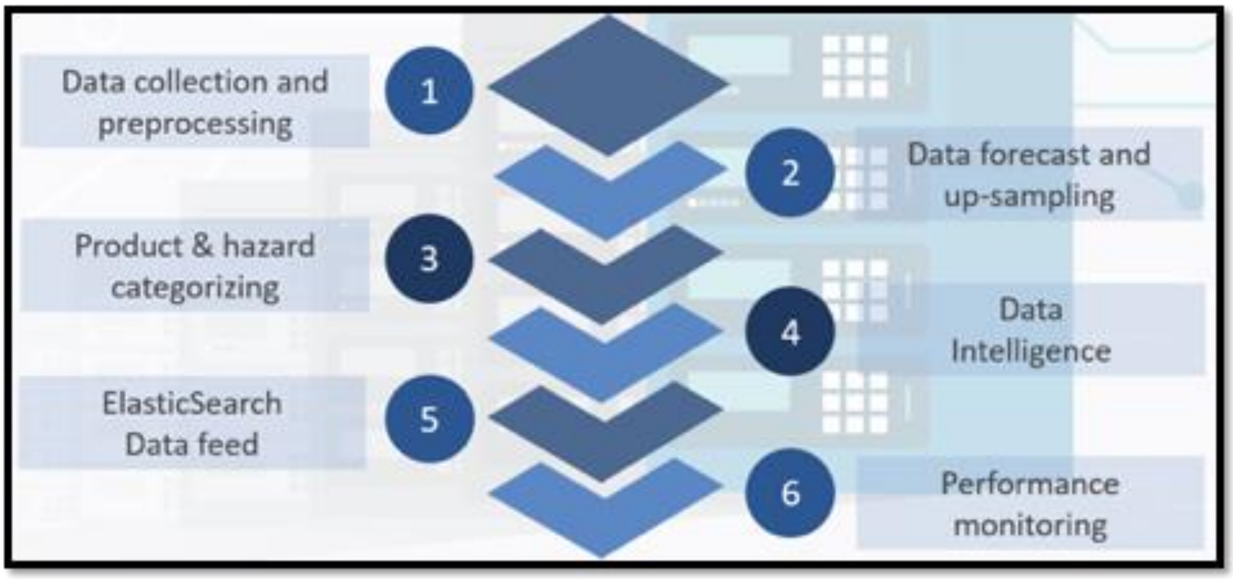

Data collection and preprocessing
1
2
Data forecast and up-sampling
Product & hazard categorizing
3
4
Data Intelligence
ElasticSearch Data feed
5
6
Performance monitoring

Figure 3: Experimentation steps

4.3.3.2. Data forecasting

In the first part of our experiment, we used a python script that collects a set of targeted fields from our dataset. We then used these fields to deduct the number of incidents reported throughout all the years, ranging from 1980 up to 2020. Incidents in our dataset include food recalls and border rejections. We stored the information required into a csv file that contains each date with its according number of incidents *(e.g., 2016-07, 3868)*. In total, we collected 488 raw observations. We plot our data so that we understand its distribution.

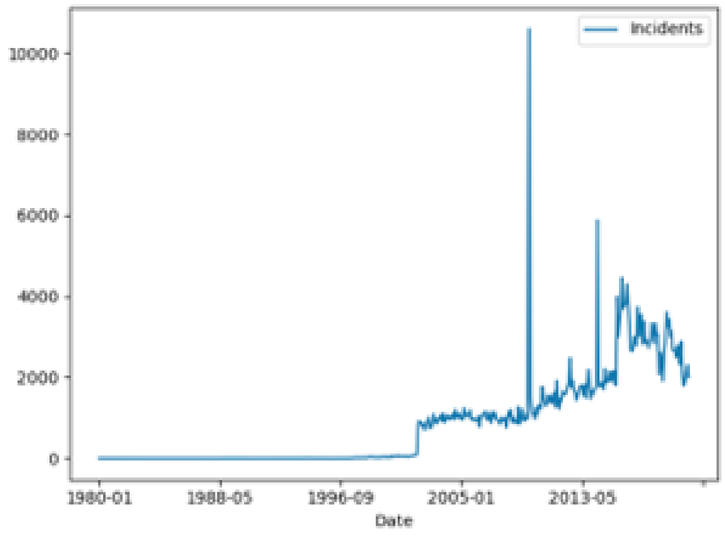


Figure 4: Our raw data distribution.

The next step was to preprocess and tidy up our data. We can see that during the first 20 years (approximately starting from 1980 up to the early 2000s) we have a slow flow of incidents reported. Also, we can notice two intense spikes during 2009 and 2014. For our first observation, we chose to completely remove the data from our incident's dataset setting our new start point in 2002, leaving us with 225 incidents. For our second observation, we chose to scale out the spikes by using the mean value of its two nearest neighbors. We preprocessed our data bringing it to the distribution as shown below.

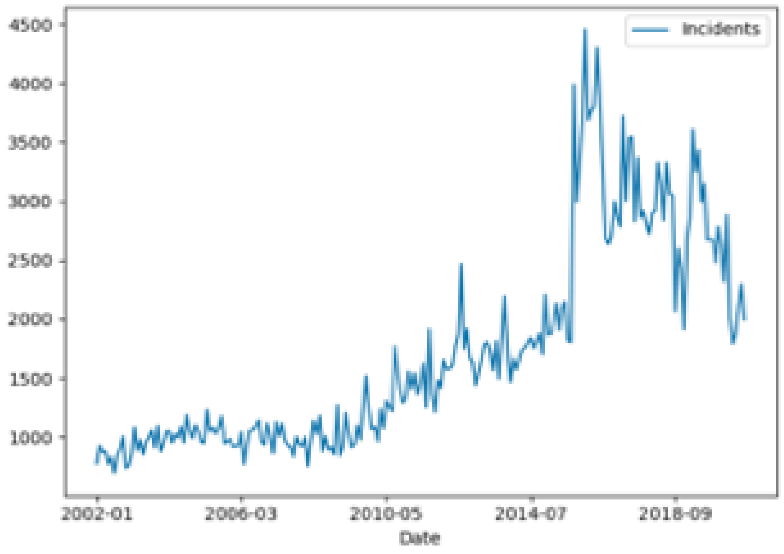


Figure 5: Preprocessed distribution.

Our focus was to forecast the number of data/incidents over the next 10 years. Again, we used a python script and more specifically, the forecasting procedure called "prophet" (*Taylor SJ, Letham B. 2017. Forecasting at scale*). We then carefully excluded and stored the actual values in a side variable in order to test our data using the RMSE metric after the forecast. The following figure depicts the forecast of our data/incidents with an RMSE of 390.

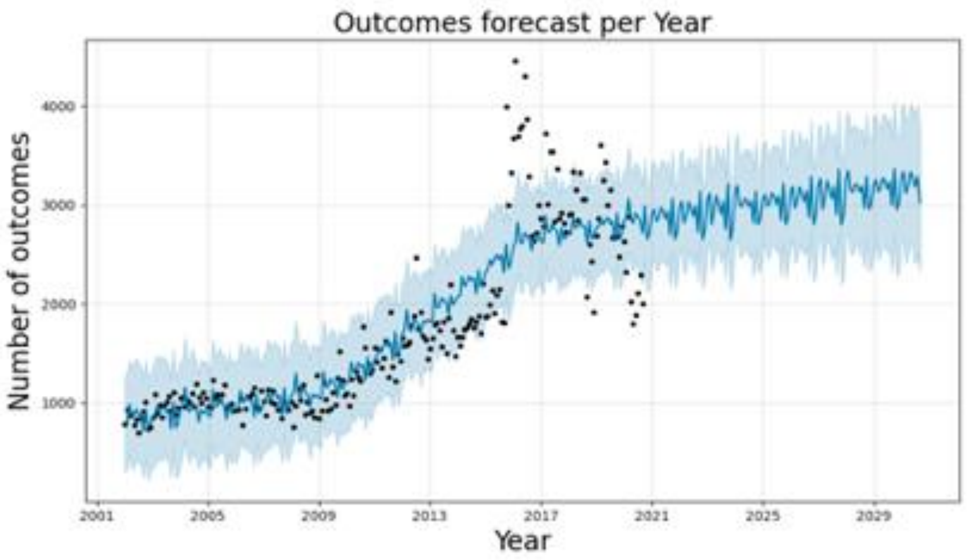


Figure 6: Forecast incidents.

In conclusion, our incidents number prior our forecasting (1980-2020) was 389.214. The incidents that were forecast by our model (2020-2030) were 386.648. In total, they added up to 775.862. We then took into consideration the RMSE error[1] to ensure that we had covered the worst-case scenario. We added up the error to each of our forecast value leading us to our final amount of 823.331 incidents by the end of 2030.

Upscaling the dataset

In the second step of our experiment, we enriched our data using a python script. We first exported all our current data/incidents, targeting a set of specific values (title, description, date of creation). After we have collected approximately 389.000 incidents, we up-sampled our data so that it reaches our future number of incidents. For the data upscaling process, we chose to randomly pick values from our existing registrations and append them at the end of our current dataset. Doing so, we finally reached the expected data/incidents number of 823.331 that was forecast in the previous step by prophet.

Characteristics of the original dataset are described below:

Dataset	**Incidents prior forecast**
Metadata/description	This dataset contains food safety incidents collected by Agroknow

Provider	Agroknow
Size in MB	130MB
Time period	1980-2020
Number of incidents	389.000

Characteristics of the synthetic dataset are described below:

Dataset	**Incidents after forecast**
Metadata/description	This dataset contains food safety incidents collected and forecast by Agroknow
Provider	Agroknow
Size in MB	220MB
Time period	1980-2030
Number of incidents	823.331

4.3.4. End to end scalability testing for the Food Protection pilot

4.3.4.1. Semantic enrichment of the data (Product and Hazard categorizing)

In this step, we attempt to categorize each data record based on its product type. To do so, we used our pre-trained machine learning model (*SGDClassifier, scikitlearn*) and provide it with the title and description of each incident. We created a new python script that will be executed in the background by one of our servers. We gave this task to one of our servers due to this being a time costly process, given the number of our data logs. After the process was over, we had our final dataset that is properly enriched. We did the exact same process once more to predict the hazard. Two new columns were created and added to our csv.

4.3.4.2. Data intelligence

Next, we attempted to further enrich our csv so that it reaches its final form. Our platform can perform risk estimation based on our current and predicted incidents. In more detail, we called one of our endpoints that performs incident prediction based on a custom-made strategy. To perform this process, a big number of data, up to 7 years ago, is used. It provided us with an incident prediction report for the next 12 months. That data was used by our second endpoint that performs the risk estimation process based on a custom formula that uses several factors. Our final column is added to our csv as well, forming our final dataset.

4.3.4.3. Ingesting data to Elastic Search

This step consists of two parts. First, we converted our csv to JSON because elastic search only supports JSON data format. We then used a python script to read all the data from our csv dataset and transform it to JSON. Next, we used another python script to send the JSON data file to our elastic search. We sent the data in patches for performance reasons.

4.3.5. Metrics and Specifications

In this section we perform the rigorous testing experimentation for each of the identified steps by using data that are stored in the Data platform. For us to track and report the performance of the stack we chose to monitor the following indicators:

- Ø Completion time, both on a step by step level, as well as on the whole end to end data flow,
- Ø CPU (central processing unit) usage, we will track the CPU usage by each of the components as they are triggered by the flow,
- Ø Memory usage of each of the components and technologies,
- Ø Network usage in terms of bytes

All the steps described are executed on one of our servers that has the followings specs:

- Ø RAM: 8GB
- Ø CPU: Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz
- Ø HDD: 200GB
- Ø OS: Ubuntu 16.04

To automate all the previously described steps, we used Apache airflow. The Figure below depicts all the steps in a Gantt diagram, emphasizing their time flow and duration.

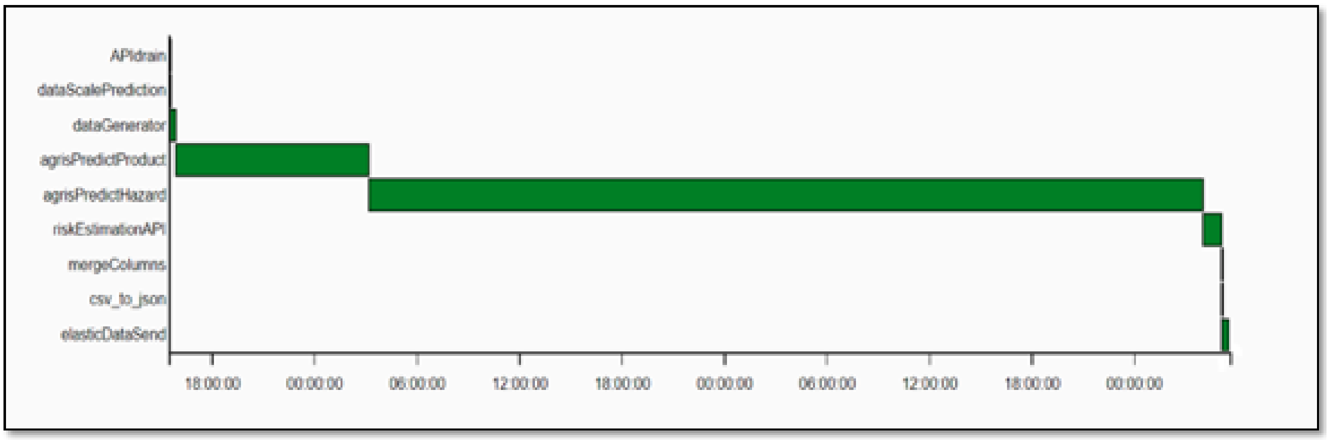


Figure 7: Gantt diagram exported from Apache airflow depicting execute times.

More specifically:

1. Draining our incident counts based on their date (*APIdrain*) has a duration of 6.443sec
2. Predicting the future data volume (*dataScalePrediction*) has a duration of 9.654sec
3. Downloading all our platform's current data and then upscaling it to reach the predicted future data volume (*dataGenerator*) has a duration of 21min 5.800sec
4. Predicting the product for each record (*PredictProduct*) has a duration of 11hours 19min 2.300sec
5. Predicting the hazard for each record (*PredictHazard*) has a duration of 48hours 49min 4sec
6. Estimating the risk value for each record (*riskEstimationAPI*) has a duration of 1hours 5min 27.130sec
7. Merging all the columns to form a final dataset (*mergeColumns*) has a duration of 4.666sec
8. Converting the final dataset to JSON (*csv_to_json*) has a duration of 25.373sec
9. Ingesting our elastic search with our data (*elasticDataSend*) has a duration of 15min

4.3.6. Performance Monitoring

4.3.6.1. Data platform scalability

In the next chapter, we will monitor the performance of the scalability so we will analyze the steps that affect the data platform's workflow. This means that we will monitor the Product prediction, hazard prediction, risk estimation and sending our data to elastic.

4.3.6.2. Product Automatic Classification

In this step we used Elastic stack's monitoring tool Kibana to inspect in more detail the server's resource drainage (*CPU usage, Memory Usage, Networking Traffic*) when the process of classifying the product category based on our input values is executed (*PredictProduct*).

The figures illustrated above present the completion time, CPU usage, network traffic and memory usage of the whole stack, throughout our experimentation for this specific step. We notice a rise in the CPU usage (*Figure* 5) when this task is executed that reaches roughly 25% usage and then drops to roughly 21% for the rest of its cycle. Memory usage is approximately 5GB for this step. The network traffic appears to have a balanced input/output flow.

Overall, we can conclude that the CPU usage is optimal for this step. The memory usage is slightly below the optimal level as more than 60% is used for this task. The network traffic can be considered minor and balanced.

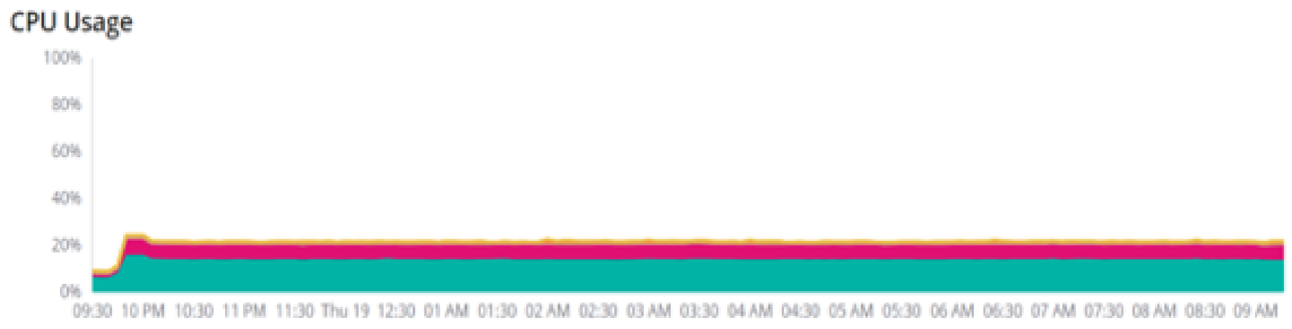


Figure 8: Product prediction CPU usage.

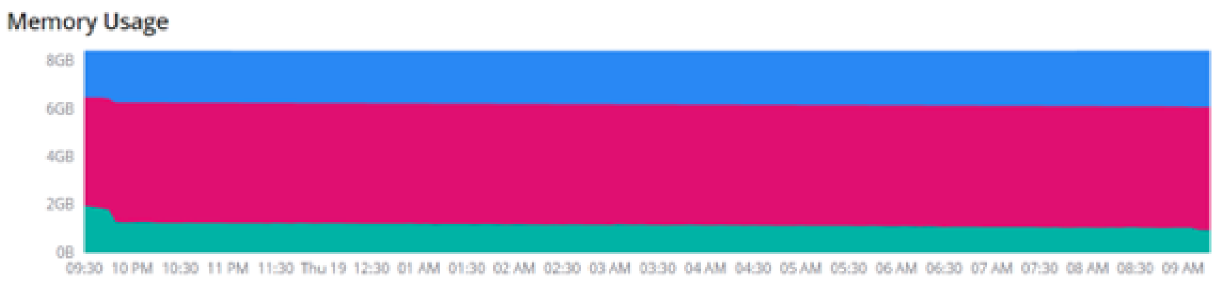


Figure 9: Product prediction memory usage.

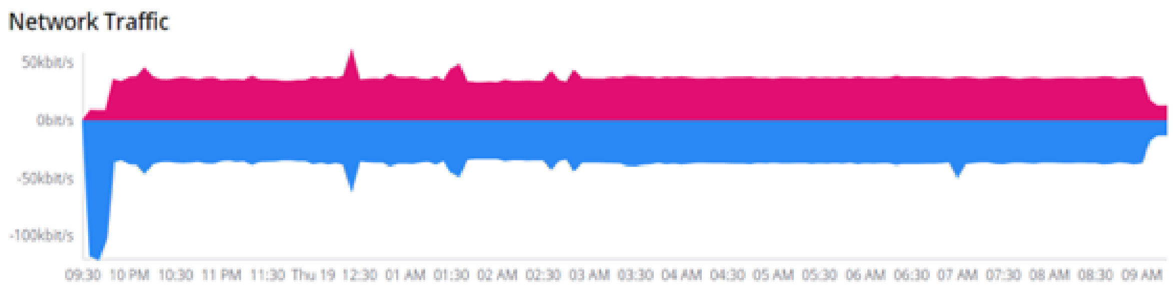


Figure 10: Product prediction network traffic.

4.3.6.3. Computationally Heavy Steps – Hazard Prediction

In this step we used Elastic stack's monitoring tool Kibana to inspect in more detail the server's resource drainage (*CPU usage, Memory Usage, Networking Traffic*) when the process of classifying the hazard category based on our input values is executed (*PredictHazard*).

The figures illustrated above present the completion time, CPU usage, network traffic and memory usage of the whole stack, throughout our experimentation for this specific step. We notice a rise in the CPU usage (*Figure 8*) when this task is executed that reaches roughly 21% usage and stabilizes around that percentage for the rest of its cycle. Memory usage is approximately 5.3GB for this step. The network traffic appears to have a spike of 302kbps output at the time we initiate the prediction and then it stabilizes at a low and balanced rate. Our produced dataset seems to have an evenly distributed output so the reason behind this spike is system usage in parallel with our hazard prediction task.

Overall, we can conclude that the CPU usage is optimal for this step. The memory usage is slightly below the optimal level as more than 60% is used for this task. The network traffic can be considered minor and balanced. The spike is not to be considered problematic.

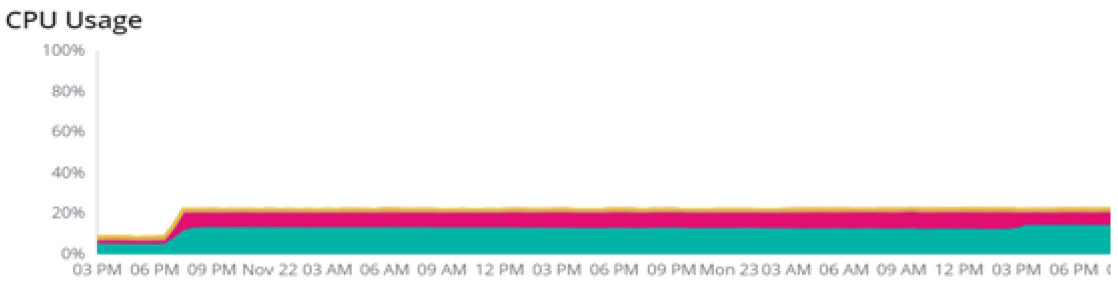


Figure 11: Hazard prediction memory usage.

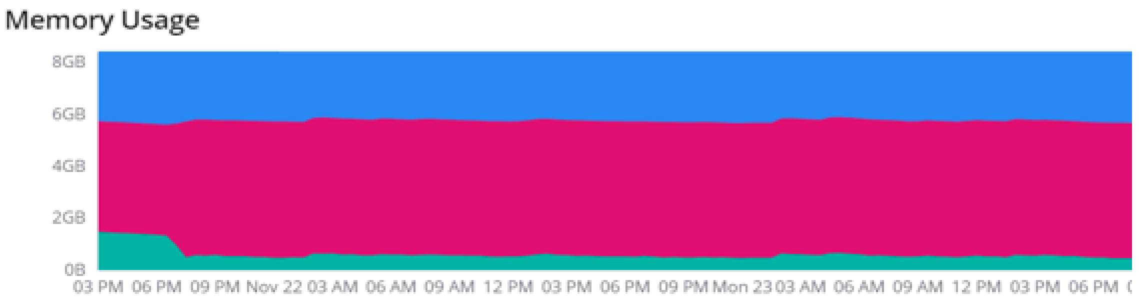


Figure 12: Hazard prediction network traffic.

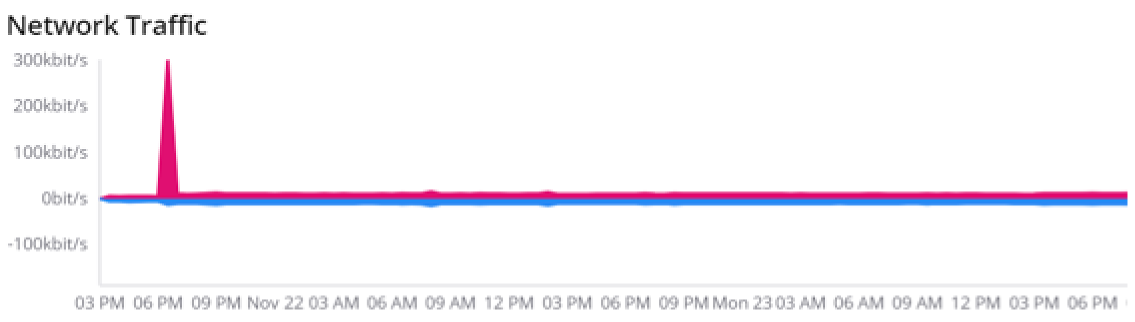


Figure 13: Hazard prediction network traffic.

4.3.6.4. Computationally Heavy Steps – Risk Estimation

In this step we used Elastic stack's monitoring tool Kibana to inspect in more detail the server's resource drainage (*CPU usage, Memory Usage, Networking Traffic*) when the process of risk estimation is executed (*riskEstimationAPI*).

The figures illustrated above present the completion time, CPU usage, network traffic and memory usage of the whole stack, throughout our experimentation for this specific step. We notice that the CPU usage is steady and low throughout the whole cycle. Memory usage is approximately 5.3GB for this step. The network traffic has its highest peak at the beginning of this step, as we send and receive a lot of data to perform risk prediction. consequently, the process of batch generating risk time series is being executed. Finally, the input/output flow becomes minimal for the rest of the cycle due to no data being sent or received.

Overall, we can conclude that the CPU usage is optimal for this step. The memory usage is slightly below the optimal level as more than 60% is used for this task. The network traffic can be considered minor and expected.

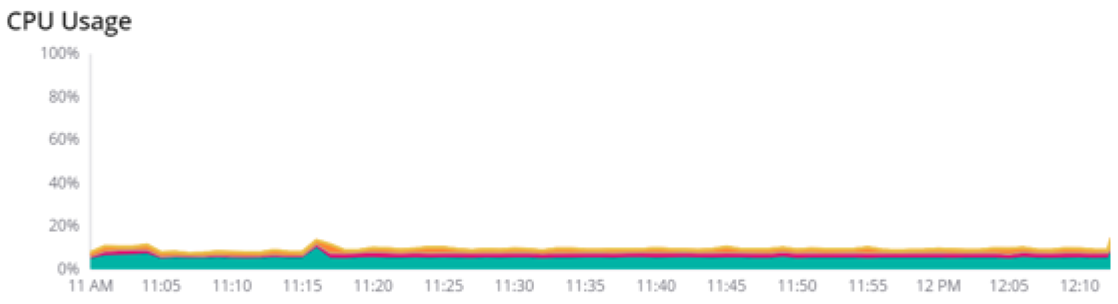


Figure 14: Risk Estimation CPU usage.

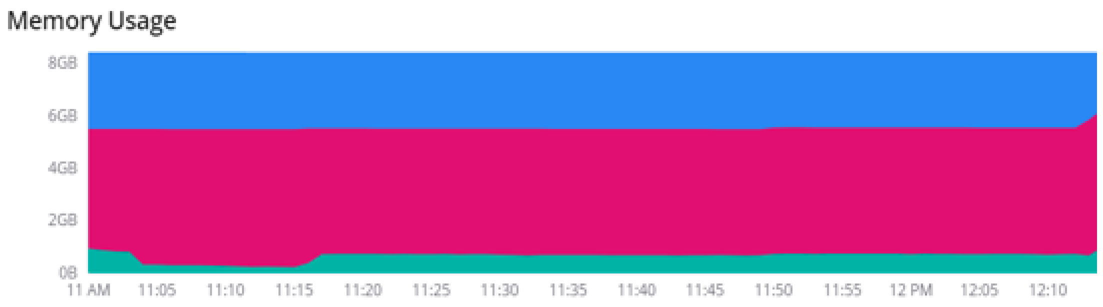


Figure 15: Risk Estimation network traffic.

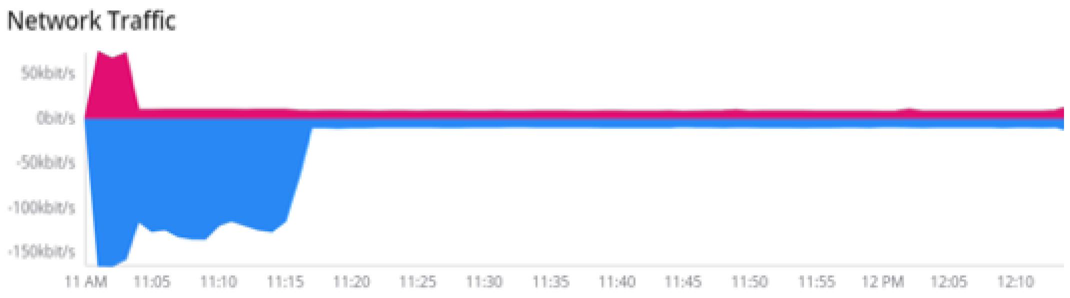


Figure 16: Risk Estimation network traffic.

The figures illustrated above present the completion time, CPU usage, network traffic and memory usage of the whole stack, throughout our experimentation for this specific step. We notice that the CPU usage is steady and low throughout the whole cycle. Memory usage is approximately 5.3GB for this step. The network traffic has its highest peak at the beginning of this step, as we send and receive a lot of data to perform risk prediction. consequently, the process of batch generating risk time series is being executed. Finally, the input/output flow becomes minimal for the rest of the cycle due to no data being sent or received.

Overall, we can conclude that the CPU usage is optimal for this step. The memory usage is slightly below the optimal level as more than 60% is used for this task. The network traffic can be considered minor and expected.

4.3.6.5. Computationally Heavy Steps – Ingesting data

In this step we used Elastic stack's monitoring tool Kibana to inspect in more detail the server's resource drainage (*CPU usage, Memory Usage, Networking Traffic*) when the process of ingesting our final dataset, in JSON form, is executed (*elasticDataSend*).

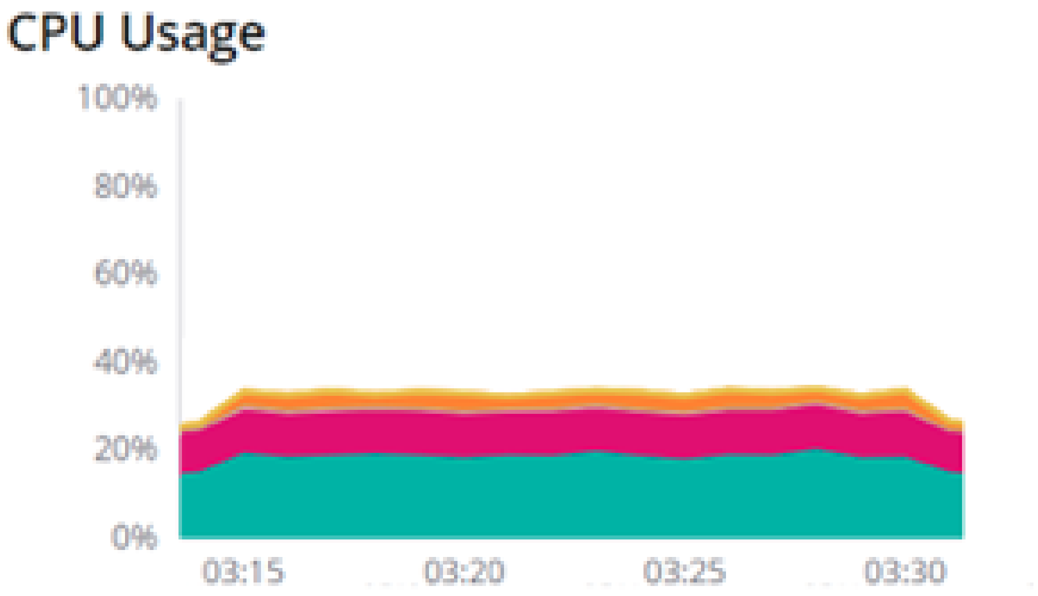


Figure 17: Ingesting data network traffic.

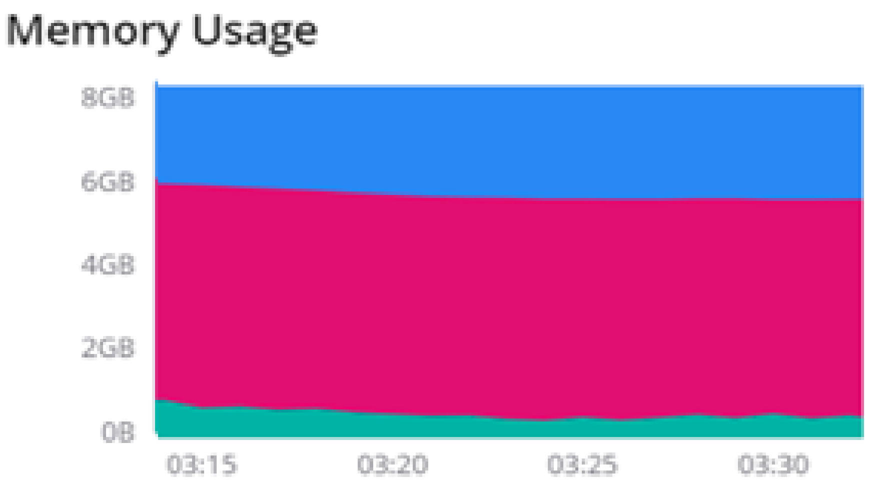


Figure 18: Ingesting data network traffic.

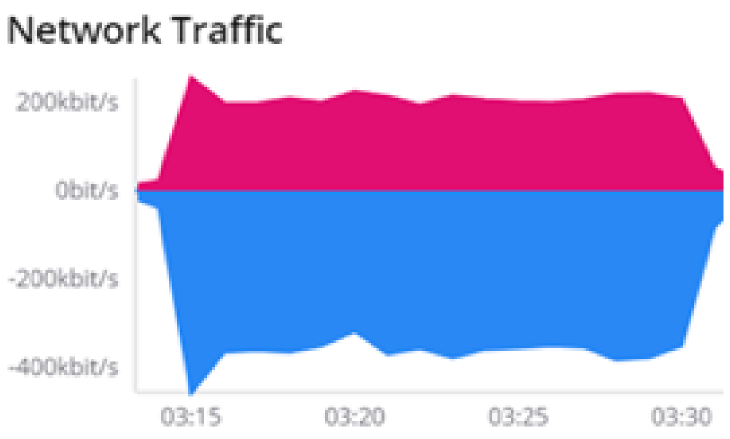


Figure 19: Ingesting data network traffic.

The figures illustrated above present the completion time, CPU usage, network traffic and memory usage of the whole stack, throughout our experimentation for this specific step. We notice that the CPU usage is steady throughout the execution cycle and it ranges at around 31%. We have a low usage of CPU because we chose to send our data with a batch size of 1000 as a threshold. The Memory usage is approximately 5.2GB for this step. The network traffic appears to be almost balanced, however, it is expected to have a bigger input flow as we send batches of data and receive only a status response.

Overall, we can conclude that the CPU usage is optimal for this step. The memory usage is slightly below the optimal level as more than 60% is used for this task. The network traffic can be considered minor, balanced, and expected.

4.3.7. Conclusions

In this scalability testing we studied the end to end scalability of the Big Data Platform for projected datasets in the Food Protection use case. According to the findings for a projected dataset that is 2.1 times larger compared to the current dataset that we are processing for food safety incidents, the workflow performed well for steps such as collection, processing etc.

Based on our Kibana monitoring results, we found that our current system performs well and has no significant performance bottlenecks with our synthetically forged dataset.

- The CPU performance was optimal for all the steps. All data was sent in batches with a threshold. The threshold was set to 1000 as, even in real life scenarios, there will not be need for a higher threshold.
- RAM usage was approximately 60% for all the steps. An upgrade could be considered in a slightly larger scale of time.
- Network traffic was minor and expected due to the nature of input/output data flows. Network traffic is considered optimal as well.

In conclusion, our Big Data Platform can scale well for the needs of our future dataset by 2030 with its current specifications but a few improvements could be made.

For instance, the task of classifying a product and hazard is very time consuming. This could be solved in two ways. First, we could consider an upgrade in our RAM and secondly, and more importantly, we could consider parallelization. Parallel execution is the ability to apply multiple CPU and I/O resources to the execution of a single operation. In our case, we could parallelize the process of predicting the product and the process of predicting the hazard so that they are executed at the same time. In this scenario, this would improve our total time by 11 hours and 19 minutes which is the minimum execution time between the two (*PredictProduct)*.

We can also notice that hazard prediction lasts approximately 48 hours. A solution to reduce that would be parallelization as well. In more detail, we know that the hazard prediction model is time consuming and we also know that we deal with a large dataset. So, a solution would be to split the dataset in half and parallelize the hazard prediction process for the two datasets reducing the total execution time to 24 hours.

Moreover, it is important to point out that this testing scenario is not a representation of real-life data flow. In this scalability testing, we download and process all our data in once. We also predict the product and hazard and estimate the risk for all our data in once. In reality, however, data is gradually produced and processed leading to lower demands.

Concerning the velocity of the data (incidents), nowadays we deal with approximately 30 records per day. If this number were to be upscaled, there would be no scalability issues as this experiment covers a much bigger incident request number (1000 incident requests).

[1] A measure of the differences between values (sample or population values) predicted by a model.

5.PROJECTION BASED ESTIMATIONS

5.1. METHODOLOGY

Besides end to end testing of the stack for the more complex pilots, we have done projection based estimations for all the pilots. This section describes the process and results.

The BDG platform collects a variety of types of data and integrates them onto a coherent knowledge graph. An essential part of the chosen methodology is to categorise these individual data sources in order to better understand how they scale given different parameters. For this we have built a model based on four relevant facets of for each data source. They are:

- **Collection mode:** Whether the data collected is the result of a deliberate effort by a human operator or the data comes from a continuous stream such as a fixed or mobile sensor. Laboratory analysis of samples is a prototypical example of manually collected datasets and meteorological data - an example of a stream.
- **Scaling Dimensions.** This is without doubt the most important parameter as different datasets scale differently with respect to time and area. Some are completely temporarily static, effectively one-time observations. Such are soil characteristics of a given field or its geographical boundaries. Other datasets scale strongly with time but very weakly with area. Sucha are meteorological observations, where in general there is only one sensor per field, which collects data at small temporal intervals.
- **Update frequency** is straightforward and measures how often the data is updated. This can vary form once such a field's boundaries, through once a year such as measuring the yield of a harvest to several times an hour as the aforementioned meteorological observations.
- **Spatial granularity** is the equivalent of frequency at the spatial level. DIfferent datasets are collected at different spatial resolutions. Meteorological data is usually collected at the level of the field or even at the level of the province or general geographical area. On the other hand, data such as soil conductivity measurement scales at the rate of one data-point at every few square centimeters.
- **Others**. Although not taken into account in this study, other potential dimensions of scalability were discussed with the data owners. These are related to probable or possible technological evolution. Such can be the multiplication of the number of sensors on a given operation. The expected launch of new satellites, which will double the number of observations and the

During the conversion to RDF data points are converted to Observations (**see D3.1**). This is the turtle representation of a set of observations of soil conductivity at different depths of soil. As we can see each observation is represented as a fact or RDF triple.

```
<data/tableGrape/Fasoulis/Geotrisi/EM38-mk2/2018-05-23T12:26:30> a qb:Observation;
 qb:dataSet        <data/tableGrape/Fasoulis/Geotrisi/EM38-mk2>;
 bdg:position       <data/tableGrape/Fasoulis/Geotrisi/EM38-mk2/2018-05-23T12:26:30/pos>;
 bdg:dateTime        "2018-05-23T12:26:30"^^xsd:dateTime;
 bdg:positionQuality <positionQuality-1>;
 bdg:satellites     12;
 bdg:HDOP          0.69;
 bdg:CV1m          144.699;
 bdg:CV05m          106.74.

<data/tableGrape/Fasoulis/Geotrisi/EM38-mk2/2018-05-23T12:26:30/pos> a geo:Geometry ;
   geo:asWKT "Point Z(22.590 37.816 299.612)"^^geo:wktLiteral .
```

Given the homogenization process we can reliably count the number of generated triples per dataset to have a unique measure of the volume of data across datasets.

5.2. Growth pattern estimation

Given the heterogeneity and the large number of individual datasets we will estimate their scaling in time and space using a technique similar to that used to estimate aloroi complexity. The Big O notation is a technique in computer science used to classify algorithms according to how their run time or space requirements grow as the input size grows[6].

While the original method is used to estimate execution time over storage space we will tweak the technique to estimate the required data storage space for data scaling over the two main dimensions - Time (T) and Area (A).

We analyse each dataset and estimate how it grows over these two dimensions. It is key to point out that we approximate in orders of magnitude where we estimate how the growth of the number of observations is related to any of the two variables. Here are 4 examples illustrating this approach.

IoT stationary data. This dataset consists of meteorological data collected by a sensor array on a given field. It's spatial granularity is very low as the recorded phenomena such as temperature and humidity don't vary spatially at the resolution implied by farming. The weather is the same for the neighbours field also. However temporally data accumulates pretty quickly. This particular dataset will grow at a rate of 1 set of 10 observations every 20 minutes. The growth formula is thus **10^3t + A** meaning that the number of observations gros cubically with time and linearly with area.

Eca sensing consists of data about the electrical conductivity of the soil of a given field. It is a costly and time consuming activity and requires pulling a sensor either by hand or with a tractor or ATV and covering the entire field. The result is a very fine resolution scan of the field with one data point for every 10cm of distance. On the other hand the measurement is done only once for a field so it does not scale with time at all. The growth formula is thus **1 + 10^5A** where as we can see the time dimension is not at all present

Sentinel-2 is satellite data collected land filtered out of the Sentinel 2 data stream. It has several data points for every 10m2 "pixel" of land roughly[7] every 2.5 days. This dataset scales both with area and with time and its approximation formula formula is **10^2t + 10^4A**

Grapevine images is a dataset consisting of pictures of grapevine plants took in laboratory by the PhenoArch phenotyping platform. The platform collected pictures of plants every two following days and pictures were taken from different side-angles. On a regular basis, plants are analyzed by experts that annotate the number of leaves of each plant. Despite the former activity is performed by making usage of automatic mechanisms (robots, conveyor belts) the latter is manually achieved and it is particularly time consuming. The growth formula is thus **10A** where as we can see the time dimension is not at all present while area dimension, i.e., the area devoted to collect plant pots, is responsible of the scaled dataset.

[6] https://en.wikipedia.org/wiki/Big_O_notation

[7] Due to possible cloud cover, not every passage of the satellite above a field yields usable data.

5.3. TABLE AND WINE GRAPES PILOT

The table and wine grapes pilot focuses on the prediction of the quality of the final product of table and wine grapes. This is achieved through the provided datasets of this pilot; datasets that involve soil and weather data, as well as crop qualitative and quantitative data. In the following sections will analyze the data patterns for each dataset based on "Dataset Analysis & Evaluation" from deliverable D7.2. We will describe the data generation approach and the result of the data projection. Finally, we will identify the data flows of this specific use case inside the stack and experiment on each of them, using the metrics and the methodology described in the previous section.

5.3.1. Data Usage Patterns

This pilot has provided 7 different datasets that cover 3 different fields, resulting in a total of 22 datasets, since one of the fields is lacking an IoT installation. In the next version of the deliverable, we will further describe the data patterns of these datasets in order to find the best data generation approach.

5.3.2. Projection Results

This pilot produces more varied data that scales mostly with area.
The meteorological data that scales with time

Dataset	**Update freq**	**Approximation**	**Formula**
Yield Mapping	1Y	t	t + 10^3*A
Grape and berry mechanical properties	1Y	t	t + 10^3A
Classical analytical techniques (HPLC)	1Y	t	t + 10^3A
Topographic data and elevation maps	once	1	1 + 10A
RapidScan	6xY	10t	10t + 10^2A
SpectroSence	6xY	10t	10t + 10^3A
CropCircle	6xY	10t	10t + 10^4A
Photosynthesis Data	3xY	t	t + 10^3A
IoT stationary data	5Min	10^3t	10^3t + A
Drone imagery			+
Eca sensing	once	1	1 + 10^4A
Soil Analysis Data (AUA)	once	1	1 + 10^3A
Mechanical Props - Palivou ALL	1Y	t	t + 10^3A
AUA CellGrid	once	1	1 + 10^4A

5.4. Farm management Pilot

The farm management pilot aims at developing a system that can support the farmer in his/her data collection, as well as in his/her day to day work. To that end, this specific pilot has granted access to datasets varying from satellite images and their respective processing to environmental and field indicators tracking by sensors deployed in the fields. In the following sections will analyze the data patterns for each dataset based on "Dataset Analysis & Evaluation" from deliverable D7.2. We will describe the data generation approach and the result of the data projection. Finally, we will identify the data flows of this specific use case inside the stack and experiment on each of them, using the metrics and the methodology described in the previous section.

5.4.1. Data Usage Patterns

This pilot is characterized by datasets concerning satellite images from the external services of Sentinel-2 and Landsat-8 and the respective processing. All of this data is accessible through API endpoints provided by GEOCLEDIAN. It is also characterized by datasets concerning the environmental indicators and sensor data coming from deployed installations on the respective fields. The growth rate of the data for the last 10 years will be estimated and will be used to generate the projected datasets.

5.4.2. Projection results

The farm management pilot produces mostly sensor data scaling over the temporal dimension. This table summarizes the results of the data accumulation projections. We can see that the sensor data scales roughly 1000 times more with time than with area.

Dataset	Update freq	Approximation	Formula
Relative Humidity	30min	10^3t	10^3t + A
Air Temperature	30min	10^3t	10^3t + A
Global Solar Radiation	30min	10^3t	10^3t + A
Wind Speed and Direction	30min	10^3t	10^3t + A
Soil Temperature	30min	10^3t	10^3t + A
Soil Moisture	30min	10^3t	10^3t + A
Precipitation	30min	10^3t	10^3t + A
Infrared Surface Temperature	30min	10^3t	10^3t + A

6.CONCLUSIONS

In this document described rigorous testing based on synthetic data for 3 pilots and projections based results for the two remaining pilots. These experiments show the capability of the BDG stack and the chosen approaches to handle the estimated data growths.

7.REFERENCES

Rabl T., Frank M., Sergieh H.M., Kosch H. (2011) A Data Generator for Cloud-Scale Benchmarking. In: Nambiar R., Poess M. (eds) Performance Evaluation, Measurement and Characterization of Complex Systems. TPCTC 2010. Lecture Notes in Computer Science, vol 6417. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-18206-8_4

A. Ghazal, T. Rabl, M. Hu, F. Raab, M. Poess, A. Crolotte, and H.-A. Jacobsen, "BigBench: Towards An Industry Standard Benchmark for Big Data Analytics," in SIGMOD, 2013, pp. 1197–1208.

Y. Tay, B. T. Dai, D. T. Wang, E. Y. Sun, Y. Lin et al., UpSizeR: Synthetically scaling an empirical relational database, Information Systems, vol.38, issue.8, pp.1168-1183, 2013.

Max Troell, Rosamond L. Naylor, Marc Metian, Malcolm Beveridge, Peter H. Tyedmers, Carl Folke, Kenneth J. Arrow, Scott Barrett, Anne-Sophie Crépin, Paul R. Ehrlich, Åsa Gren, Nils Kautsky, Simon A. Levin, Karine Nyborg, Henrik Österblom, Stephen Polasky, Marten Scheffer, Brian H. Walker, Tasos Xepapadeas, Aart de Zeeuw
Proceedings of the National Academy of Sciences Sep 2014, 111 (37) 13257-13263; DOI: 10.1073/pnas.1404067111

OECD, A Green Growth Strategy for Food and Agriculture: PRELIMINARY REPORT©, (2011), url: https://www.oecd.org/greengrowth/sustainable-agriculture/48224529.pdf

Shen E. and Antova L, (2013), Reversing Statistics for Scalable Test Databases Generation, Proceedings of the Sixth International Workshop on Testing Database Systems

Rabl, Tilmann and Danisch, Manuel and Frank, Michael and Schindler, Sebastian and Jacobsen, Hans-Arno, (2015), Just Can't Get Enough: Synthesizing Big Data, Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 1457–1462, Melbourne, Victoria, Australia

Simard, Patrice Y., David Steinkraus, and John C. Platt. "Best practices for convolutional neural networks applied to visual document analysis." Icdar. Vol. 3. No. 2003. 2003.