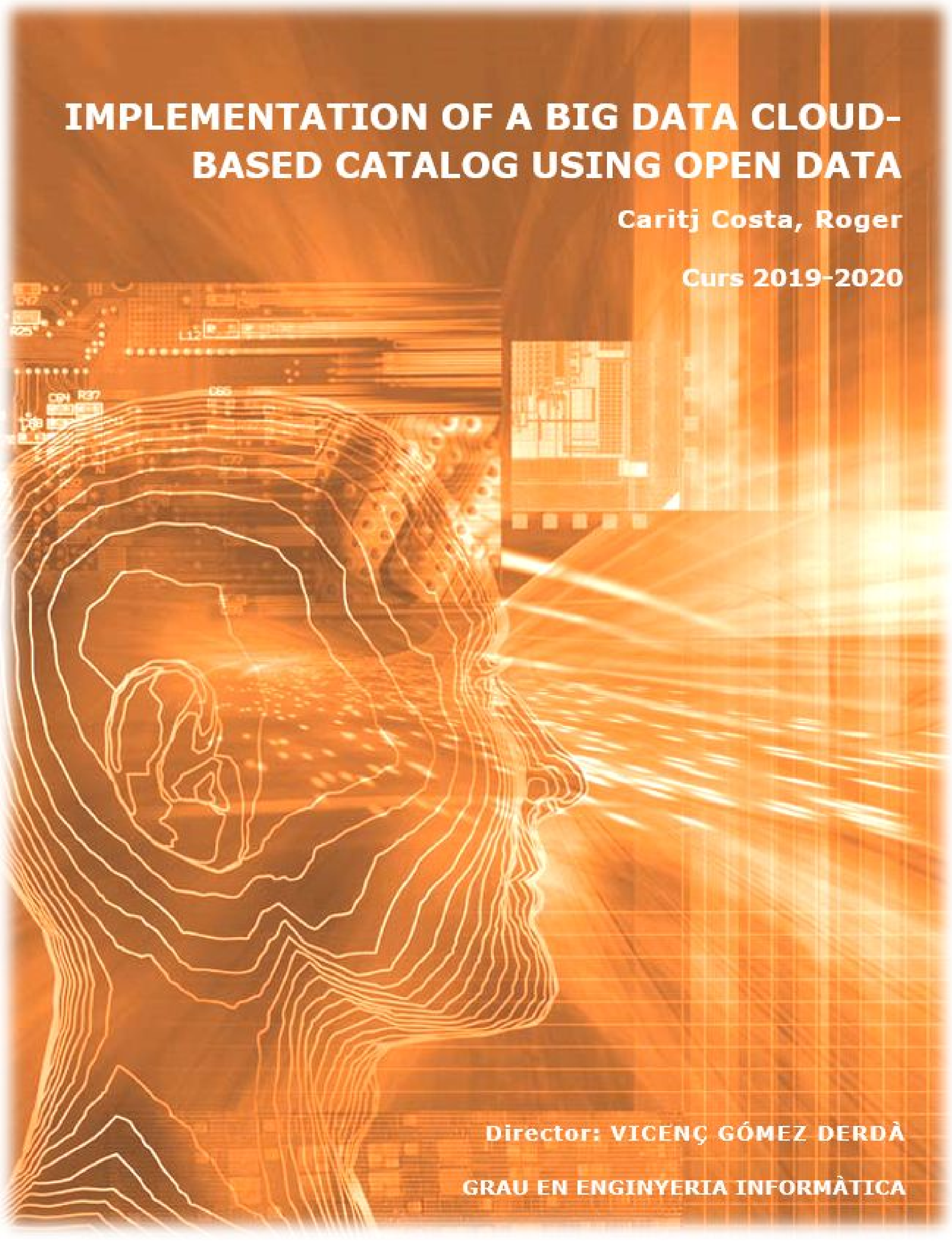


IMPLEMENTATION OF A BIG DATA CLOUD-BASED CATALOG USING OPEN DATA

Caritj Costa, Roger

Curs 2019-2020

Director: VICENÇ GÓMEZ DERDÀ

GRAU EN ENGINYERIA INFORMÀTICA

Universitat Pompeu Fabra Barcelona

Escola Superior Politècnica

Treball de Fi de Grau

IMPLEMENTATION OF A BIG DATA CLOUD-BASED CATALOG USING OPEN DATA

CARITJ COSTA, ROGER

TREBALL FI DE GRAU
GRAU EN ENGINYERIA EN INFORMÀTICA
ESCOLA SUPERIOR POLITÈCNICA UPF
CURS 2019-2020

Project Director from ServiZurich:
Sánchez Ros, Jose Luis

Supervisor from UPF:
Gómez Cerdà, Vicenç

"To my father, mother, and sister,
my idols and supporters."

ACKNOWLEDGMENTS

I would like to acknowledge everyone who played a role in my end-of-degree project. First of all, my parents and sister, who supported me with love and understanding. Without you, I could never have reached this current level of success. You teach me the importance of trying to reach excellence in what passionate us.

Secondly, my project director Jose Luis Sanchez Ros and my supervisor Vicenç Gómez Cerdà. Thanks to Jose Luis I had the opportunity to explore this field and all the technology behind, you were not a boss but a leader in this project. If it were not for Vicenç this document would be very different in style and format. Your guidance in this, my first serious project, is proof that the devil is in the details.

Last but not least, all the people in ServiZurich who shared his knowledge with me and treated me like part of a family for almost a year. Some of them are Victor Vallejo, Carlos Aires, Alicia Sanchez, Jesus Segura, Marta Tolos, Aurelien Remy Antoine, Pere Brull, Francesc Pelegrin, Ester Suñer, Joan Ficapal, Marc Vila, Angela Marques, and Pere Llimona.

PROLOGUE

Data is power. Nowadays we are all aware of it. More data leads to smarter decisions. Huge companies can change the world based on this fact. Google knows everything I search on the web, Amazon knows what stuff I buy, even the more private things, Facebook knows which people I hang out with. All this data retrieved by big companies is very powerful, as they proved. And all these private companies can use our private data in tons of different ways.

It is easier for companies to work with their own data since they control everything about it: the format, the period they retrieve it, how to store it, how to retrieve it, etc. Since this end-of-grade project is developed in Zurich, an insurance company, we are interested in data related to risks and direct danger for properties. It was shown that natural disasters take their toll on insurance companies [1]. How can they succeed when they don't have the infrastructure to measure these data?

Open Data is the forgotten youngest brother. There is no general rule when updating open data, so it is very difficult to work with more than one dataset at the same time. What could happen if we take several open datasets with regularity, in order to get good quality and updated data, and make them capable of working together for a concrete objective? How smart could our decisions be if we use only data that is freely available for everybody?

This project is an initial attempt to improve the usage of this open data, the first step towards a practical tool that can be the basis of any data-driven company in the future.

ABSTRACT

In this project, we will develop a cloud-based data catalog. We will first determine the services that will be necessary and afterward build the infrastructure in Azure. Then we will explore the Open Data sources available and evaluate them to see if they fit our purposes.
Once they are selected we will periodically retrieve and transform this data to a similar format in order to work with all the datasets at the same time. Once this part is concluded, it will provide us an automatically updated data catalog with information from different sources. We will also see some use cases and will start using this data to give useful risk-related information.

RESUM

En aquest projecte, desenvoluparem un catàleg de dades basat en el núvol. Mostrarem els serveis que seran necessaris i aixecarem la infraestructura a Azure, després explorarem les fonts de dades obertes disponibles i les avaluarem per veure si s'ajusten a les nostres necessitats. Un cop seleccionades, les extraurem periòdicament i les transformarem en un format similar per poder treballar amb tots els conjunts de dades a el mateix temps. Una vegada això estigui funcionant, ens proporcionarà un catàleg de dades actualitzat automàticament amb informació de diferents fonts. També veurem algun cas d'ús i utilitzarem aquestes dades per donar informació útil relacionada amb el risc.

RESUMEN

En este proyecto, desarrollaremos un catálogo de datos basado en la nube. Mostraremos los servicios que serán necesarios y levantaremos la infraestructura en Azure, luego exploraremos las fuentes de datos abiertos disponibles y las evaluaremos para ver si se ajustan a nuestros necesidades. Una vez seleccionado, los extraemos periódicamente y lo transformaremos en un formato similar para poder trabajar con todos los conjuntos de datos al mismo tiempo. Una vez esto esté funcionando , nos proporcionará un catálogo de datos actualizado automáticamente con información de diferentes fuentes. También veremos algún caso de uso y utilizaremos estos datos para dar información útil relacionada con el riesgo.

INTRODUCTION

Motivation

Over the last few years, Data has been proclaimed as the new Oil [2]. That leads both people and organizations to start storing huge amounts of it. Ninety percent of the world's data was generated in the last two years [3]. The question now is: What to do with all these data? Governments and organizations have released several amounts of data for free, or what is known as Open Data.

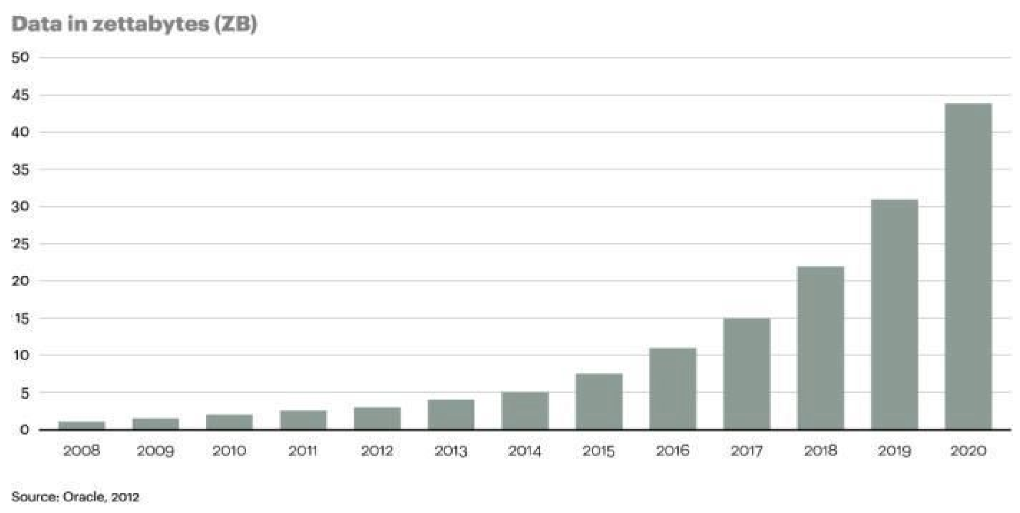


Fig. 1: Data generated per year.
(1 Zettabyte = 1 billion Terabytes)

Some parts of this data have been explored for very concrete purposes. It's not easy to work with different open datasets from different sources since there is not a pattern or a general way to extract and save it. Not even the format.

There is Open Data from a lot of different topics, we can find from worldwide economic statistics[1] to registered dog names[2]. We will mostly focus on daily nature-related data. Another problem with this kind of data is the lack of periodicity on its launch, so we can have no data for several weeks and we want to maintain an updated Data Catalog as long as it is possible.

My project director in Zurich, Jose Luis, shared with me an experiment he ran in a congress. They set a table with Lego pieces separated in two different groups: one group had the pieces classified by type and color and the other one was in a box, all mixed up. They told two

[1] Datosmacro.com. Retrieved from https://datosmacro.expansion.com/.
[2] Dog Names. (2019, March 29). Retrieved from https://catalog.data.gov/dataset/dog-names.

people to build a small Lego car in less than a minute. The person who worked with the ordered pieces built it in a few seconds while the other did not finish in the given minute.

This shows that even though the information is available out there, in the box that represents internet, we won't be able to make proper decisions and build good models if we do not have it cataloged and classified. We could end up with a car built with the wrong pieces or, in case we end up with a functional car, it could have taken too long to build it.

This project will set the basis of the data catalog for the data scientist so that in the future, the smartest decisions can be made based on our extracted data (Data-driven decisions).

Objective

The objective will be to improve the efficiency of the workflow by integrating multiple sources of open data. This data must be treated and transformed into a format that allows working with the most commonly used frameworks. We also expect the data to be daily updated, or as soon as new information is available. To prove that, we expect to have some use cases that work with all this data.

All this will be set up in the cloud, concretely, we will work with Microsoft Azure. We will use Databricks for the code and Azure Data Lake Storage (ADLS) to store the data. In the end, we expect to have:

1. A set of programs that can be automatically executed every certain amount of time, that retrieve information through APIs, requests or web scraping from different sources.
2. A Data Catalog where all this data will be stored, classified, organized and updated periodically through the programs.
3. Some use cases to show which risk-related decisions can we make using only the extracted data.

Index

Figure List

1. SETTING UP THE ENVIRONMENT

1.1 What is Microsoft Azure

"Microsoft Azure is an ever-expanding set of cloud services to help your organization meet your business challenges. It's the freedom to build, manage, and deploy applications on a massive, global network using your favorite tools and frameworks.[3]"

Leaving the marketing explanation of what is Azure aside, we could describe it as a portal with a catalog of cloud services given by Microsoft. Those services are modular and work quite well between them. It uses all the technologies and advantages involved in distributed and parallel services with an abstraction layer that allows us to think at a much higher level.

It works by what they called Azure Subscription. It grants us access to the more than 800 services that they have in the catalog. Each subscription manages its own permissions, services, security, and costs. It can be used to create different environments for different purposes, commonly: dev, test, and production. In big companies like Zurich, they have subscriptions per each country too.

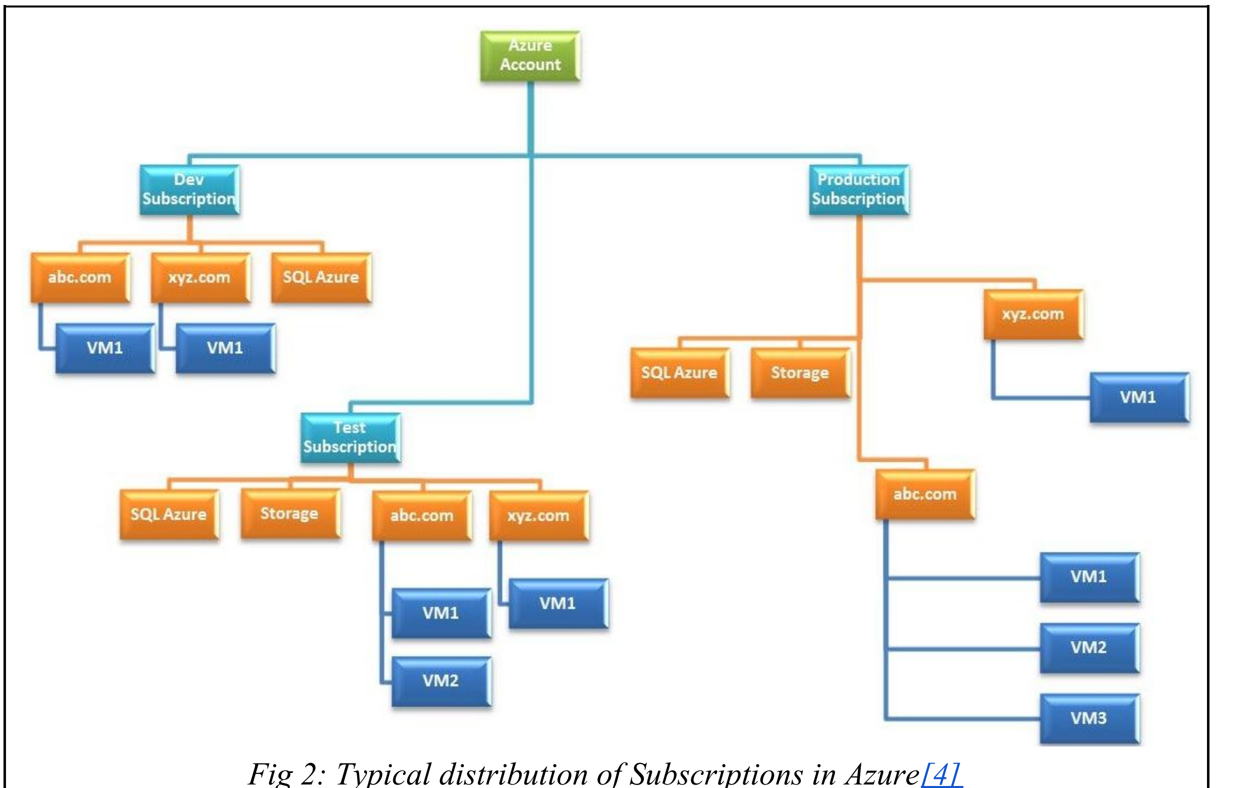


Fig 2: Typical distribution of Subscriptions in Azure[4]

[3] What is Azure-Microsoft Cloud Services?. Retrieved from https://azure.microsoft.com/en-us/overview/what-is-azure/.

1.2 Azure Data Lake Storage (ADLS)

"Azure Data Lake Storage is an enterprise-wide hyper-scale repository for big data analytics workloads. Azure Data Lake enables you to capture data of any size, type, and ingestion speed in one single place for operational and exploratory analytics.[4]*"*

In this project, we will be using ADLS as storage for our data since it is already included in Azure and its build for use cases like ours.

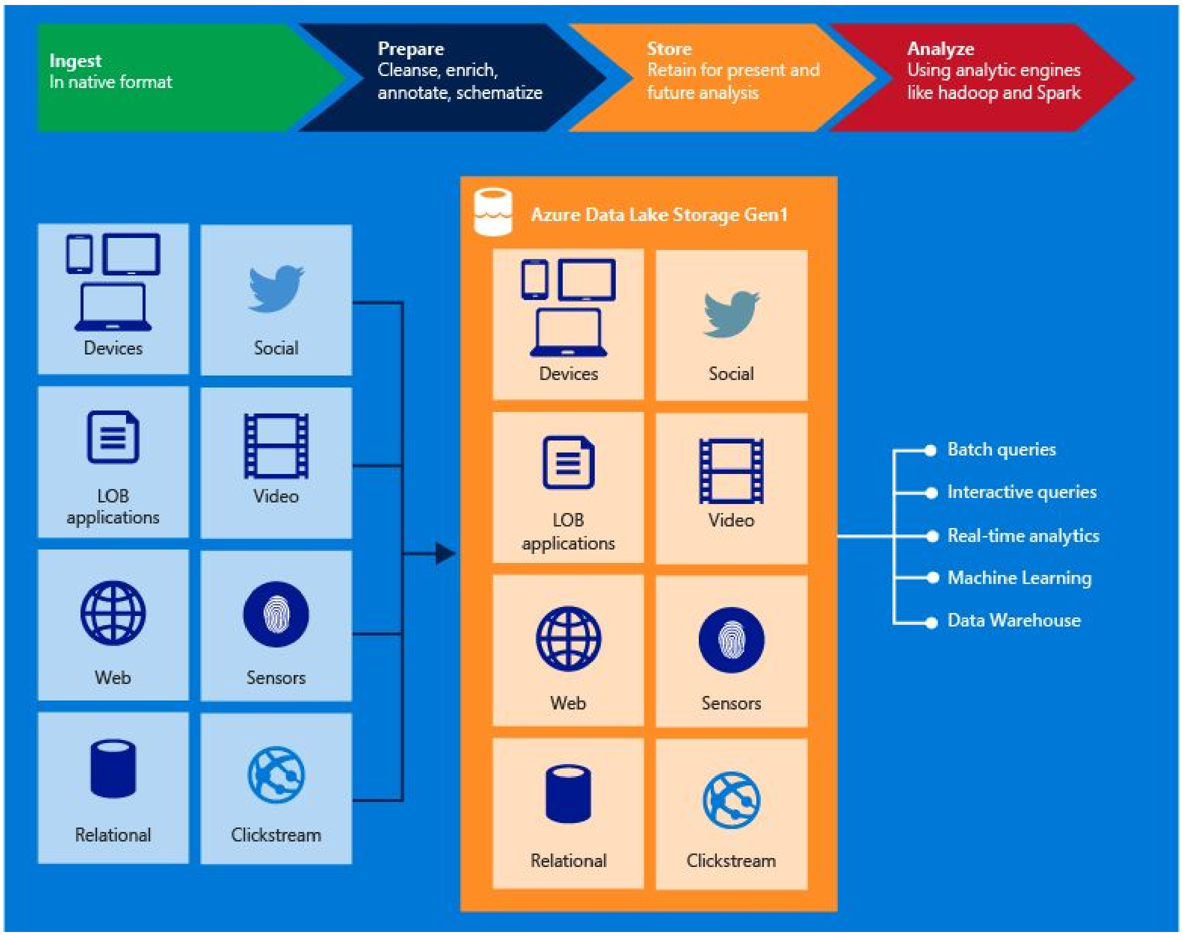


Fig 3: ADLS documentation schema

It is a distributed file system like Hadoop and it could work with all the Hadoop ecosystem. It is also compatible with Hadoop analytic frameworks such as MapReduce and Hive. It provides unlimited storage as long as we paid for it. It was created for Big Data, so it doesn't impose any limits on account size, file sizes or amount of data. Individual files can range from kilobytes to petabytes.

[4] What is Azure Data Lake Storage? Retrieved from https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-overview.

ADLS is built for running large-scale analytic systems that require massive throughput to query and analyze large amounts of data. The data lake spreads parts of a file over a number of individual storage servers. This improves the read throughput when reading the file in parallel for performing data analytics.

It provides industry-standard availability and reliability. Our data assets are stored durably by making redundant copies to guard against any unexpected failures. ADLS also provides enterprise-grade security for the stored data.

1.3 Azure Databricks

"Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. Designed with the founders of Apache Spark, Databricks is integrated with Azure to provide one-click setup, streamlined workflows, and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts.[5]*"*

Databricks is meant to be used as a Python notebook as could be Jupyter. There exists a pipeline in Azure to work with data. This involves several services that cover Ingest, Explore, Preparation and Train, Model and Serve, Store and Visualize. Every one of these tasks can be performed by a different service. Databricks work for the preparation and training of the data, but we will also use it to retrieve the data using Python.

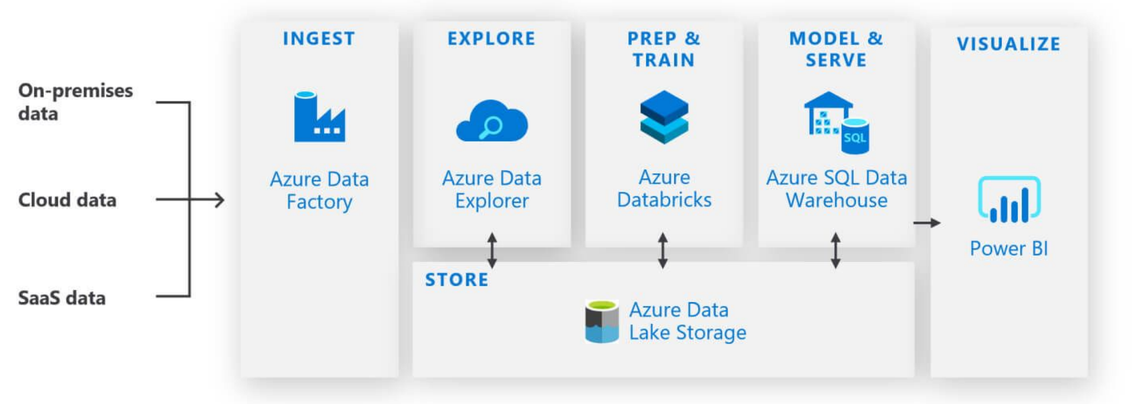
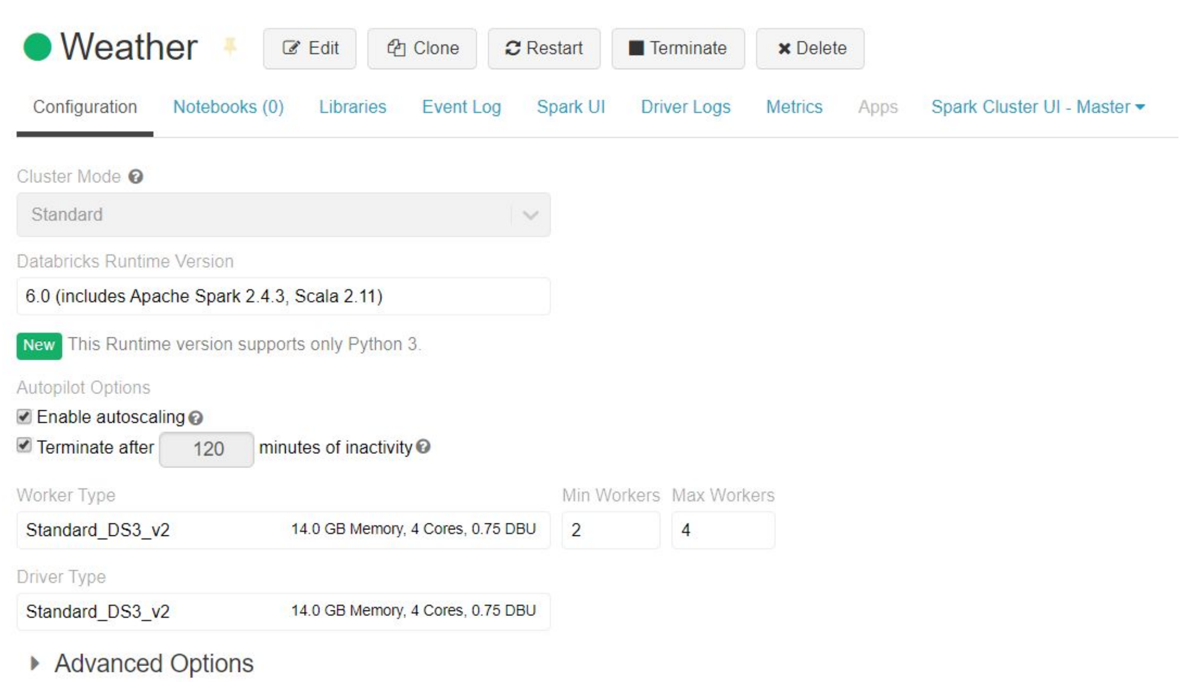


Fig 4: Recommended pipeline from Azure

Databricks provide some tools that will be very useful for us along with this project. The first one is the possibility to create different clusters for different projects and attach them to some specific notebook. Those clusters can be attached and detached any amount of time. Clusters

[5] What is Azure Databricks? Retrieved from https://docs.microsoft.com/en-us/azure/azure-databricks/what-is-azure-databricks.

are scalable and we can decide which one with which version of Python, Scala and Apache Spark version we want. We can also decide the memory, the number of cores and the number of workers desired.


Weather

Configuration | Notebooks (0) | Libraries | Event Log | Spark UI | Driver Logs | Metrics | Apps | Spark Cluster UI - Master

Cluster Mode: Standard

Databricks Runtime Version: 6.0 (includes Apache Spark 2.4.3, Scala 2.11)

New This Runtime version supports only Python 3.

Autopilot Options
- [✓] Enable autoscaling
- [✓] Terminate after 120 minutes of inactivity

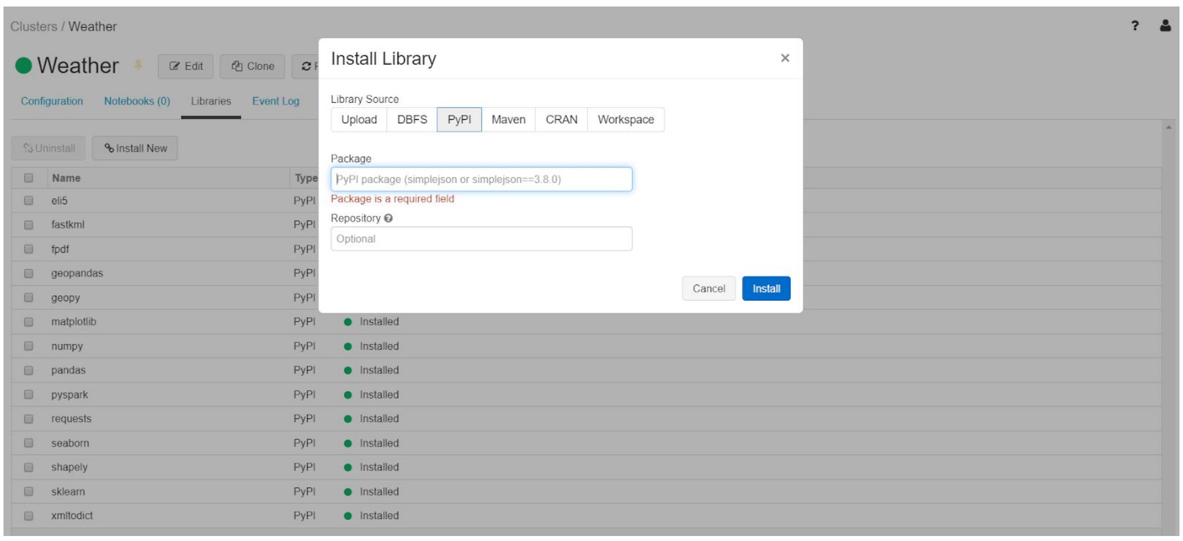
Worker Type		Min Workers	Max Workers
Standard_DS3_v2	14.0 GB Memory, 4 Cores, 0.75 DBU	2	4

Driver Type	
Standard_DS3_v2	14.0 GB Memory, 4 Cores, 0.75 DBU

Advanced Options

Fig 5: Cluster configuration window

Each cluster starts with a few libraries installed but we can add new libraries from different sources, e.g. similar to pip install for Python if we have the library in jar format, or the whl, or the egg, etc. Almost all libraries that we are going to use can be installed by PyPl (similar to pip install).


Clusters / Weather

Weather

Configuration | Notebooks (0) | Libraries | Event Log

Install Library

Library Source: Upload | DBFS | [✓ PyPI] | Maven | CRAN | Workspace

Package: PyPI package (simplejson or simplejson==3.8.0)

Package is a required field

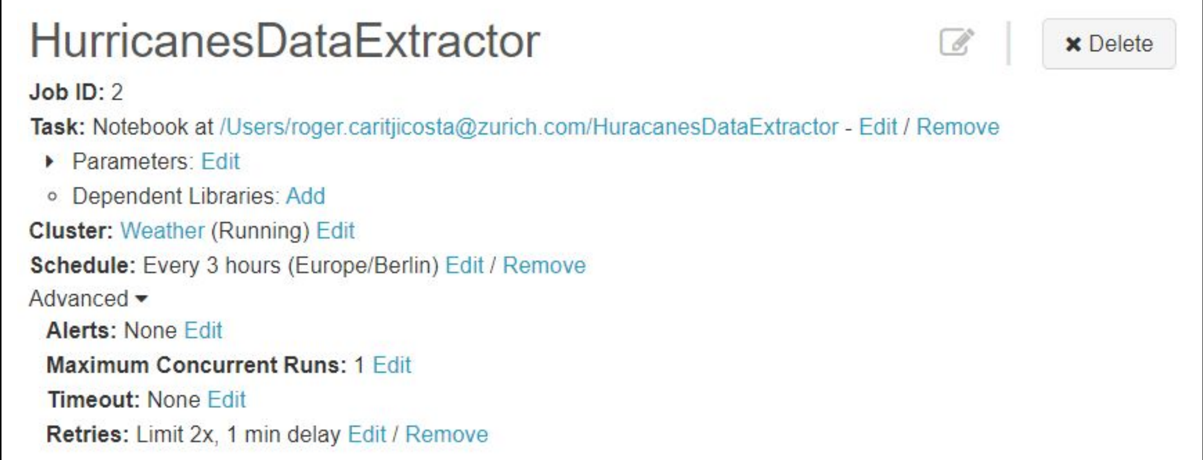
Repository: Optional

Cancel | Install

Name	Type	
eli5	PyPI	
fastkml	PyPI	
fpdf	PyPI	
geopandas	PyPI	
geopy	PyPI	
matplotlib	PyPI	● Installed
numpy	PyPI	● Installed
pandas	PyPI	● Installed
pyspark	PyPI	● Installed
requests	PyPI	● Installed
seaborn	PyPI	● Installed
shapely	PyPI	● Installed
sklearn	PyPI	● Installed
xmltodict	PyPI	● Installed

Fig 6: Installing a new library with PyPl

Another good feature of our project is job scheduling. This part will be very important since we want to retrieve data every day to actualize our datasets. Here we must declare which Notebook will be executed, with which parameters, if necessary, and dependent libraries. We decided to assign our cluster as a default since it worked during all the tests. We can decide the period for the job. Another interesting feature is that it can send emails when it fails or succeeds.



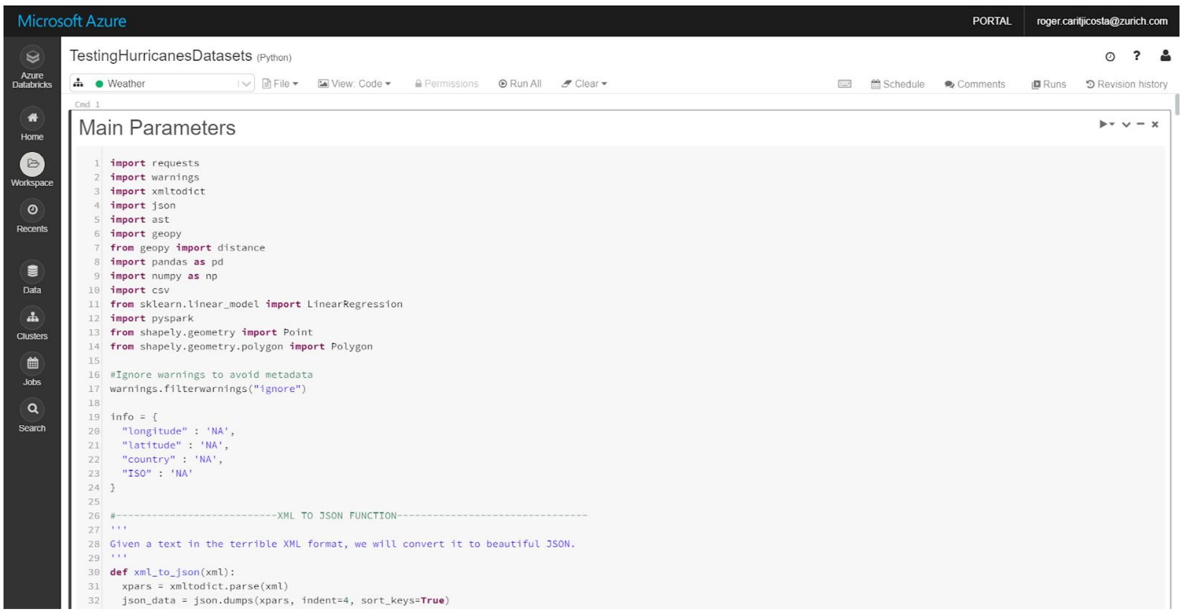
HurricanesDataExtractor

Job ID: 2
Task: Notebook at /Users/roger.caritjicosta@zurich.com/HuracanesDataExtractor - Edit / Remove
- Parameters: Edit
- Dependent Libraries: Add

Cluster: Weather (Running) Edit
Schedule: Every 3 hours (Europe/Berlin) Edit / Remove
Advanced
Alerts: None Edit
Maximum Concurrent Runs: 1 Edit
Timeout: None Edit
Retries: Limit 2x, 1 min delay Edit / Remove

Fig 7: Job configuration window

Finally, the main tool of Databricks is the notebook. There we will program the main part of this project. Here you can program in some programming languages by writing some commands at the beginning of each cell but we will mainly use Python. It is very similar to Jupyter Notebooks but with the extra thing that you can edit notebooks online, so more than one person can be coding and executing at the same time.



TestingHurricanesDatasets (Python)

Main Parameters

```python
import requests
import warnings
import xmltodict
import json
import ast
import geopy
from geopy import distance
import pandas as pd
import numpy as np
import csv
from sklearn.linear_model import LinearRegression
import pyspark
from shapely.geometry import Point
from shapely.geometry.polygon import Polygon

#Ignore warnings to avoid metadata
warnings.filterwarnings("ignore")

info = {
  "longitude" : 'NA',
  "latitude" : 'NA',
  "country" : 'NA',
  "ISO" : 'NA'
}

#--------------------------XML TO JSON FUNCTION--------------------------------
'''
Given a text in the terrible XML format, we will convert it to beautiful JSON.
'''
def xml_to_json(xml):
  xpars = xmltodict.parse(xml)
  json_data = json.dumps(xpars, indent=4, sort_keys=True)
```

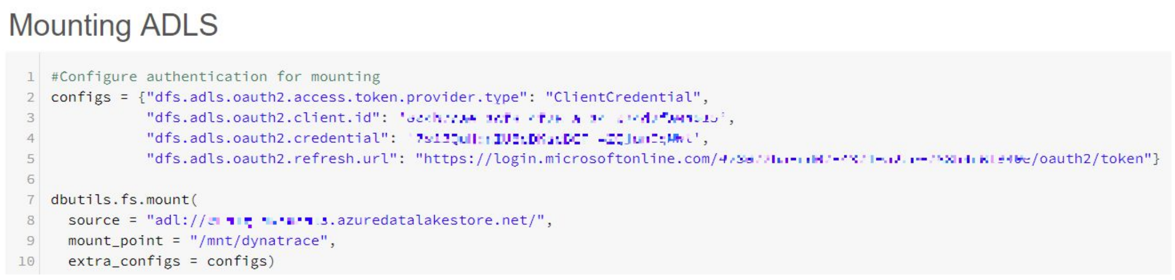
Fig 8: Example of Azure Databricks Notebook

1.4 Linking ADLS with Databricks

“In this section, we will describe the necessary steps to link our ADLS and Databricks so we can work with files stored in ADLS as local files.”

We want to store data in our Data Lake (ADLS) but we will work from Databricks. That means that we must connect in some way these two services. As we saw previously, those services are meant to work together, so it shouldn’t be a big deal. This can be done in different ways, we will do it in Python from the Databricks Notebook.

Every service in Azure has some identifiers. When we connect two services we must find some of them, like the client-id, the credentials, the refresh URL and the source.

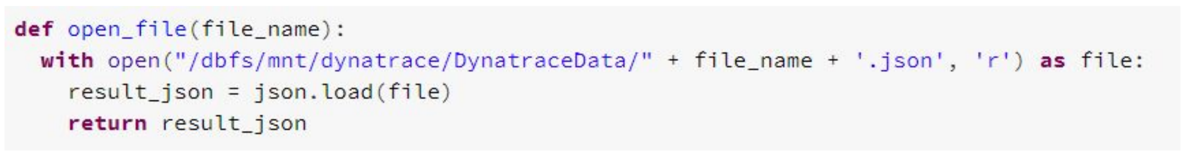


Mounting ADLS

```
#Configure authentication for mounting
configs = {"dfs.adls.oauth2.access.token.provider.type": "ClientCredential",
           "dfs.adls.oauth2.client.id": '[redacted]',
           "dfs.adls.oauth2.credential": '[redacted]',
           "dfs.adls.oauth2.refresh.url": "https://login.microsoftonline.com/[redacted]/oauth2/token"}

dbutils.fs.mount(
  source = "adl://[redacted].azuredatalakestore.net/",
  mount_point = "/mnt/dynatrace",
  extra_configs = configs)
```

Fig 9: Code to connect with the ADLS, credentials censored

Once this is done there is no need to execute it never again. From now on we can access our ADLS as if it was our local directory on the computer. We can open, close and create files using the common function in Python without worrying about all the distributed and parallel mechanisms behind.



```
def open_file(file_name):
  with open("/dbfs/mnt/dynatrace/DynatraceData/" + file_name + '.json', 'r') as file:
    result_json = json.load(file)
    return result_json
```

Fig 10: Working with ADLS files as local files

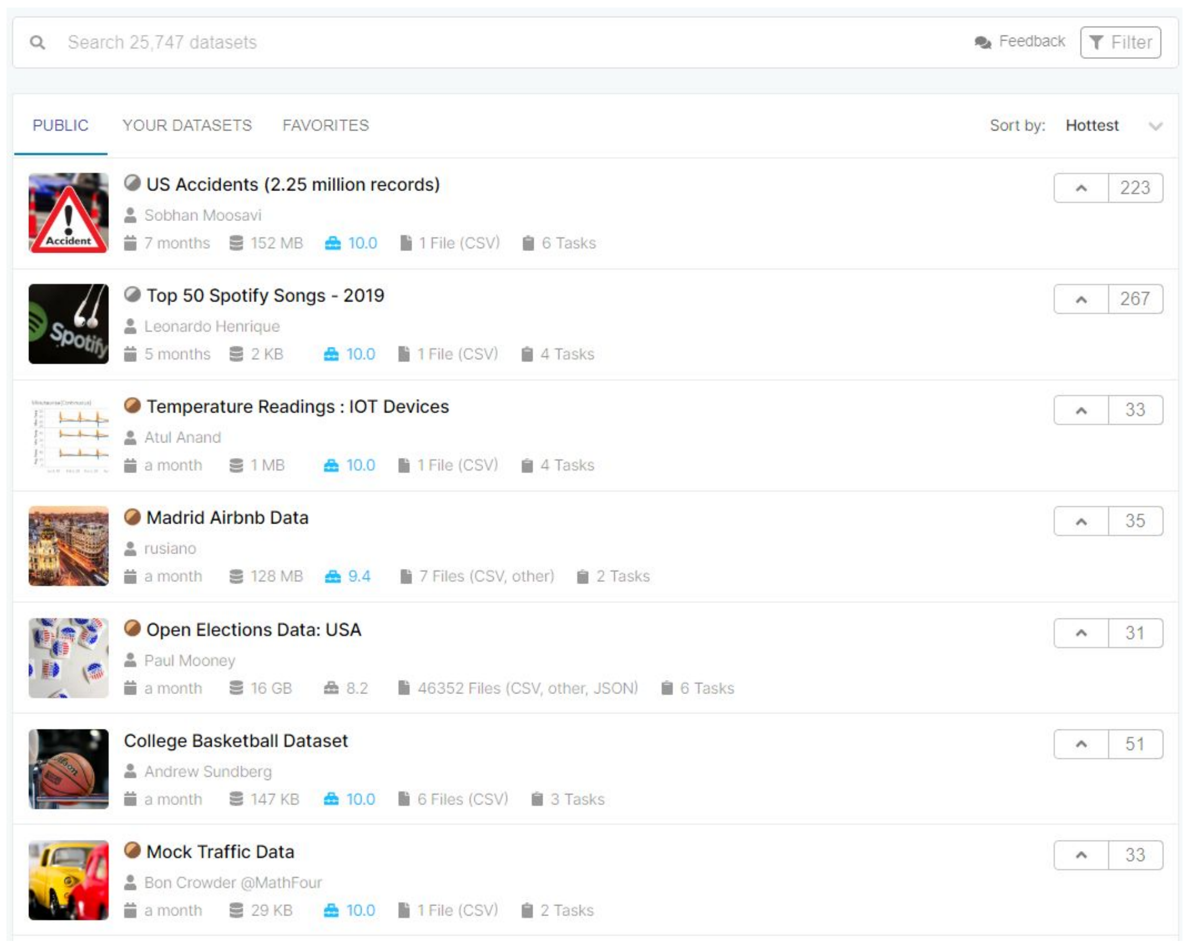
2. OPEN DATA PORTALS RESEARCH

2.1 First approach

"We will see our research procedure when looking for Open Data and which criteria we followed when discarding datasets."

When looking for Open Data we can use some useful tools and known portals. Kaggle[6], a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. It allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

We are interested in Kaggle datasets. There are huge amounts of them of different themes and they are evaluated by their usability. Currently, we can find more than 25k datasets available in the following format:



US Accidents (2.25 million records)
Sobhan Moosavi
7 months · 152 MB · 10.0 · 1 File (CSV) · 6 Tasks · 223

Top 50 Spotify Songs - 2019
Leonardo Henrique
5 months · 2 KB · 10.0 · 1 File (CSV) · 4 Tasks · 267

Temperature Readings : IOT Devices
Atul Anand
a month · 1 MB · 10.0 · 1 File (CSV) · 4 Tasks · 33

Madrid Airbnb Data
rusiano
a month · 128 MB · 9.4 · 7 Files (CSV, other) · 2 Tasks · 35

Open Elections Data: USA
Paul Mooney
a month · 16 GB · 8.2 · 46352 Files (CSV, other, JSON) · 6 Tasks · 31

College Basketball Dataset
Andrew Sundberg
a month · 147 KB · 10.0 · 6 Files (CSV) · 3 Tasks · 51

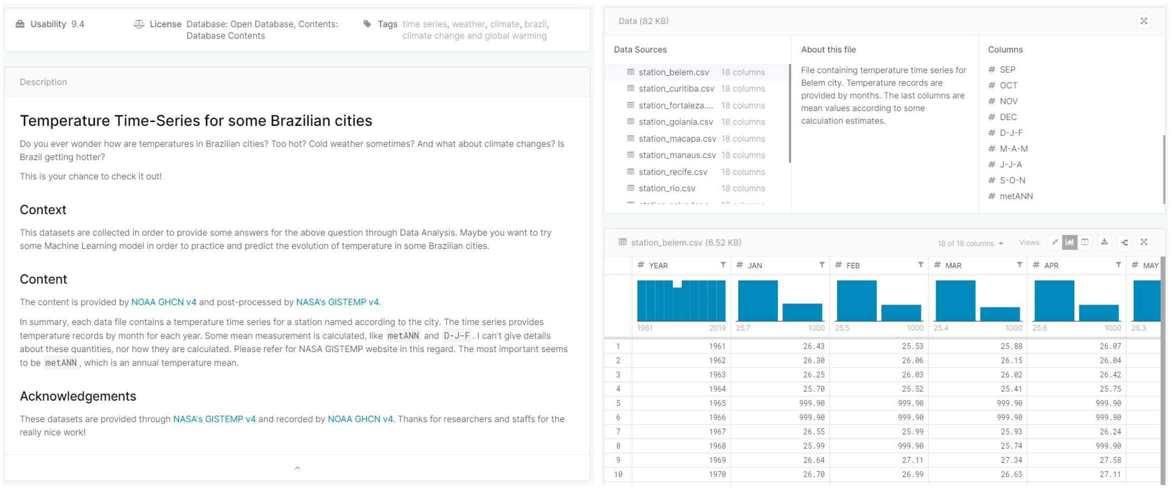
Mock Traffic Data
Bon Crowder @MathFour
a month · 29 KB · 10.0 · 1 File (CSV) · 2 Tasks · 33

Fig 11: Data Catalog from Kaggle

[6] Open Datasets and Machine Learning Projects. Retrieved from https://www.kaggle.com/datasets

It has a lot of good things, we can preview each file containing the data, we can see the distribution of each column in our dataset, a little explanation of itself, etc. It provides us good tools for data exploration and we can easily understand how it works.

The bad thing about Kaggle datasets is that they are not subject to revisions and periodic updates. This is because in the end they are uploaded by users who find it and, usually but not always, clean this data and update it. Some users comment on the dataset information where they find it, so we could track it back to the source and check if we could periodically download from there.


Usability 9.4 License Database: Open Database, Contents: Database Contents Tags time series, weather, climate, brazil, climate change and global warming

Description

Temperature Time-Series for some Brazilian cities

Do you ever wonder how are temperatures in Brazilian cities? Too hot? Cold weather sometimes? And what about climate changes? Is Brazil getting hotter?

This is your chance to check it out!

Context

This datasets are collected in order to provide some answers for the above question through Data Analysis. Maybe you want to try some Machine Learning model in order to practice and predict the evolution of temperature in some Brazilian cities.

Content

The content is provided by NOAA GHCN v4 and post-processed by NASA's GISTEMP v4.

In summary, each data file contains a temperature time series for a station named according to the city. The time series provides temperature records by month for each year. Some mean measurement is calculated, like `metANN` and `D-J-F`. I can't give details about these quantities, nor how they are calculated. Please refer for NASA GISTEMP website in this regard. The most important seems to be `metANN`, which is an annual temperature mean.

Acknowledgements

These datasets are provided through NASA's GISTEMP v4 and recorded by NOAA GHCN v4. Thanks for researchers and staffs for the really nice work!

Data (82 KB)

Data Sources	
station_belem.csv	18 columns
station_curitiba.csv	18 columns
station_fortaleza....	18 columns
station_goiania.csv	18 columns
station_macapa.csv	18 columns
station_manaus.csv	18 columns
station_recife.csv	18 columns
station_rio.csv	18 columns

About this file

File containing temperature time series for Belem city. Temperature records are provided by months. The last columns are mean values according to some calculation estimates.

Columns

- # SEP
- # OCT
- # NOV
- # DEC
- # D-J-F
- # M-A-M
- # J-J-A
- # S-O-N
- # metANN

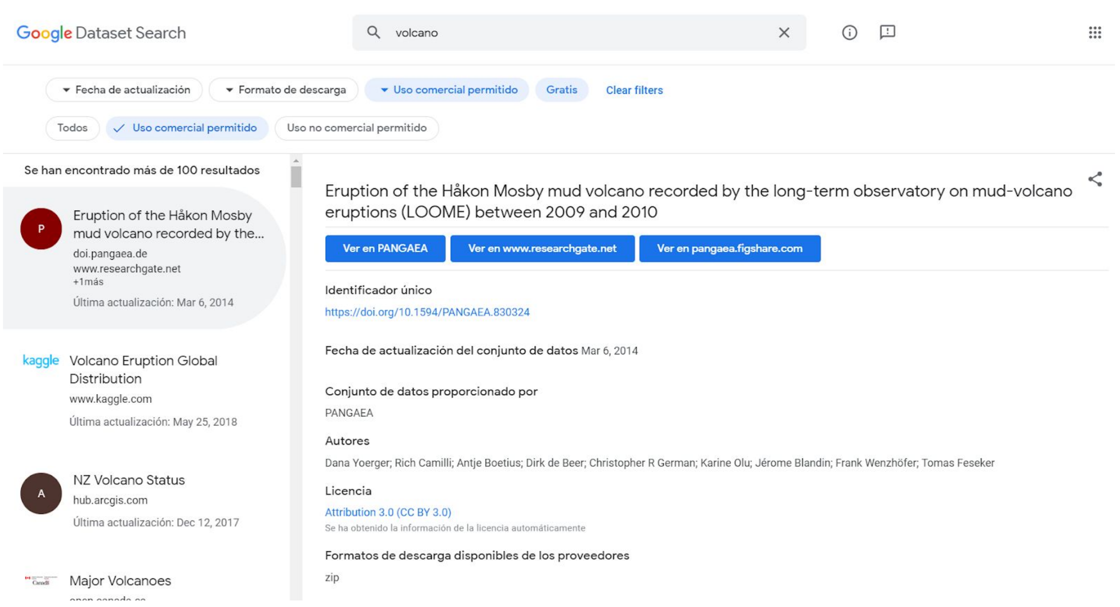
station_belem.csv (6.52 KB) 18 of 18 columns Views

	# YEAR	# JAN	# FEB	# MAR	# APR	# MAY
	1961 – 2019	25.7 – 1000	25.5 – 1000	25.4 – 1000	25.8 – 1000	26.3
1	1961	26.43	25.53	25.88	26.07	
2	1962	26.30	26.06	26.15	26.04	
3	1963	26.25	26.03	26.02	26.42	
4	1964	25.70	25.52	25.41	25.75	
5	1965	999.90	999.90	999.90	999.90	
6	1966	999.90	999.90	999.90	999.90	
7	1967	26.55	25.99	25.93	26.24	
8	1968	25.99	999.90	25.74	999.90	
9	1969	26.64	27.11	27.34	27.58	
10	1970	26.70	26.99	26.63	27.11	

Fig 12: Description and Data exploration of a Kaggle Dataset.

Another useful tool released at the end of 2019 is Google Datasets[7]. It works like the images searcher but for datasets. It is still very new and it is subject to several changes but the new come tool from Google is probably going to be the main reference when searching for Open Data. It also has a few filters that can be useful like the format we want it, if it's free, if we can use it for commercial purposes or when was it updated for the last time.

[7] Google Dataset Search. Retrieved from https://datasetsearch.research.google.com/


Google Dataset Search

volcano

Fecha de actualización | Formato de descarga | Uso comercial permitido | Gratis | Clear filters

Todos | ✓ Uso comercial permitido | Uso no comercial permitido

Se han encontrado más de 100 resultados

Eruption of the Håkon Mosby mud volcano recorded by the...
doi.pangaea.de
www.researchgate.net
+1más
Última actualización: Mar 6, 2014

Volcano Eruption Global Distribution
www.kaggle.com
Última actualización: May 25, 2018

NZ Volcano Status
hub.arcgis.com
Última actualización: Dec 12, 2017

Major Volcanoes

Eruption of the Håkon Mosby mud volcano recorded by the long-term observatory on mud-volcano eruptions (LOOME) between 2009 and 2010

Ver en PANGAEA | Ver en www.researchgate.net | Ver en pangaea.figshare.com

Identificador único
https://doi.org/10.1594/PANGAEA.830324

Fecha de actualización del conjunto de datos Mar 6, 2014

Conjunto de datos proporcionado por
PANGAEA

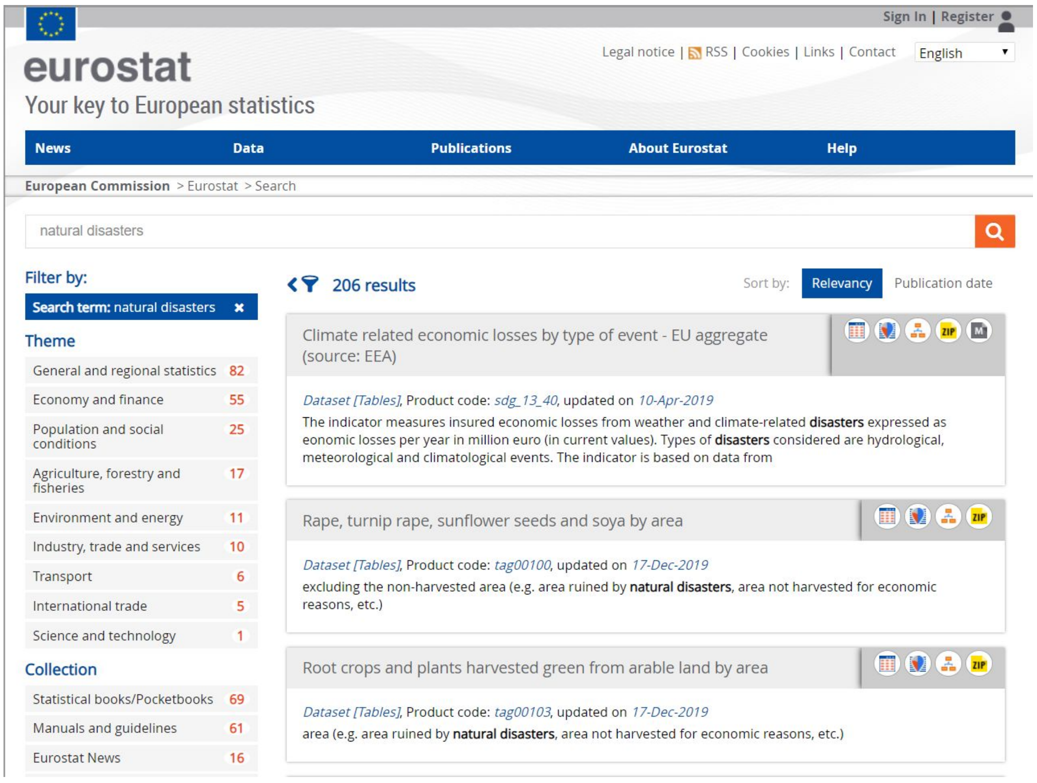
Autores
Dana Yoerger; Rich Camilli; Antje Boetius; Dirk de Beer; Christopher R German; Karine Olu; Jérome Blandin; Frank Wenzhöfer; Tomas Feseker

Licencia
Attribution 3.0 (CC BY 3.0)
Se ha obtenido la información de la licencia automáticamente

Formatos de descarga disponibles de los proveedores
zip

Fig 13: Example of a search in Google Datasets

Another option is to search for open data portals or websites that provide this information. We found some interesting options in our first approach. Eurostat[8] has tons of datasets from a lot of themes. We will look for those related to natural disasters.


Sign In | Register

Legal notice | RSS | Cookies | Links | Contact | English

eurostat
Your key to European statistics

News | Data | Publications | About Eurostat | Help

European Commission > Eurostat > Search

natural disasters

Filter by:
Search term: natural disasters ✖

Theme

General and regional statistics	82
Economy and finance	55
Population and social conditions	25
Agriculture, forestry and fisheries	17
Environment and energy	11
Industry, trade and services	10
Transport	6
International trade	5
Science and technology	1

Collection

Statistical books/Pocketbooks	69
Manuals and guidelines	61
Eurostat News	16

206 results

Sort by: Relevancy | Publication date

Climate related economic losses by type of event - EU aggregate (source: EEA)
Dataset [Tables], Product code: *sdg_13_40*, updated on *10-Apr-2019*
The indicator measures insured economic losses from weather and climate-related **disasters** expressed as eonomic losses per year in million euro (in current values). Types of **disasters** considered are hydrological, meteorological and climatological events. The indicator is based on data from

Rape, turnip rape, sunflower seeds and soya by area
Dataset [Tables], Product code: *tag00100*, updated on *17-Dec-2019*
excluding the non-harvested area (e.g. area ruined by **natural disasters**, area not harvested for economic reasons, etc.)

Root crops and plants harvested green from arable land by area
Dataset [Tables], Product code: *tag00103*, updated on *17-Dec-2019*
area (e.g. area ruined by **natural disasters**, area not harvested for economic reasons, etc.)

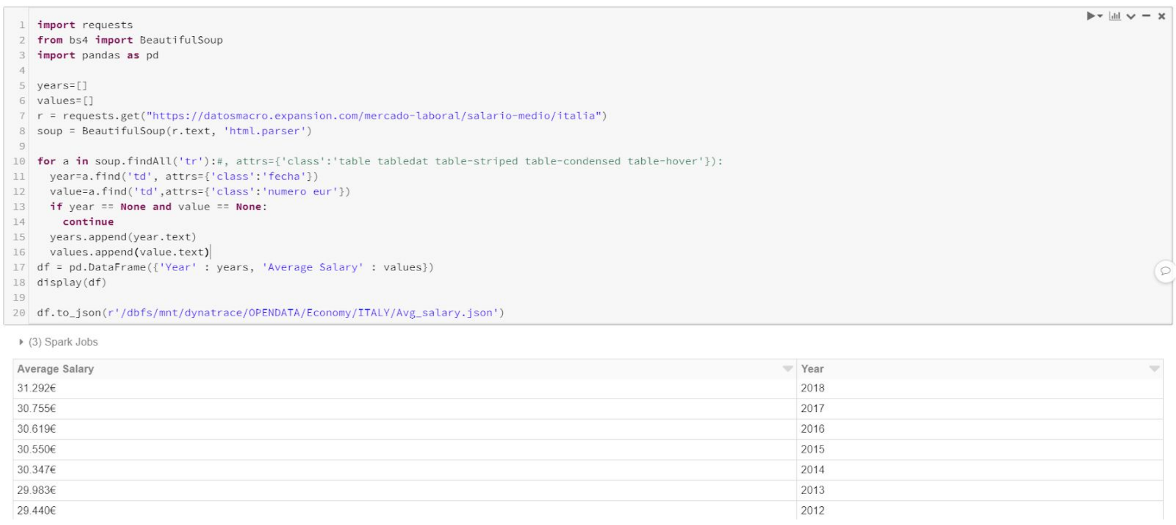
Fig 14: Eurostat search example for natural disasters

[8] Your key to European statistics. Retrieved from https://ec.europa.eu/eurostat.

Checking out the data and the formats we identified some potential issues. This data cannot be used in this project because most of them are only for Europe and Zurich has clients worldwide. The company told us that we must also cover at least the USA where natural disasters are a real problem. We kept looking and then found some other interesting Open Data portals.

We also found Europa Press portal[9], which claims to have more than 57M datasets, but unfortunately, they show almost no results when we search for climate data and found that most of them are only related to Spain.

Datosmacro shows data at a worldwide level and they are supposed to have a lot of information but they only provide a few tables on their web page comparing very few rows of their information. In this case, we started a process of web scraping to retrieve this data.



```python
import requests
from bs4 import BeautifulSoup
import pandas as pd

years=[]
values=[]
r = requests.get("https://datosmacro.expansion.com/mercado-laboral/salario-medio/italia")
soup = BeautifulSoup(r.text, 'html.parser')

for a in soup.findAll('tr'):#, attrs={'class':'table tabledat table-striped table-condensed table-hover'}):
  year=a.find('td', attrs={'class':'fecha'})
  value=a.find('td',attrs={'class':'numero eur'})
  if year == None and value == None:
    continue
  years.append(year.text)
  values.append(value.text)
df = pd.DataFrame({'Year' : years, 'Average Salary' : values})
display(df)

df.to_json(r'/dbfs/mnt/dynatrace/OPENDATA/Economy/ITALY/Avg_salary.json')
```

▸ (3) Spark Jobs

Average Salary	Year
31.292€	2018
30.755€	2017
30.619€	2016
30.550€	2015
30.347€	2014
29.983€	2013
29.440€	2012

Fig 15: Successful web scraping code and output

We started with a simple table before trying to go further and we succeeded. Using libraries like requests to get to the link, BeautifulSoup to navigate through the HTML of the web and pandas to save the data in JSON format we end up with the table on the web in our data lake. We noticed that using web scraping could put us into trouble in some different ways.

First of all, our data is expected to be retrieved with some frequency, so we must be sure that the source where we get it has the updated data every time we visit it. Even if this was the case, which was not since Datosmacro follows a blog structure and new data is published as a new post in another link so we cannot retrieve new information from the same link. Also, if we apply this technique we are subjective to fail if the web page changes its structure. This

[9] La actualidad informativa, en datos. Retrieved from https://www.epdata.es/.

kind of change is not unusual since nowadays is very easy to change the visuals with no need to code.

Another problem is that web scraping is what could be described as *“alegal”*, it means that it is not strictly forbidden by law but depending on how it is done it could be interpreted as illegal. All related to extracting information that was not given to us directly could be seen as stealing information. We must consider all of this when extracting data and be sure that everything is Open Data with all his characteristics. We will try to use web scraping if and only if it is absolutely necessary and ensuring that we are retrieving Open Data.

2.2 Going to the source

"When going to the source of the data we find that data can be represented in very different ways. We must transform this data into the same format. This is the toll to pay when we want periodicity and reliability."

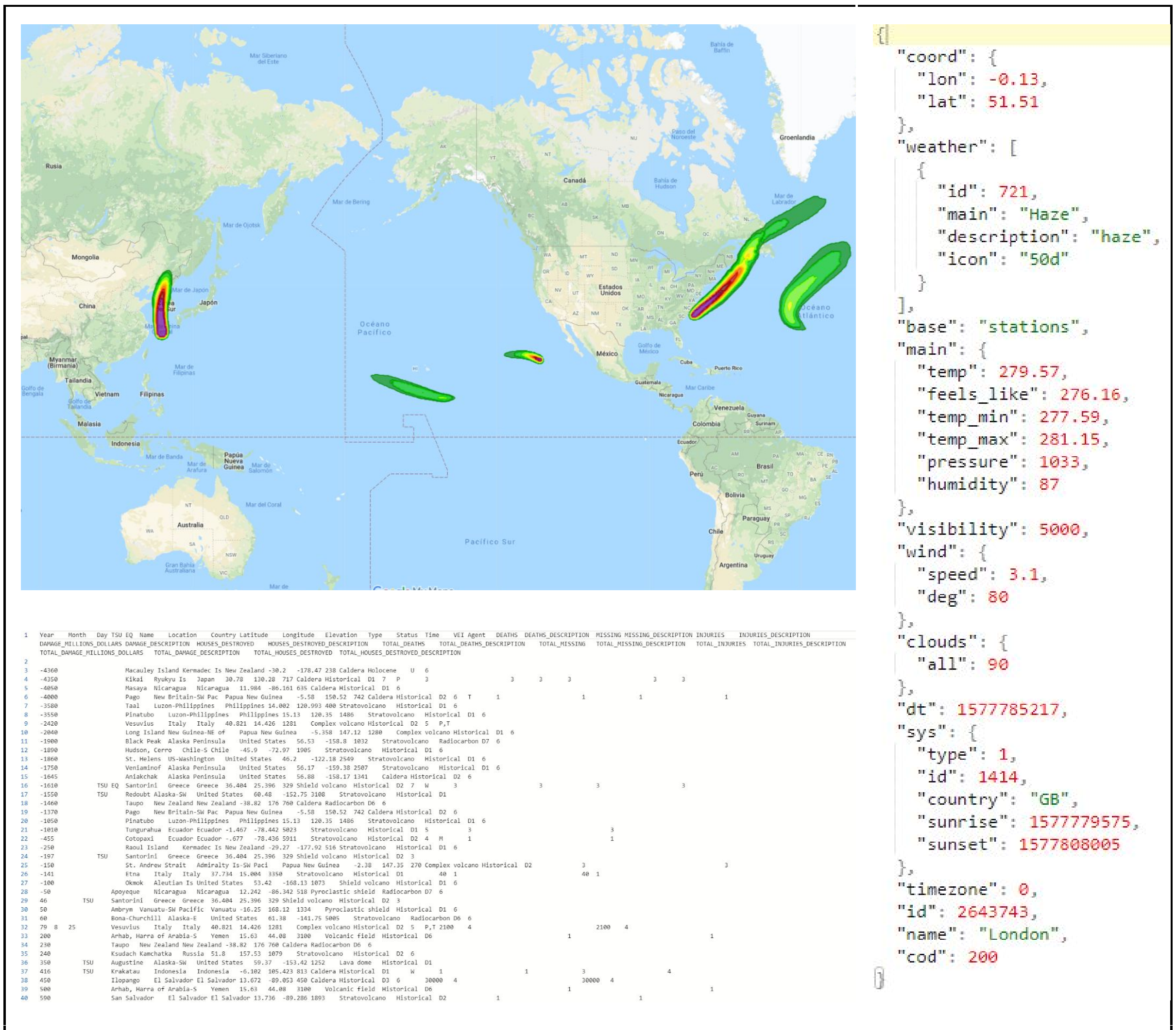
After the first approach, we decided that the best idea is going directly to the source. Using the tools described before we can find where to extract exactly the raw data. We can also assume that if the data that we are looking for is in Kaggle or passes the Google Datasets search filters it is Open Data.

We want all data related to natural disasters, so some of the datasets that we are looking for are: Hurricanes, Earthquakes, Wildfires, Volcano eruptions, Weather Forecast, etc. The main distributors of all this data are usually governments or associations like NASA that claim that the data that they give are Open Data.

Another point that we didn't mention but it is also important when referring to Open Data is the reliableness of the data. We want to be sure that what we get is correct in order to make good decisions in the end. That's why we must dedicate some extra time to this part to ensure that our sources are trustful.

After some research, we end up with good quality sources like the National Oceanic and Atmospheric Administration (NOAA), NASA, United States Geological Survey (USGS), and open weather API. All of these sources are considered trustable enough for our purposes.

Once we found our sources, we manually downloaded the data. That is another problem with Open Data, each dataset is in a different format and has a different structure. We find data that is in CSV, JSON, KML, etc. Each format has a specific purpose, some of them like KML are created to be visualized in some specific viewers like MyMaps from Google, others are more generic like XML or JSON. We must define a general pattern for the data.



```
Year	Month	Day	TSU	EQ	Name	Location	Country	Latitude	Longitude	Elevation	Type	Status	Time	VEI	Agent	DEATHS	DEATHS_DESCRIPTION	MISSING	MISSING_DESCRIPTION	INJURIES	INJURIES_DESCRIPTION
DAMAGE_MILLIONS_DOLLARS	DAMAGE_DESCRIPTION	HOUSES_DESTROYED	HOUSES_DESTROYED_DESCRIPTION	TOTAL_DEATHS	TOTAL_DEATHS_DESCRIPTION	TOTAL_MISSING	TOTAL_MISSING_DESCRIPTION	TOTAL_INJURIES	TOTAL_INJURIES_DESCRIPTION
TOTAL_DAMAGE_MILLIONS_DOLLARS	TOTAL_DAMAGE_DESCRIPTION	TOTAL_HOUSES_DESTROYED	TOTAL_HOUSES_DESTROYED_DESCRIPTION

-4360				Macauley Island	Kermadec Is	New Zealand	-30.2	-178.47	238	Caldera	Holocene		U	6
-4350				Kikai	Ryukyu Is	Japan	30.78	130.28	717	Caldera	Historical	D1	7	P	3			3	3	3			3	3
-4050				Masaya	Nicaragua	Nicaragua	11.984	-86.161	635	Caldera	Historical	D1	6
-4000				Pago	New Britain-SW Pac	Papua New Guinea	-5.58	150.52	742	Caldera	Historical	D2	6	T	1			1		1			1
-3580				Taal	Luzon-Philippines	Philippines	14.002	120.993	400	Stratovolcano	Historical	D1	6
-3550				Pinatubo	Luzon-Philippines	Philippines	15.13	120.35	1486	Stratovolcano	Historical	D1	6
-2420				Vesuvius	Italy	Italy	40.821	14.426	1281	Complex volcano	Historical	D2	5	P,T
-2040				Long Island	New Guinea-NE of	Papua New Guinea	-5.358	147.12	1280	Complex volcano	Historical	D1	6
-1900				Black Peak	Alaska Peninsula	United States	56.53	-158.8	1032	Stratovolcano	Radiocarbon	D7	6
-1890				Hudson, Cerro	Chile-S	Chile	-45.9	-72.97	1905	Stratovolcano	Historical	D1	6
-1860				St. Helens	US-Washington	United States	46.2	-122.18	2549	Stratovolcano	Historical	D1	6
-1750				Veniaminof	Alaska Peninsula	United States	56.17	-159.38	2507	Stratovolcano	Historical	D1	6
-1645				Aniakchak	Alaska Peninsula	United States	56.88	-158.17	1341	Caldera	Historical	D2	6
-1610			TSU	EQ	Santorini	Greece	Greece	36.404	25.396	329	Shield volcano	Historical	D2	7	W	3			3		3			3
-1550			TSU		Redoubt	Alaska-SW	United States	60.48	-152.75	3108	Stratovolcano	Historical	D1
-1460				Taupo	New Zealand	New Zealand	-38.82	176	760	Caldera	Radiocarbon	D6	6
-1370				Pago	New Britain-SW Pac	Papua New Guinea	-5.58	150.52	742	Caldera	Historical	D2	6
-1050				Pinatubo	Luzon-Philippines	Philippines	15.13	120.35	1486	Stratovolcano	Historical	D1	6
-1010				Tungurahua	Ecuador	Ecuador	-1.467	-78.442	5023	Stratovolcano	Historical	D1	5		3				3
-455				Cotopaxi	Ecuador	Ecuador	-.677	-78.436	5911	Stratovolcano	Historical	D2	4	M	1				1
-250				Raoul Island	Kermadec Is	New Zealand	-29.27	-177.92	516	Stratovolcano	Historical	D1	6
-197			TSU		Santorini	Greece	Greece	36.404	25.396	329	Shield volcano	Historical	D2	3
-150				St. Andrew Strait	Admiralty Is-SW Paci	Papua New Guinea	-2.38	147.35	270	Complex volcano	Historical	D2			3				3
-141				Etna	Italy	Italy	37.734	15.004	3350	Stratovolcano	Historical	D1		40	1			40	1
-100				Okmok	Aleutian Is	United States	53.42	-168.13	1073	Shield volcano	Historical	D1	6
-50				Apoyeque	Nicaragua	Nicaragua	12.242	-86.342	518	Pyroclastic shield	Radiocarbon	D7	6
46			TSU		Santorini	Greece	Greece	36.404	25.396	329	Shield volcano	Historical	D2	3
50				Ambrym	Vanuatu-SW Pacific	Vanuatu	-16.25	168.12	1334	Pyroclastic shield	Historical	D1	6
60				Bona-Churchill	Alaska-E	United States	61.38	-141.75	5005	Stratovolcano	Radiocarbon	D6	6
79	8	25			Vesuvius	Italy	Italy	40.821	14.426	1281	Complex volcano	Historical	D2	5	P,T	2100	4			2100	4
200				Arhab, Harra of	Arabia-S	Yemen	15.63	44.08	3100	Volcanic field	Historical	D6		1				1
230				Taupo	New Zealand	New Zealand	-38.82	176	760	Caldera	Radiocarbon	D6	6
240				Ksudach	Kamchatka	Russia	51.8	157.53	1079	Stratovolcano	Historical	D2	6
350			TSU		Augustine	Alaska-SW	United States	59.37	-153.42	1252	Lava dome	Historical	D1
416			TSU		Krakatau	Indonesia	Indonesia	-6.102	105.423	813	Caldera	Historical	D1		W	1		1		3		4
450				Ilopango	El Salvador	El Salvador	13.672	-89.053	450	Caldera	Historical	D3	6		30000	4			30000	4
500				Arhab, Harra of	Arabia-S	Yemen	15.63	44.08	3100	Volcanic field	Historical	D6		1				1
590				San Salvador	El Salvador	El Salvador	13.736	-89.286	1893	Stratovolcano	Historical	D2		1			1
```

```json
{
  "coord": {
    "lon": -0.13,
    "lat": 51.51
  },
  "weather": [
    {
      "id": 721,
      "main": "Haze",
      "description": "haze",
      "icon": "50d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 279.57,
    "feels_like": 276.16,
    "temp_min": 277.59,
    "temp_max": 281.15,
    "pressure": 1033,
    "humidity": 87
  },
  "visibility": 5000,
  "wind": {
    "speed": 3.1,
    "deg": 80
  },
  "clouds": {
    "all": 90
  },
  "dt": 1577785217,
  "sys": {
    "type": 1,
    "id": 1414,
    "country": "GB",
    "sunrise": 1577779575,
    "sunset": 1577808005
  },
  "timezone": 0,
  "id": 2643743,
  "name": "London",
  "cod": 200
}
```

Figure 16: Comparison between KML (Hurricanes) format, CSV (Volcanoes) format and nested JSON (Weather) format.

3. EXTRACT-TRANSFORM-LOAD (ETL)

"In computing, extract, transform, load (ETL) is the general procedure of copying data from one or more sources into a destination system that represents the data differently from the source(s) or in a different context than the source(s)."

ETL is an important part of the ecosystem of Data Science and Business Analytics. We will focus on this part of this project. We can define extract as the process of reading data from a database. In this stage, the data is collected, in our case from multiple and different types of sources. Transform is the process of converting the extracted data from its previous form into the form it needs to be so that it can be placed into another database. Finally, the load is the process of writing the data into the target database, in our case our ADLS.

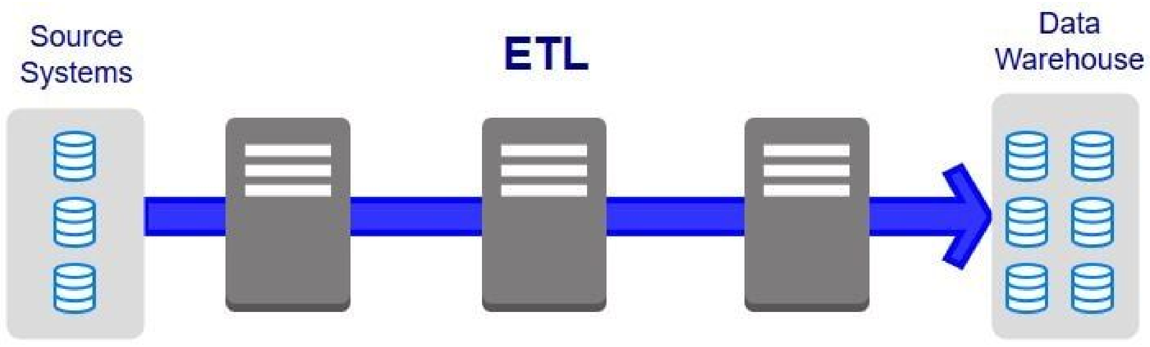


Fig 17: Schema of the ETL process

3.1 Prerequisites using Databricks

As we explained before Databricks is like a Python Notebook with some extra functionalities. We will follow some made-up standards in order to set a pattern when coding.

- Each source will be extracted and treated first individually in a different notebook.
- Every notebook will use the same cluster.
- The code must be optimal but also readable.
- The first cell will be called Main Parameters, there we will place all the imports and functions that will be used along the process.
- The data must be saved in any pandas exportable format (JSON, CSV, etc).
- All tables must have the Longitude and Latitude column
- All data must be transformed to table format, no nested relations are allowed.
- The job must return a Succeeded if all the data was retrieved or Failed if no data were extracted.

3.2 Using Open Weather API

"The term API refers to a specific kind of interface between a client and a server, which has been described as a "contract" between both - such that if the client makes a request in a specific format, it will always get a response in a specific format or initiate a defined action."

After trying some other options, we ended up with Open Weather API as the way to retrieve our weather forecast information. It has the biggest dataset of cities and it is updated every day. It also contained a lot of information that other services do not provide, like the speed of the wind, pressure, humidity, etc.

The first thing to do is to download the dataset of cities. We had more than 200K cities with its id, name, country, longitude, and latitude. We will need this information since we must call the API using the exact names from this dataset, otherwise, we will receive an error. Instead of downloading all the information for all the cities, which would generate too many gigabytes per day, we will do another approach.

We will always download the information about certain cities, the most important ones in Europe and the USA. Then we will also download the information of 5 random cities per country to have a representation of each country. The schema is the following.

Step	Task
1	Get our historical dataset
2	Retrieve the information for important cities
3	Select 5 random cities per country
4	Retrieve information of the random cities
5	Append this new information to the historical dataset
6	Save this data as the new historic

Expected Result	Dataset of cities and timestamps with the weather forecast information. Some cities will be repeated, like the important ones. This dataset can be later ordered by the city or timestamp. It will increase its size every day it is executed.

3.2.1 Implementation

"In this section, we will see the implementation of the Weather ETL process."

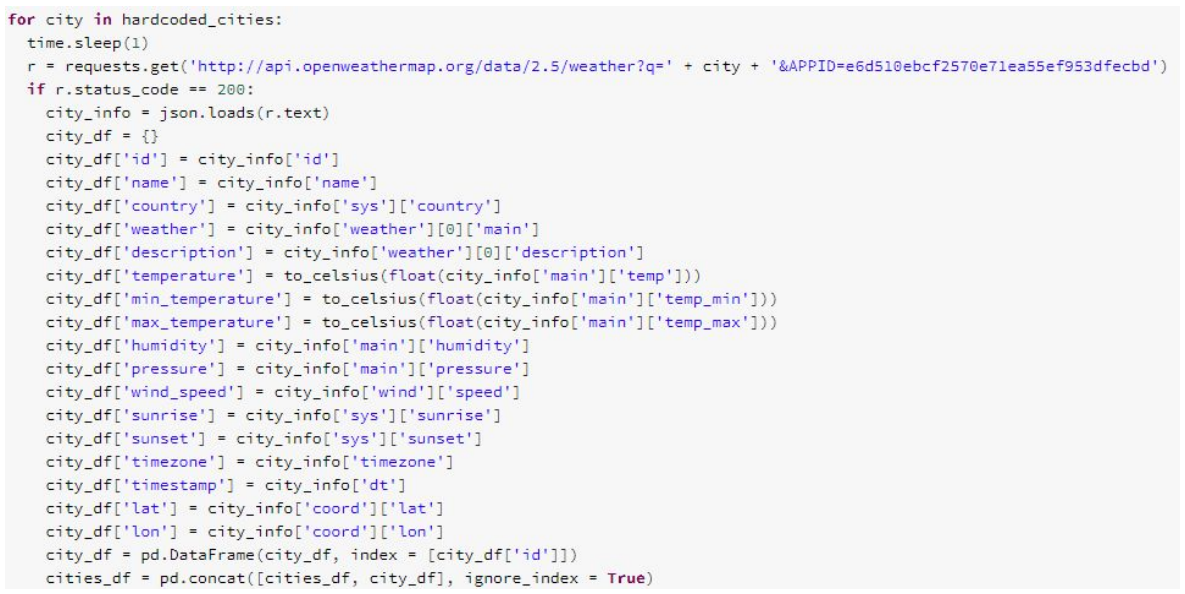
First of all, we have created a file in our ADLS where to store all the data. We have created a master folder called OPENDATA, then a subfolder called Natural Disasters. Here we will place all the data that we will retrieve related to nature-related events. Inside this folder we created a new folder called Weather so we know exactly what's inside. Finally, a folder called Global, since we want to store data form cities all around the world, not only a concrete country. We will follow this schema for every source of data.

The next step is to create a dummy empty historic to start. We've saved the cities list in the ADLS. We have been searching for some important information about the Open Weather API, so now we know the format of the response and that no more than 3600 requests per hour are allowed.

Request	**Response**
requests.get('http://api.openweathermap.org/data/2.5/weather?q=london&APPID=e6d510ebcf2570e71ea55ef953dfecbd')	`{"coord":{"lon":-0.13,"lat":51.51},"weather":[{"id":804,"main":"Clouds","description":"overcast clouds","icon":"04d"}],"base":"stations","main":{"temp":282.55,"feels_like":278.83,"temp_min":281.15,"temp_max":283.71,"pressure":1020,"humidity":81},"visibility":10000,"wind":{"speed":4.1,"deg":200},"clouds":{"all":90},"dt":1577969000,"sys":{"type":1,"id":1414,"country":"GB","sunrise":1577952366,"sunset":1577980925},"timezone":0,"id":2643743,"name":"London","cod":200}`

Knowing the structure of the response, we can now select the information that we want to keep. Some cities give more parameters than others, so we must be sure to claim only those parameters that are in every city in order to have a consistent dataset. We will work first with the hardcoded list of cities in the same format that is in the cities database. Our list includes the most important cities in Europe and the USA. We read that list from an Excel file that other teams have available, so they can add any city at any moment.

For each one of these cities, we make a request to the API. We need to hold each operation at least one second to be sure that we are not trespassing the maximum requests per hour. The idea now is to not request more than 3600 cities and to execute this code only once per day but we don't know if this is subject to changes so we will place the wait command just in case. The strategy here is to have one dictionary object, the JSON with the request and an empty dictionary that will be filled. We will extract only the info that we want from one dictionary to another. This dictionary is transformed into a dataframe and append to our historic.



```
for city in hardcoded_cities:
  time.sleep(1)
  r = requests.get('http://api.openweathermap.org/data/2.5/weather?q=' + city + '&APPID=e6d510ebcf2570e71ea55ef953dfecbd')
  if r.status_code == 200:
    city_info = json.loads(r.text)
    city_df = {}
    city_df['id'] = city_info['id']
    city_df['name'] = city_info['name']
    city_df['country'] = city_info['sys']['country']
    city_df['weather'] = city_info['weather'][0]['main']
    city_df['description'] = city_info['weather'][0]['description']
    city_df['temperature'] = to_celsius(float(city_info['main']['temp']))
    city_df['min_temperature'] = to_celsius(float(city_info['main']['temp_min']))
    city_df['max_temperature'] = to_celsius(float(city_info['main']['temp_max']))
    city_df['humidity'] = city_info['main']['humidity']
    city_df['pressure'] = city_info['main']['pressure']
    city_df['wind_speed'] = city_info['wind']['speed']
    city_df['sunrise'] = city_info['sys']['sunrise']
    city_df['sunset'] = city_info['sys']['sunset']
    city_df['timezone'] = city_info['timezone']
    city_df['timestamp'] = city_info['dt']
    city_df['lat'] = city_info['coord']['lat']
    city_df['lon'] = city_info['coord']['lon']
    city_df = pd.DataFrame(city_df, index = [city_df['id']])
    cities_df = pd.concat([cities_df, city_df], ignore_index = True)
```

Fig 18: Data extraction of the common information

The data extraction process is the same for randomized cities. What matters here is how we select 5 random cities from each country. To do this we have created a function called shuffle and reduce, the output of this function is the list of cities for the new request. We send the list of all cities minus the hardcoded ones in order to avoid duplicates. We also delete some special cases such as the world and the continents itself, we want only cities. This function receives the amount of elements that need to hold and the column. Then it is shuffled and then we only keep the first N elements.

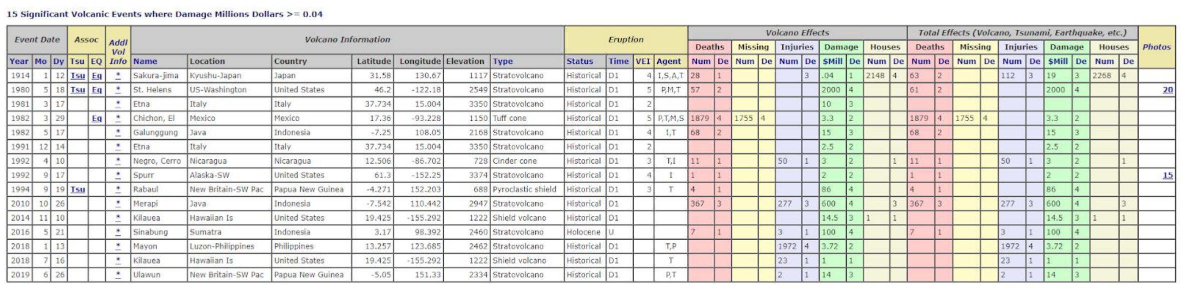
We repeat the data extraction process with this new list and append the results to our global historical dataset.

3.3 Volcano eruptions Dataset

“Volcanoes eruptions are another natural disaster that causes a lot of damage. We can think that nowadays they are less frequent, but the truth is that we don’t need huge explosions of magma and lava to cause damage.”

It was difficult to find a good dataset of volcanic eruptions since they are not so common and they are not tracked daily, like the weather forecast. The National Oceanic and Atmospheric Administration[10] (NOAA) had a pretty good history with a first record in 4360 BC.

The most interesting thing is that they also have some nice information that could help an insurance company like Zurich, such as the number of houses that were destroyed, the damage caused to the infrastructure in millions of dollars, the injuries, missings, and deaths. It also has the Longitude and Latitude of the volcano, so we will be able to relate it with other datasets in the future. Not only this but they also tracked the total effect of the volcano, it means that if the volcano also generated a tsunami or earthquake it is also measured.



15 Significant Volcanic Events where Damage Millions Dollars >= 0.04

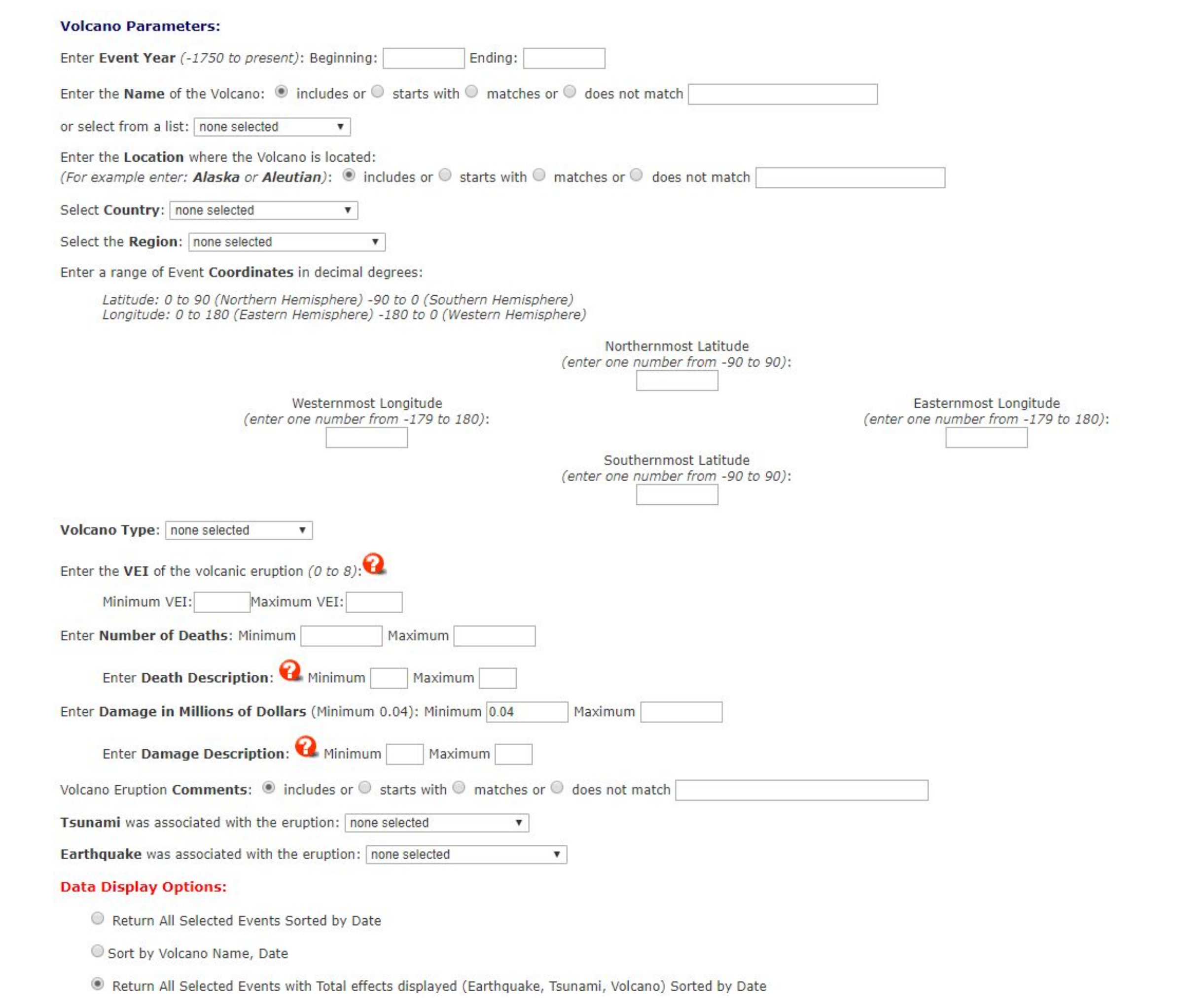
Event Date			Assoc		Addl Vol Info	Volcano Information							Eruption			
Year	Mo	Dy	Tsu	EQ		Name	Location	Country	Latitude	Longitude	Elevation	Type	Status	Time	VEI	Agent
1914	1	12	Tsu	Eq	*	Sakura-jima	Kyushu-Japan	Japan	31.58	130.67	1117	Stratovolcano	Historical	D1	4	I,S,A,T
1980	5	18	Tsu	Eq	*	St. Helens	US-Washington	United States	46.2	-122.18	2549	Stratovolcano	Historical	D1	5	P,M,T
1981	3	17			*	Etna	Italy	Italy	37.734	15.004	3350	Stratovolcano	Historical	D1	2	
1982	3	29		Eq	*	Chichon, El	Mexico	Mexico	17.36	-93.228	1150	Tuff cone	Historical	D1	5	P,T,M,S
1982	5	17			*	Galunggung	Java	Indonesia	-7.25	108.05	2168	Stratovolcano	Historical	D1	4	I,T
1991	12	14			*	Etna	Italy	Italy	37.734	15.004	3350	Stratovolcano	Historical	D1	2	
1992	4	10			*	Negro, Cerro	Nicaragua	Nicaragua	12.506	-86.702	728	Cinder cone	Historical	D1	3	T,I
1992	9	17			*	Spurr	Alaska-SW	United States	61.3	-152.25	3374	Stratovolcano	Historical	D1	4	I
1994	9	19	Tsu		*	Rabaul	New Britain-SW Pac	Papua New Guinea	-4.271	152.203	688	Pyroclastic shield	Historical	D1	3	T
2010	10	26			*	Merapi	Java	Indonesia	-7.542	110.442	2947	Stratovolcano	Historical	D1		
2014	11	10			*	Kilauea	Hawaiian Is	United States	19.425	-155.292	1222	Shield volcano	Historical	D1		
2016	5	21			*	Sinabung	Sumatra	Indonesia	3.17	98.392	2460	Stratovolcano	Holocene	U		
2018	1	13			*	Mayon	Luzon-Philippines	Philippines	13.257	123.685	2462	Stratovolcano	Historical	D1		T,P
2018	7	16			*	Kilauea	Hawaiian Is	United States	19.425	-155.292	1222	Shield volcano	Historical	D1		T
2019	6	26			*	Ulawun	New Britain-SW Pac	Papua New Guinea	-5.05	151.33	2334	Stratovolcano	Historical	D1		P,T

		Volcano Effects										Total Effects (Volcano, Tsunami, Earthquake, etc.)										Photos
		Deaths		Missing		Injuries		Damage		Houses		Deaths		Missing		Injuries		Damage		Houses		
Year	Name	Num	De	Num	De	Num	De	$Mill	De	Num	De	Num	De	Num	De	Num	De	$Mill	De	Num	De	
1914	Sakura-jima	28	1				3	.04	1	2148	4	63	2			112	3	19	3	2268	4	
1980	St. Helens	57	2					2000	4			61	2					2000	4			20
1981	Etna							10	3													
1982	Chichon, El	1879	4	1755	4			3.3	2			1879	4	1755	4			3.3	2			
1982	Galunggung	68	2					15	3			68	2					15	3			
1991	Etna							2.5	2									2.5	2			
1992	Negro, Cerro	11	1			50	1	3	2		1	11	1			50	1	3	2		1	
1992	Spurr	1	1					2	2			1	1					2	2			15
1994	Rabaul	4	1					86	4			4	1					86	4			
2010	Merapi	367	3			277	3	600	4		3	367	3			277	3	600	4		3	
2014	Kilauea							14.5	3	1	1							14.5	3	1	1	
2016	Sinabung	7	1			3	1	100	4			7	1			3	1	100	4			
2018	Mayon					1972	4	3.72	2							1972	4	3.72	2			
2018	Kilauea					23	1	1	1							23	1	1	1			
2019	Ulawun					2	1	14	3							2	1	14	3			

Fig 19: Volcanoes Dataset filter by Damage

As we can see in the Figure 19, this dataset contains a lot of information. Every column is an important parameter of the volcano, and we got more than 30.The only problem with this is that the way to get to the data is through filling a form and then sending it. It looks like we will need to do some web scraping. We will try to be the least restrictive possible, so we download all the dataset and then if we need to filter we can do it from our data in our ADLS.

[10] NOAA. (2012, February 10). National Centers for Environmental Information (NCEI). Retrieved from https://www.ngdc.noaa.gov/.


Volcano Parameters:

Enter **Event Year** *(-1750 to present)*: Beginning: ____ Ending: ____

Enter the **Name** of the Volcano: [✓ includes or] starts with matches or does not match ____

or select from a list: none selected

Enter the **Location** where the Volcano is located:
*(For example enter: **Alaska** or **Aleutian**)*: [✓ includes or] starts with matches or does not match ____

Select **Country**: none selected

Select the **Region**: none selected

Enter a range of Event **Coordinates** in decimal degrees:

Latitude: 0 to 90 (Northern Hemisphere) -90 to 0 (Southern Hemisphere)
Longitude: 0 to 180 (Eastern Hemisphere) -180 to 0 (Western Hemisphere)

Northernmost Latitude
(enter one number from -90 to 90):

Westernmost Longitude
(enter one number from -179 to 180):

Easternmost Longitude
(enter one number from -179 to 180):

Southernmost Latitude
(enter one number from -90 to 90):

Volcano Type: none selected

Enter the **VEI** of the volcanic eruption *(0 to 8)*:

Minimum VEI: ____ Maximum VEI: ____

Enter **Number of Deaths**: Minimum ____ Maximum ____

Enter **Death Description**: Minimum ____ Maximum ____

Enter **Damage in Millions of Dollars** (Minimum 0.04): Minimum 0.04 Maximum ____

Enter **Damage Description**: Minimum ____ Maximum ____

Volcano Eruption **Comments**: [✓ includes or] starts with matches or does not match ____

Tsunami was associated with the eruption: none selected

Earthquake was associated with the eruption: none selected

Data Display Options:

Return All Selected Events Sorted by Date

Sort by Volcano Name, Date

[✓ Return All Selected Events with Total effects displayed (Earthquake, Tsunami, Volcano) Sorted by Date]

Fig 20: Volcanoes Form

A good thing about this dataset is that when it is updated they add the new data to the historic. This means that we only need to check it every day and replace the old data with the new one instead of keeping a copy of what exists and then append the new information. The procedure will be much shorter.

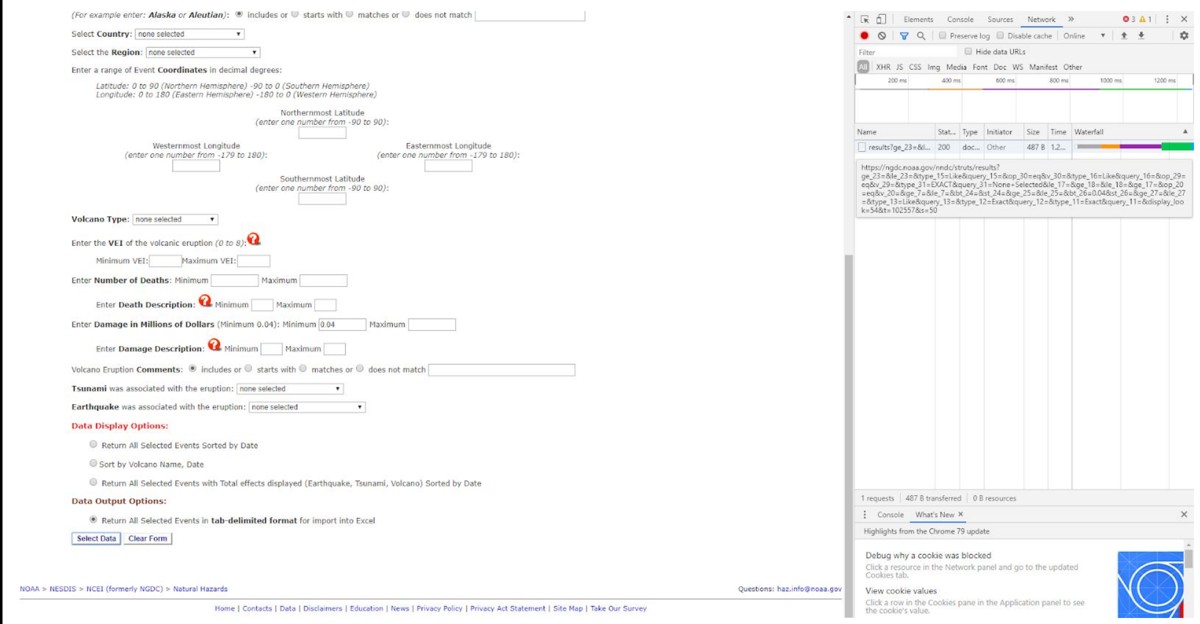
Step	**Task**
1	Fill the form in a way that maximum information is retrieved
2	Transform the data into the desired format
3	Replace the old dataset with the new one

Expected Result	A dataset with the eruptions of volcanoes until the present with all the information related to the damage that it caused.

3.3.1 Implementation

"In this section, we will see the implementation of the volcanoes ETL process. We did some "network" scraping to find the request."

The first thing to do in this case is to explore the structure of the web page to see if we could find the request behind the select data button. While testing we found a way that instead of sending us to the web view of the table it downloaded a file. Instead of web scraping the page we must look in the Network part and check if we can find the file path.



(For example enter: **Alaska** or **Aleutian**): [✓ includes or] starts with matches or does not match

Select **Country**: none selected

Select the **Region**: none selected

Enter a range of Event **Coordinates** in decimal degrees:

Latitude: 0 to 90 (Northern Hemisphere) -90 to 0 (Southern Hemisphere)
Longitude: 0 to 180 (Eastern Hemisphere) -180 to 0 (Western Hemisphere)

Northernmost Latitude *(enter one number from -90 to 90)*:

Westernmost Longitude *(enter one number from -179 to 180)*:

Easternmost Longitude *(enter one number from -179 to 180)*:

Southernmost Latitude *(enter one number from -90 to 90)*:

Volcano Type: none selected

Enter the **VEI** of the volcanic eruption *(0 to 8)*:

Minimum VEI: Maximum VEI:

Enter **Number of Deaths**: Minimum Maximum

Enter **Death Description**: Minimum Maximum

Enter **Damage in Millions of Dollars** (Minimum 0.04): Minimum 0.04 Maximum

Enter **Damage Description**: Minimum Maximum

Volcano Eruption **Comments**: [✓ includes or] starts with matches or does not match

Tsunami was associated with the eruption: none selected

Earthquake was associated with the eruption: none selected

Data Display Options:

Return All Selected Events Sorted by Date

Sort by Volcano Name, Date

Return All Selected Events with Total effects displayed (Earthquake, Tsunami, Volcano) Sorted by Date

Data Output Options:

[✓ Return All Selected Events in **tab-delimited format** for import into Excel]

Name	Stat...	Type	Initiator	Size	Time
results?ge_23=&l...	200	doc...	Other	487 B	1.2...

```
https://ngdc.noaa.gov/nndc/struts/results?
ge_23=&le_23=&type_15=Like&query_15=&op_30=eq&v_30=&type_16=Like&query_16=&op_29=
eq&v_29=&type_31=EXACT&query_31=None+Selected&le_17=&ge_18=&le_18=&ge_17=&op_20
=eq&v_20=&ge_7=&le_7=&bt_24=&st_24=&ge_25=&le_25=&bt_26=0.04&st_26=&ge_27=&le_27
=&type_13=Like&query_13=&type_12=Exact&query_12=&type_11=Exact&query_11=&display_loo
k=54&t=102557&s=50
```

1 requests | 487 B transferred | 0 B resources

Highlights from the Chrome 79 update

Debug why a cookie was blocked
Click a resource in the Network panel and go to the updated Cookies tab.

View cookie values
Click a row in the Cookies pane in the Application panel to see the cookie's value.

Select Data
Clear Form
NOAA > NESDIS > NCEI (formerly NGDC) > Natural Hazards
Questions: haz.info@noaa.gov
Home | Contacts | Data | Disclaimers | Education | News | Privacy Policy | Privacy Act Statement | Site Map | Take Our Survey
Elements
Console
Sources
Network
Preserve log
Disable cache
Online
Filter
Hide data URLs
All
XHR
JS
CSS
Img
Media
Font
Doc
WS
Manifest
Other
200 ms
400 ms
600 ms
800 ms
1000 ms
1200 ms
Waterfall
Console
What's New

Fig 21: Network Scraping

Once we find the link we can use it as a request and see what it returns. We get a text that is a representation of the table in TSV (tab-separated value). So thanks to tracking this we've done the two first steps at once. Looking at the query we find that there is a parameter that is called "t". If this is the current timestamp we cannot reuse the link because, in the end, we will not download any new information.

We made a test and used the same link in November and later in December. It showed a new volcano eruption on the 9th of December without changing the link, so we are now sure that this link retrieves the information until the present, no matter when it was executed.

Once this was checked we can now save the file and replace it every day without worrying about losing information. In the end, we have got a very simple code. Basically, we send a request and store that output as .tsv file in our ADLS.

3.4 Earthquakes Dataset

“Earthquakes are very common in America. If the infrastructure of the buildings is not specifically prepared to resist an earthquake it can causes damage by millions of dollars. It makes sense to track these events to consider them when calculating risks.”

Also from the USA, we have the United States Geological Survey (USGS). This institution is specialized in tracking earthquakes all around the globe. They release the data in different formats, so we can choose the one that better fits our purposes. In this dataset, we can find daily information about the earthquakes every two and a half days. Some of the important parameters that we have are timestamp, latitude, longitude, magnitude, type, place, error, status, location, etc. We will start tracking all this information and storing it.

This is the same source that Google uses when it displays information about earthquakes, so we can say that it is trustable enough. The problem here is that they don’t upload the whole historic every day, so we must create our dataset daily while monitoring this information. Ideally, we could find a larger dataset of earthquakes with the same information.

Step	Task
1	Download earthquake daily information
2	Append to our historic or set this day as historic if not exists
3	Delete duplicates from the historic

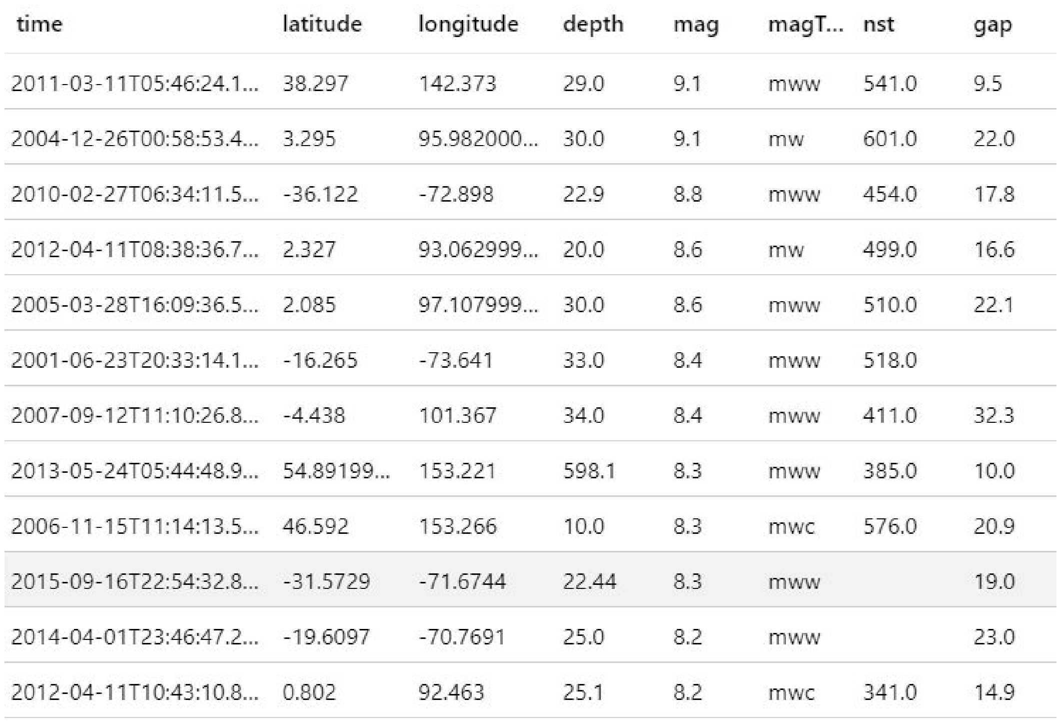
Expected Result	A dataset with all the earthquakes in the world updated every day.

3.4.1 Implementation

"In this section, we will show the procedure that we followed to end up with historical data since the year 2000 and how we extracted and appended it to have a dataset that is updated automatically every day."

The webpage has a tool that is a filter to show the earthquakes with custom filters. We will use this to get our historic data. We can force it to show it as CSV instead of using the interactive map. At first attempt, we've tried to retrieve data for the last 20 years. It broke and show an error because we are trying to download more data than what is allowed, but in this error, we had access to the request query, so now we can tune it by requests.

Our request is now retrieving data from the last 20 years, all those which has magnitude more than 6. We only need to save it to our data lake and set it as initial historical data.



time	latitude	longitude	depth	mag	magT...	nst	gap
2011-03-11T05:46:24.1...	38.297	142.373	29.0	9.1	mww	541.0	9.5
2004-12-26T00:58:53.4...	3.295	95.982000...	30.0	9.1	mw	601.0	22.0
2010-02-27T06:34:11.5...	-36.122	-72.898	22.9	8.8	mww	454.0	17.8
2012-04-11T08:38:36.7...	2.327	93.062999...	20.0	8.6	mw	499.0	16.6
2005-03-28T16:09:36.5...	2.085	97.107999...	30.0	8.6	mww	510.0	22.1
2001-06-23T20:33:14.1...	-16.265	-73.641	33.0	8.4	mww	518.0	
2007-09-12T11:10:26.8...	-4.438	101.367	34.0	8.4	mww	411.0	32.3
2013-05-24T05:44:48.9...	54.89199...	153.221	598.1	8.3	mww	385.0	10.0
2006-11-15T11:14:13.5...	46.592	153.266	10.0	8.3	mwc	576.0	20.9
2015-09-16T22:54:32.8...	-31.5729	-71.6744	22.44	8.3	mww		19.0
2014-04-01T23:46:47.2...	-19.6097	-70.7691	25.0	8.2	mww		23.0
2012-04-11T10:43:10.8...	0.802	92.463	25.1	8.2	mwc	341.0	14.9

Fig 22: Historical Earthquakes Data from USGS

Once we have this we only need to get the daily data and add it to our table. We will use requests in Python to retrieve this information, save it as daily and work with it. Since pandas cannot pass raw text in CSV to a Dataframe object we will store it temporarily in the ADLS and then retrieve it from them. The same happens with the historic. The idea is to run this notebook once per day, so we need to delete the duplicated entries to not store irrelevant data.

3.5 Hurricanes to risk

"From June to November, the USA suffers what they call hurricane season. With every passing year, these hurricanes are becoming more and more powerful. Nowadays, they cause several damages to both people and properties. Zurich was worried about the fact that they have no way to measure these events even though they have a big impact on the company."

This is probably one of the most difficult datasets to work with. Hurricanes are in general hard to transform into data, they are generated from different ways and they don't usually have a center. A hurricane can also generate other dangerous winds that take other routes.

Again, NOAA is the source of this information. The data is in KMZ/KML format and it is updated every hour if there is a hurricane or tropical storm. This means that most of the year there are no information available.

Instead of storing the natural disaster itself like we did until now, Zurich want to have a dataset of USA cities and know how prone a city is to suffer any kind of dangerous wind, this go from tropical storms to hurricanes. This must include not only the current hurricanes but also to historical data. The data visualization team also require us to store the hurricane itself in JSON format so they can visualize it in Power BI.

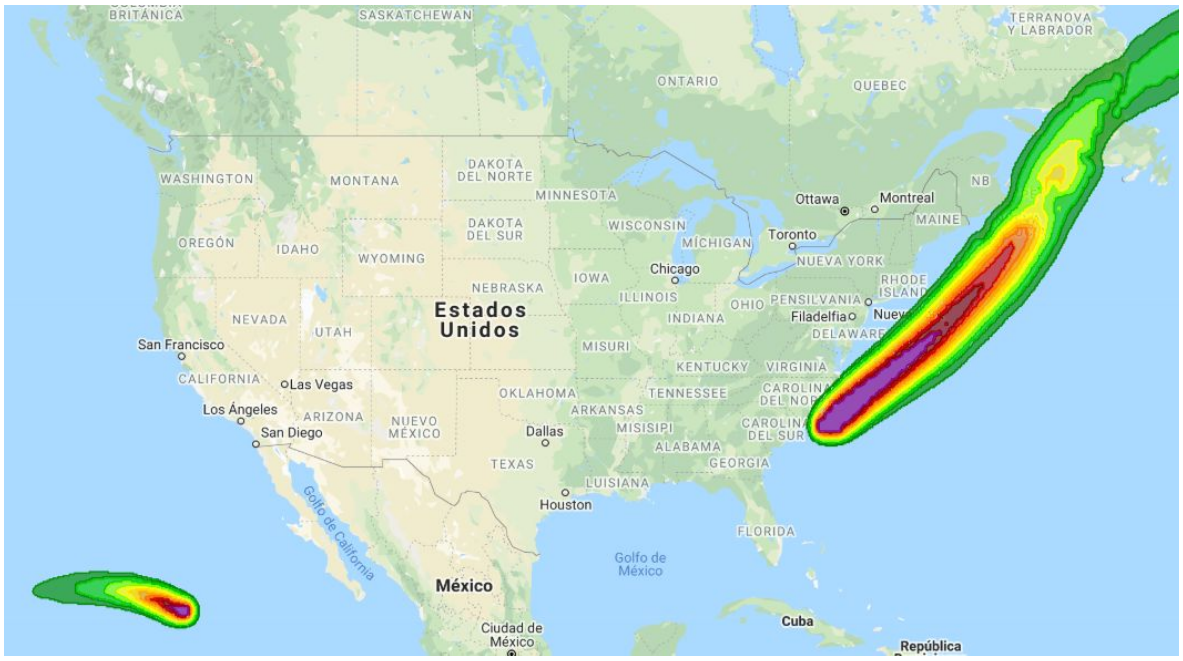


Fig 23: KML with probabilities of Tropical Cyclones

We have the hurricane represented as several points and each point has a value according to its danger associated. The idea is to create a cloud of points and connect them, and then check whether or not a city is inside the hurricane. We need to take into account the fact that not all points need to be considered as the hurricane itself. The value of each point represents the probability of having a phenomenon in this area, so we could ignore those points with low values.

Another problem is that not all the points in the dataset belong to the same hurricane, as we can see in Figure 30, several hurricanes could be happening at the same time. If we connect these points indistinctly we will find that all the space between hurricanes is also considered part of the hurricanes.

We also need to find a big dataset of cities from the USA. We want as much accuracy as possible, so bigger datasets of cities will give us better results. While using the Open Weather API we find that it has information from tons of cities around the world. If we could list all the cities that Open Weather API uses and then filter by the country we could have a good enough dataset where to start.

This new approach to the problem will make us work with two different Open Data datasets and also start applying some models that data scientists could apply in the future, such as clustering in order to detect different hurricanes.

Step	Task
1	Find a good dataset of cities from the USA
2	Download hurricanes information every 3 hours
3	Transform daily data to interpretable format
4	Find a Historic Dataset of hurricanes
5	Transform the historic if necessary
6	Apply K-means to detect different hurricanes for all the data
7	Check whether or not the cities are inside the dangerous zone
8	Extra: Plot the results in Power BI

Expected Result	A dataset with the USA cities and information about how many times they were affected by the three types of dangerous winds (Tropical storms, cyclones, and hurricanes). This dataset must be also updated daily.

3.5.1 Implementation

"In this section, we will see how we selected the cities dataset, the daily hurricane information and the hurricanes historical data. Then we will show its transformations, and the techniques applied to calculate the number of times that a city has been in danger."

For the city dataset search we went back to Open Weather API, we found that we can download a JSON file with the id that they use when calling the API, the coordinates, the country and the name of the city. This is great because we can filter by country. When we filter by the US we find that we have 43191 cities homogeneously distributed.

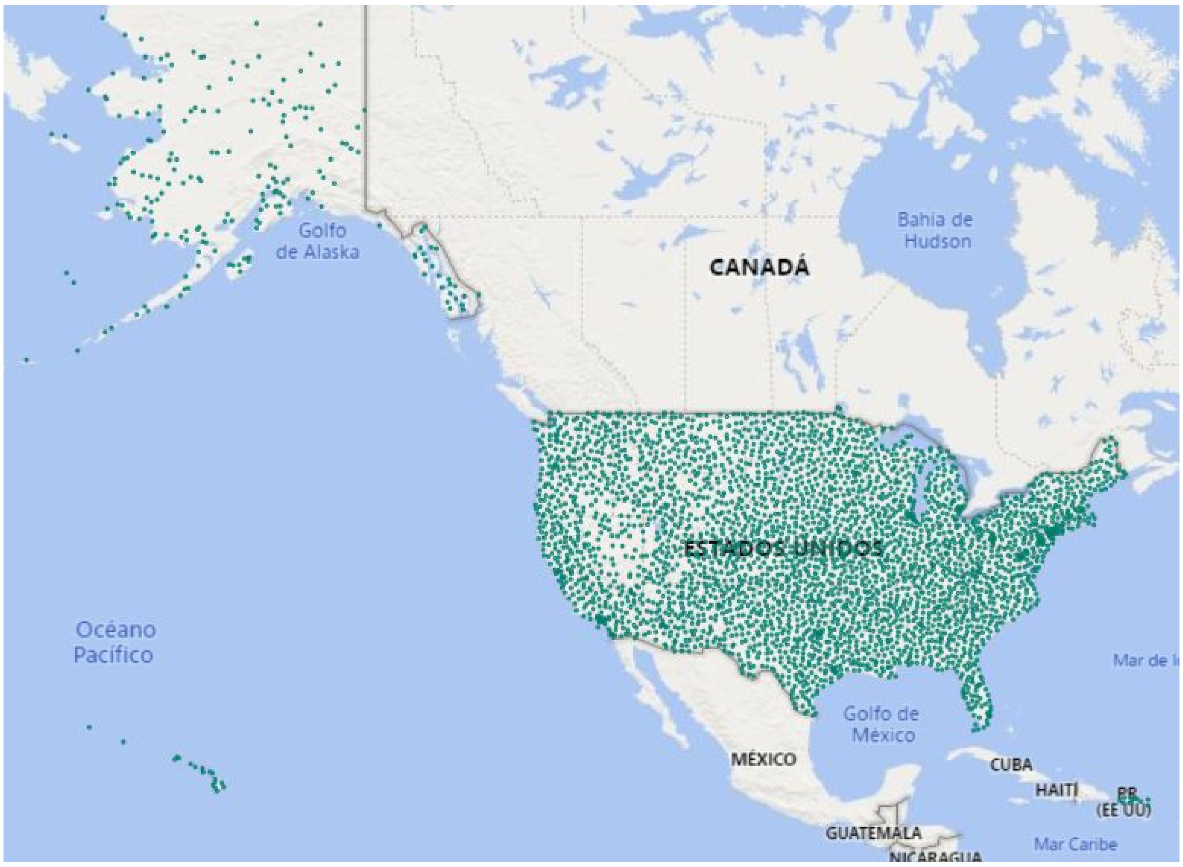
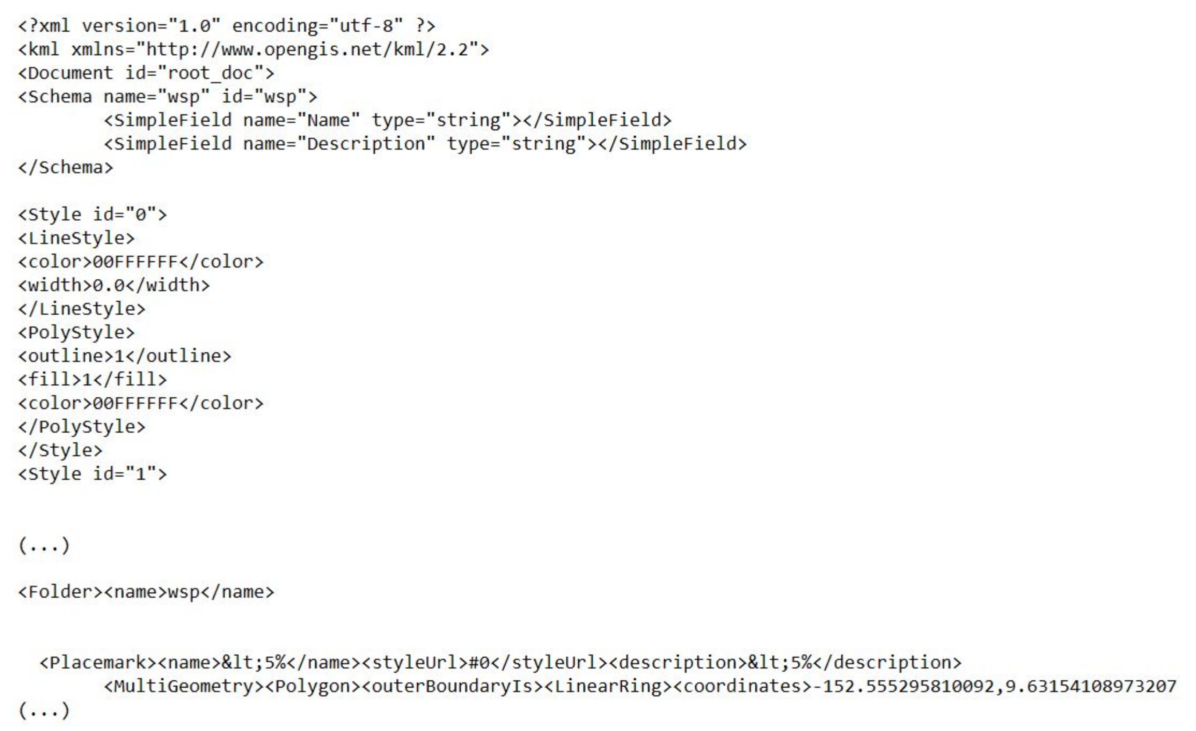


Fig 24: Coverage of cities in the USA

For the daily hurricane dataset, our first intuition is to download the data from the NOAA web page and try to manually visualize it in Power BI. Even though the main purpose in Power BI is to visualize data, we find that it has no support for either KMZ or KML. We investigated the program and tested several files that could be displayed on a world map. The conclusion was that Power BI mainly works with tables, and the way that it has to put points in a map is having a column that represents the longitude and another one that represents the latitude.

It makes sense that if a KMZ/KML file can be displayed in Google Maps, and Google Maps uses longitude and latitude, this information would be at some point inside the structure of the file. So the first thing to do is to read the format of the file.



```
<?xml version="1.0" encoding="utf-8" ?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document id="root_doc">
<Schema name="wsp" id="wsp">
        <SimpleField name="Name" type="string"></SimpleField>
        <SimpleField name="Description" type="string"></SimpleField>
</Schema>

<Style id="0">
<LineStyle>
<color>00FFFFFF</color>
<width>0.0</width>
</LineStyle>
<PolyStyle>
<outline>1</outline>
<fill>1</fill>
<color>00FFFFFF</color>
</PolyStyle>
</Style>
<Style id="1">


(...)

<Folder><name>wsp</name>


  <Placemark><name>&lt;5%</name><styleUrl>#0</styleUrl><description>&lt;5%</description>
        <MultiGeometry><Polygon><outerBoundaryIs><LinearRing><coordinates>-152.555295810092,9.63154108973207
(...)
```

Fig 25: Simplified version of what a KML file contains

As we can see in the picture, the raw data in a KML file is displayed as XML. The data is quite long but in the picture, we can see the structure. Points are displayed in an array and they are divided by polygons, lines or points each one with the different structures depending on the type. The first thing we must do is transform this XML file to something else that we can work with, for example, JSON.

We have created a function, using an external library, to convert XML to JSON by only calling this function. Now we can navigate through the data easily using Python. Once we downloaded the data, we saw that the first KML file could not be represented in Google Maps.

If we open the file and read it with the parser we can see that this file is not a common KML file. It contains a kind of index and it redirects us to other links that have attached a KMZ file. We saved every KMZ file independently. After some extractions of the data, we can see that every day we are getting different KMZ files. If there are any hurricanes or tropical storms they can have, or not, information about its cone of uncertainty, past track, track forecast, initial extent of winds, etc.

The most interesting, and constant, file is the one with the probability of having different dangerous winds, this includes 34-knot winds (Tropical Storms), 50-knot winds (Cyclones) and 64-knot winds (Hurricanes). Since these files are the only ones that are uploaded every day we will focus on them. We will keep storing the other files as well but we will focus on the probability ones. A representation of these files is what we see on Figure 23.

It is easy to infer from the probabilities files. We can see the probabilities as the impact caused by those dangerous winds. Meaning that zones with probability more than 70% are closer to the danger and are very likely to be affected by the winds.

The KMZ format is a compressed version of the data, the structure of which depends on the shape of the hurricane. This variability in the data made it especially challenging to obtain a readable JSON version from the KMZ. To address this issue, we have created a program that manages all the KMZ versions and transforms it to obtain a stable JSON file of the hurricane.

This file is a readable version of the KML original file. We can see the structure in Figure 25. But what we want is a new version that contains each point of the hurricane and information about it. After applying non-trivial transformations to the data we ended up with a JSON file where each row contains the longitude and latitude of the point, the kind of wind, the timestamp and the probability that this point is affected by the winds (we will consider it as magnitude). We had 30k ~ 60k rows on average for each hurricane. We tested the new format in Power BI.

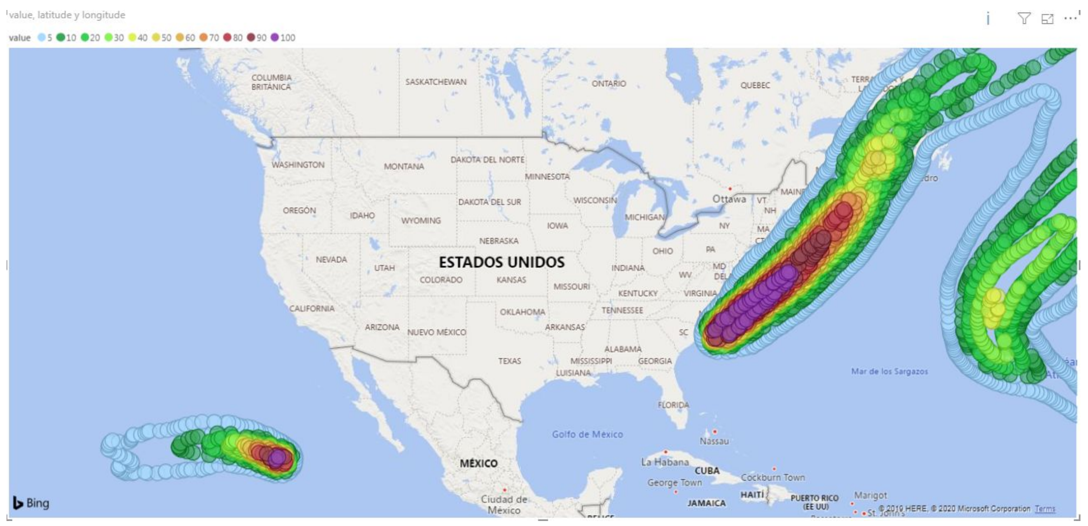
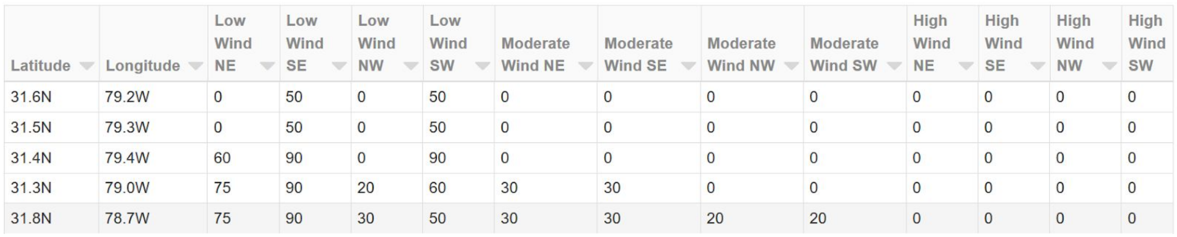

value, latitude y longitude
value ● 5 ● 10 ● 20 ● 30 ● 40 ● 50 ● 60 ● 70 ● 80 ● 90 ● 100
ESTADOS UNIDOS
MÉXICO
Golfo de México
Mar de los Sargazos
CUBA
Bing
© 2019 HERE, © 2020 Microsoft Corporation Terms

Fig 26: Hurricane Probabilities in Power BI

Even though the visualization can look worse than in Google Maps, Power BI allows us to apply filters, show the data in several ways, clean the data, work with more than one dataset at the same time and even create a video of how the hurricanes evolve based on the timestamp of the data.

For the historical dataset, we went back to Kaggle, there we find two historic datasets from 1851 until 2016. We have a dataset for the Atlantic and the Pacific oceans. Both of them are interesting since those are the oceans that surround the USA. As expected, the problem with these datasets is that they are in a different format than what we have done before, so we need to transform it before proceeding. This is one of the main issues when working with Open Data.

Until now we were able to transform the KMZ files to a JSON with the points delimiting the dangerous zones of the winds. This new format is quite different.



Latitude	Longitude	Low Wind NE	Low Wind SE	Low Wind NW	Low Wind SW	Moderate Wind NE	Moderate Wind SE	Moderate Wind NW	Moderate Wind SW	High Wind NE	High Wind SE	High Wind NW	High Wind SW
31.6N	79.2W	0	50	0	50	0	0	0	0	0	0	0	0
31.5N	79.3W	0	50	0	50	0	0	0	0	0	0	0	0
31.4N	79.4W	60	90	0	90	0	0	0	0	0	0	0	0
31.3N	79.0W	75	90	20	60	30	30	0	0	0	0	0	0
31.8N	78.7W	75	90	30	50	30	30	20	20	0	0	0	0

Fig 27: Historical dataset format pre-processed

They use different coordinates systems, and then they assign a center of the hurricane. They make a distinction between winds that correspond to our tropical storm, cyclone, and hurricane. The gist of this is that they take four directions from the center, NE, SE, NW, and SW and they indicate the distance in this direction where winds of that type happened. In the end, we should be able to understand the shape of the hurricane based on this data.

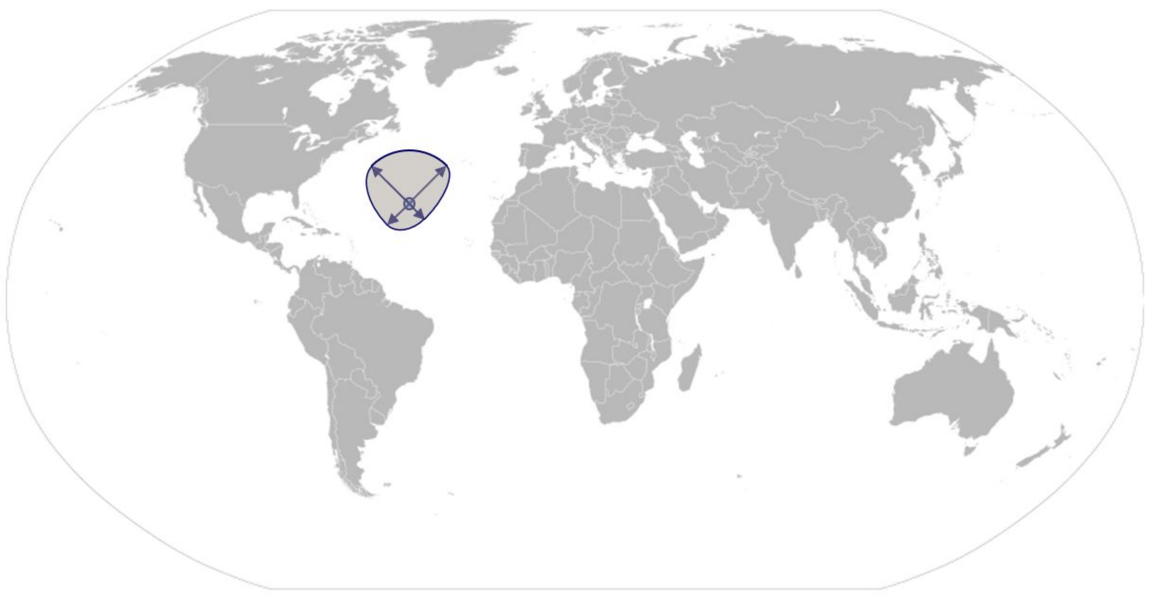


Fig 28: Visual representation of a hurricane in the new format

First of all, we want to transform the coordinates system. The first step is to pass from miles to kilometers, this is only a multiplication. After researching we find that we can transform those distances in kilometers to coordinates following a formula that takes into account the fact that this distance belongs to a sphere, not a plane.

Once this is done we must check if the data belong to the South or West, in those cases the will add the minus before. Remember that we use the global system where coordinates go from left, negative, to right, positive, and from the bottom, negative, to the top, positive.

Then we need to add or subtract that information depending on which row we are. For example, if we are in NE row we want to add the latitude and longitude, while if we are in NW we want to subtract the longitude and add the latitude. We take into account the type of wind that we are analyzing, so we also classify by tropical storm, cyclone or hurricane.

We do this for both datasets, the Atlantic and Pacific. We merge this into a new dataset and we have now a historic for the USA in a more treatable format. Now we must come up with some method to see if cities were inside the danger area or not.

We researched and find that exists a library called shapely[5] that can create polygons from points and check whether or not another point is inside this polygon. The idea is to model our hurricanes as polygons and our cities and points to see if they were affected or not by the winds.

If we apply this method directly to our cities we find strange results, like, most of the cities are affected most of the time, even those that are in the center of the continent that we know that they have never been reached by any kind of dangerous wind. This happens because we did not take into account the fact that we have several winds in one file and we cannot count them as one single polygon.

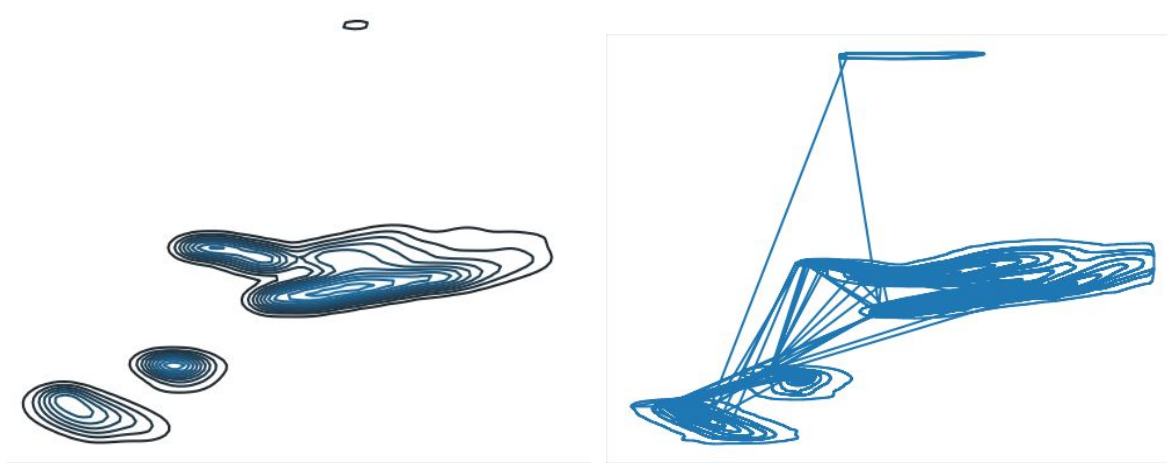


Fig 29: Expected representation vs real representation

As we can see in Figure 29 the polygon that we have has more lines than the reality. The problem came when a city between winds is not in danger but with this representation it considers the space between hurricanes as a part of hurricanes. We need to find a way to classify each wind individually.

This problem is perfectly solved by any clustering method since what we have is a bunch of unlabeled data points distributed in the space and we want to find reasonable groups of points. For this concrete problem, we will use k means[6] from sklearn[7] library.

The first thing to take into consideration in this kind of problem is the number of clusters that we should use. To do this we will use something called silhouette score. The silhouette value is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). The silhouette ranges from −1 to +1, where a high value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters. If most objects have a high value, then the clustering configuration is appropriate. If many points have a low or negative value, then the clustering configuration may have too many or too few clusters[8].

What we will do is to set a range of clusters, from 2 to 10, and see which score we get when we test the k means model. We will pick the number of clusters that have the best score + 4 since we want to be sure that every hurricane has his own cluster. If we use more clusters than necessary we can have one polygon divided into some parts, which does not really affect when we are checking if a city is inside or not. On the other hand, if we use fewer clusters we could end putting in the same group two different hurricanes and had the same problem that we try to solve.

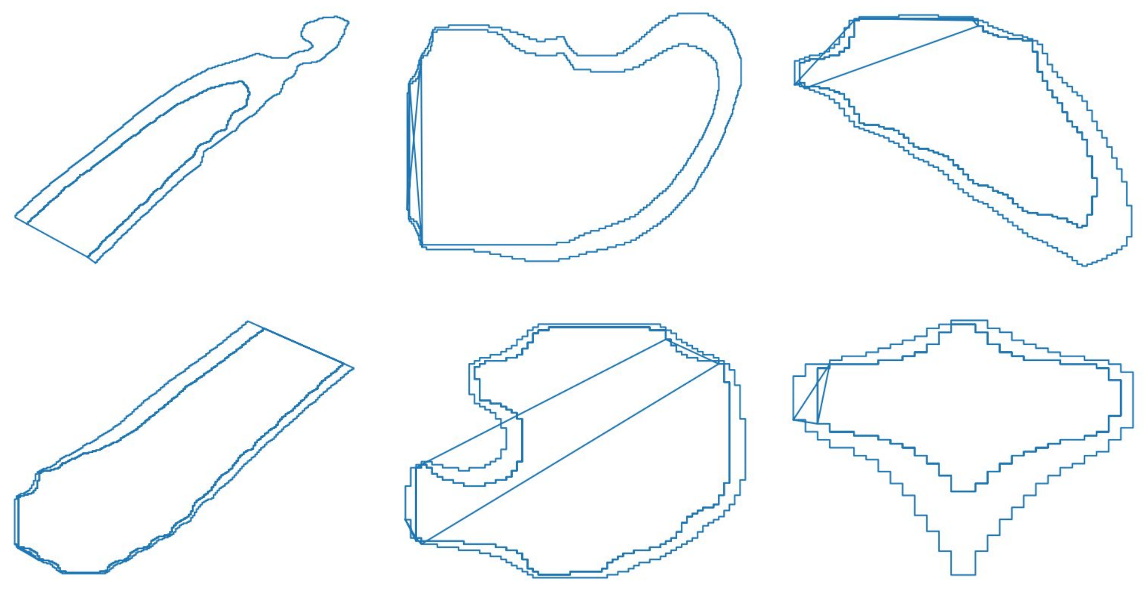


Fig 30: Clustered Data (Polygons) from Hurricanes

Once we have the polygons of each dangerous wind individually the only thing left is to, for every cluster, check our list of cities and set them as points. We check if the point (city) is inside the polygon (hurricane), if this happens we will check the type of wind that we are looking at, and then increase the times that the city was visited by this wind one time.

Once this is done for the historical data we will go for the data that we retrieve every day. As we know this data contains the probabilities, not the representation of the wind itself. We decided to filter this data for those values where the probability is at least 90.

Now that we have a dataset with information about the city, coordinates, and name, and how many times it was in danger we can plot it in Power BI. We also added some filters in the dashboard so they can filter with no need to code. They have an output window with the number of cities affected by their current choice of filters.

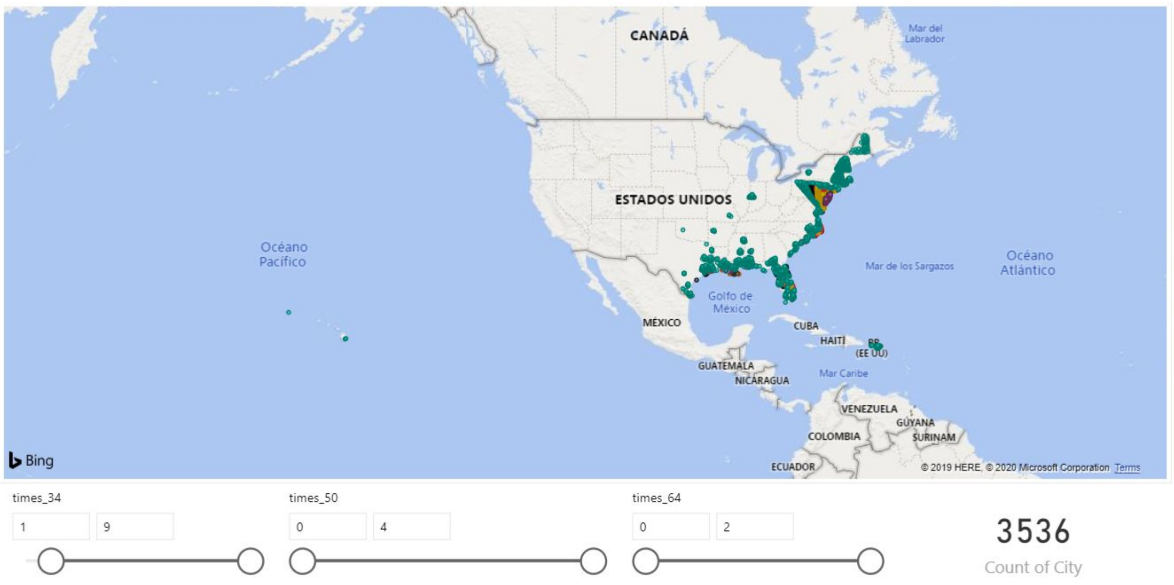


Fig 31: Power BI Dashboard with the Data and the Filters

3.6 Wildfires Dataset

"Fire is another phenomenon that causes several damages to properties, it will be interesting to monitor this kind of events in order to evaluate the risk that has a certain property to be involved in a wildfire based on his historical information."

It is well-know that fires are increasing as time passes. Events like what happened in Australia at the beginning of 2020, are very common this decade with an average of 2,279 fires and 44,568 acres burnt per year.[11] This factor is not been taking into account when calculating the risk of a property, despite being one of the most devastating events that tend to destroy totally or partially the place where it was originated.

It is interesting to track the fires that happen in nature but Zurich, as an insurance company, is interested in all kinds of fires, especially those that occur in urban zones, like cities or towns. We must find an updated and trustable dataset that tracks all these events.

We find that NASA has a website with a map with wildfires and it is updated every 24, 48, 72 hours and even weekly. After searching and filtering with their web tool we saw that it tracks even the fires reported by cities and towns.

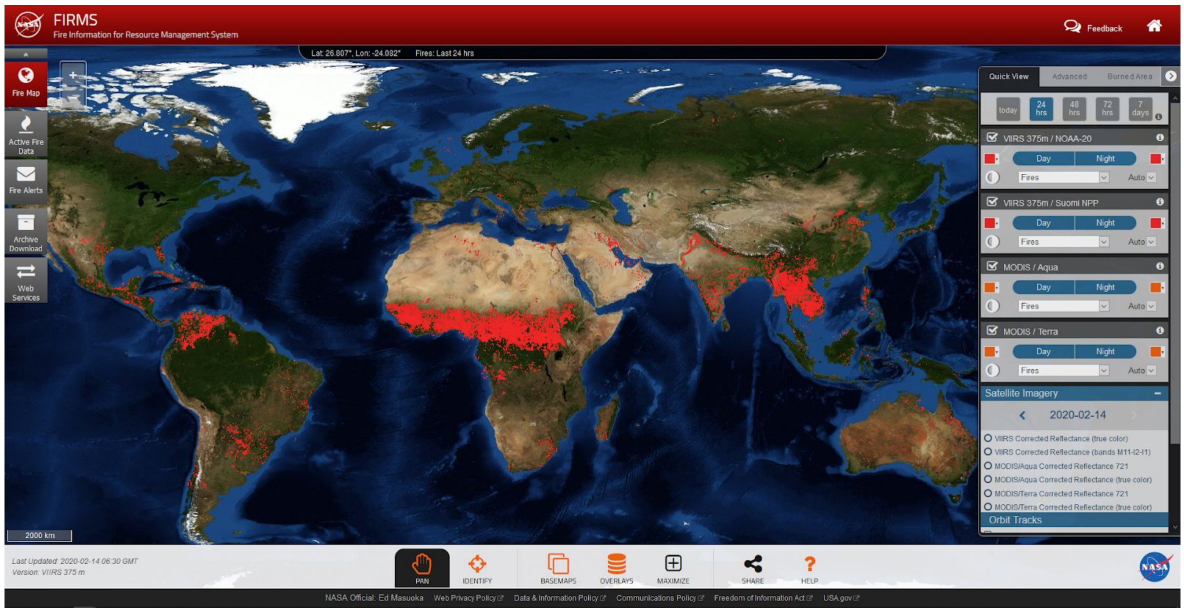

FIRMS
Fire Information for Resource Management System
Feedback
Lat: 26.807°, Lon: -24.082°
Fires: Last 24 hrs
Fire Map
Active Fire Data
Fire Alerts
Archive Download
Web Services
Quick View
Advanced
Burned Area
today
24 hrs
48 hrs
72 hrs
7 days
VIIRS 375m / NOAA-20
Day
Night
Fires
Auto
VIIRS 375m / Suomi NPP
MODIS / Aqua
MODIS / Terra
Satellite Imagery
2020-02-14
VIIRS Corrected Reflectance (true color)
VIIRS Corrected Reflectance (bands M11-I2-I1)
MODIS/Aqua Corrected Reflectance 721
MODIS/Aqua Corrected Reflectance (true color)
MODIS/Terra Corrected Reflectance 721
MODIS/Terra Corrected Reflectance (true color)
Orbit Tracks
2000 km
Last Updated: 2020-02-14 06:30 GMT
Version: VIIRS 375 m
PAN
IDENTIFY
BASEMAPS
OVERLAYS
MAXIMIZE
SHARE
HELP
NASA Official: Ed Masuoka
Web Privacy Policy
Data & Information Policy
Communications Policy
Freedom of Information Act
USA.gov

Fig 32: NASA Fire Map

We want to extract this information and keep an updated database with all the fires tracked since the day we have started until the present. Since what we have is points in a map we should probably work with KML/KMZ files.

[11] National Interagency Fire Center (n.d.). Retrieved from https://www.nifc.gov/fireInfo/nfn.htm

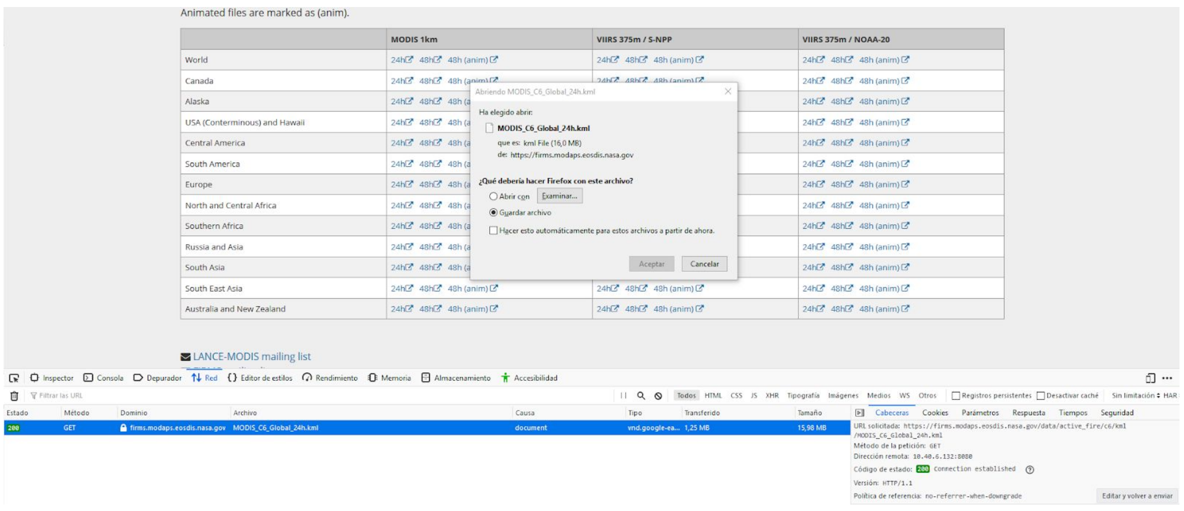
Step	Task
1	Extract information from the NASA Fire Map
2	Transform this file from KMZ/KML to table
3	Append to the historic if exists
4	Upload the data to the ADLS

Expected Result	A dataset with the information of the fires since the day that the program is launched until the present, updated daily.

3.6.1 Implementation

"We will use some techniques that we have learned during this project like Network Scrapping and KML/KMZ data extraction. Using all these known techniques and functions we can simplify all the processes."

We find on the web where we can download the data from the latest 24h, 48h or 7 days. The format is KML as expected. It does not have API to retrieve this data but a button to manually downloads. Using what we have learned while downloading volcanos we can do what we have called "Network Scrapping" and find the request that is sent when we press this button.



Animated files are marked as (anim).

	MODIS 1km	VIIRS 375m / S-NPP	VIIRS 375m / NOAA-20
World	24h 48h 48h (anim)	24h 48h 48h (anim)	24h 48h 48h (anim)
Canada	24h 48h 48h (anim)	24h 48h 48h (anim)	24h 48h 48h (anim)
Alaska	24h 48h 48h (a		24h 48h 48h (anim)
USA (Conterminous) and Hawaii	24h 48h 48h (a		24h 48h 48h (anim)
Central America	24h 48h 48h (a		24h 48h 48h (anim)
South America	24h 48h 48h (a		24h 48h 48h (anim)
Europe	24h 48h 48h (a		24h 48h 48h (anim)
North and Central Africa	24h 48h 48h (a		24h 48h 48h (anim)
Southern Africa	24h 48h 48h (a		24h 48h 48h (anim)
Russia and Asia	24h 48h 48h (a		24h 48h 48h (anim)
South Asia	24h 48h 48h (a		24h 48h 48h (anim)
South East Asia	24h 48h 48h (anim)	24h 48h 48h (anim)	24h 48h 48h (anim)
Australia and New Zealand	24h 48h 48h (anim)	24h 48h 48h (anim)	24h 48h 48h (anim)

Abriendo MODIS_C6_Global_24h.kml

Ha elegido abrir:

MODIS_C6_Global_24h.kml

que es: kml File (16,0 MB)

de: https://firms.modaps.eosdis.nasa.gov

¿Qué debería hacer Firefox con este archivo?

( ) Abrir con [Examinar...]

(●) Guardar archivo

[ ] Hacer esto automáticamente para estos archivos a partir de ahora.

[Aceptar] [Cancelar]

LANCE-MODIS mailing list

Estado	Método	Dominio	Archivo	Causa	Tipo	Transferido	Tamaño
200	GET	firms.modaps.eosdis.nasa.gov	MODIS_C6_Global_24h.kml	document	vnd.google-ea...	1,25 MB	15,98 MB

URL solicitada: https://firms.modaps.eosdis.nasa.gov/data/active_fire/c6/kml/MODIS_C6_Global_24h.kml

Método de la petición: GET

Dirección remota: 10.40.6.132:8080

Código de estado: 200 Connection established

Versión: HTTP/1.1

Política de referencia: no-referrer-when-downgrade

Fig 33: Network Scrapping with Wildfires

Now we know the link where we need to request in our code. Using the same functions that we have used when working with hurricanes, we extract information about longitude and latitude. Then we can also extract other interesting information that is displayed on the webpage like the timestamp, the brightness, which allows us to imagine the magnitude of the fire, the track, the scan, the satellite, if it was during day or night, etc. We save all this information in a pandas dataframe.

As we have seen, we can retrieve information from the last week, so the first iteration of this project will be to download this file and set this dataset as historic, a poor one but NASA does not provide any older information. Every day we extract the information and append it to the last historic dataset.

3.7 Automatization of the ETL

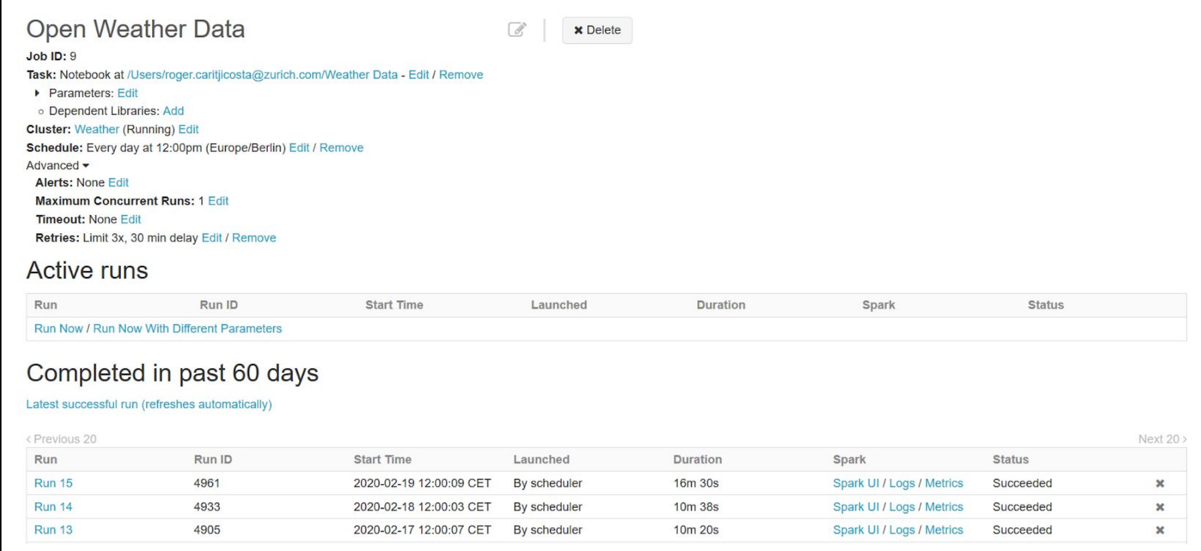
"Here we will see how we automatized all the procedure through Databricks Jobs, to auto-download all the data daily."

Once we have created the Databricks Notebooks that do the ETL process, we want it to work standalone and automated following our prerequisites. The code was cleaned and optimized. We followed the given pattern where the first cell contains the main functions and parameters. All libraries were installed in the same cluster and checked that there are no compatibility errors. Notebooks were distributed according to their retrieval period.

We ended with 3 different jobs. The first one is the Open Weather Data job. This Job executes a notebook with all the extraction process of weather. This notebook has his own job because it is the one that takes longer, remember that we wait one second between requests to not exceed the API request limit, and it can fail if we encounter some country that has a non-ASCII character when looking random cities. It happened very few times (<1%), but if it fails we want to run the notebook again. It is executed every day at midday.

The second one is called Hurricane Data Extractor Job. This job executes all the procedures from extracting the hurricane in KML format until the calculation per city in the USA. Instead of using the historic, we transform the KML to data points and apply the same effect that we did before. This job is executed every 3 hours. If there are any risks of dangerous winds we want to monitor it as continuously as possible. Most of the time of the year this job fails since, according to our prerequisites, it must return a Failed status if no data was retrieved.

The last job is the Weather Job. This job contains all the code to download the rest of the Open Data: Wildfires, Earthquakes, and Volcanos. It is executed every day at midday.


Open Weather Data

Job ID: 9
Task: Notebook at /Users/roger.caritjicosta@zurich.com/Weather Data - Edit / Remove
- Parameters: Edit
- Dependent Libraries: Add

Cluster: Weather (Running) Edit
Schedule: Every day at 12:00pm (Europe/Berlin) Edit / Remove
Advanced
Alerts: None Edit
Maximum Concurrent Runs: 1 Edit
Timeout: None Edit
Retries: Limit 3x, 30 min delay Edit / Remove

Active runs

Run	Run ID	Start Time	Launched	Duration	Spark	Status
Run Now / Run Now With Different Parameters						

Completed in past 60 days

Latest successful run (refreshes automatically)

‹ Previous 20 Next 20 ›

Run	Run ID	Start Time	Launched	Duration	Spark	Status	
Run 15	4961	2020-02-19 12:00:09 CET	By scheduler	16m 30s	Spark UI / Logs / Metrics	Succeeded	✖
Run 14	4933	2020-02-18 12:00:03 CET	By scheduler	10m 38s	Spark UI / Logs / Metrics	Succeeded	✖
Run 13	4905	2020-02-17 12:00:07 CET	By scheduler	10m 20s	Spark UI / Logs / Metrics	Succeeded	✖

Fig 34: Open Weather Data Job information dashboard

4. FROM DATA LAKE TO DATA CATALOG

"If you have tons of data but you are not able to find what you need despite being stored, do you really have any data?"

It is said nowadays that too much information leads to disinformation. It is very common in big companies that every user has his data in a concrete format stored in different places. We will try to solve this problem by implementing a Data Catalog.

Until now we have been storing all our data in a Data Lake in Azure. The problem is that this ADLS does not allow us to search in the data as well as we need. There is also the problem that, like everything in Azure, needs a lot of permissions, and subscriptions. There are no previsualization of the data and no description at all. It is hard to understand what are we looking at if we are not the owners and we know all the procedures behind.

Ideally, we want something that can be understood by the marketing team, the human resources team, engineers, etc. In order to reach this, we require some features.

- Description of the data.
- Labeled data.
- Downloadable data, manually and by API.
- Uploadable data, manually and by API.
- Easy to access.
- Secure.
- Optional: Previsualization of the data.
- Optional: Agrupations of data by topic.
- Optional: Agrupations of data by organization.
- Optional: Search by content in data, not only by tags.

We have done the research and contacted several companies that showed us their data catalogs products. We took into account all the previous requisites plus the economic cost that it has for the company, since that if it works, it could be implemented in the whole company, even in other countries. After all this process we end up with 3 options.

4.1 Watson Knowledge Catalog vs Dremio vs CKAN

"In this section, we will show, at a very high level, the process that we followed until we made the decision."

After hours of meetings, discussions, and tests we end up with these 3 options as a possible candidate for the Data Catalog.

The IBM Watson Knowledge Catalog[12] had very good functions that match with our requisites. It had a very good searching system, they had internal tools that can be purchased as extras to do some prework on the data. They had a very good governance system, so not everybody can manage, create, delete or even see all the data. This allows us to think in terms of departments and countries, so we could put all kinds of data and then decide who had access to what.

Some bad things about this are the fact that it was too expensive, even the basic with no extra add-ons or tools. All the data that we upload there will then belong to IBM, this could be a problem when we manage personal information data. While negotiating with them they insisted on the fact that they require all the information about the project, even before we agreed to anything.

Dremio[13] had really good features. It had the option to create collections from ADLS. It also had a very good governance system and its UI is easy to understand to everyone. It has a very good feature, you can search from the tags and name but also from the values from the data. It means that you could search for data from Europe, for example, and you will find those tables that in some fields have Europe as value, not only on the tags. The price was correct according to the given services.

The bad things about Dremio are that data has no description. We thought that this problem could be solved by putting several tags but we find that the API does not support tagging. We could automatically upload all the data but the tagging should be done manually.

CKAN[14] is an open-source data portal used by many governments in the world. It has a very good API that allows creating public or private datasets, divide them by groups, organizations, and tags. Is very intuitive and its search system can search by tags and word in the description of the dataset and resource. It can previsualize the data in a table format or in a map if the data has longitude and latitude. Since it is open-source the only cost will be the maintainer of the machine where we decide to allocate it.

[12] Watson Knowledge Catalog. Retrieved from https://www.ibm.com/uk-en/cloud/watson-knowledge-catalog
[13] Dremio. Retrieved from https://www.dremio.com/
[14] CKAN. Retrieved from https://ckan.org/

Some bad things about CKAN are that it must be deployed by hand. We need to work on the front-end and back-end, there is no tool or online portal created like the cases before. This means a lot of work focused on developing a web.

Regarding the required features we can see that all of them cover almost every feature, even the optional ones:

	Watson Knowledge	Dremio	CKAN
Description of the data.	✔	✖	✔
Labeled data.	✔	✔	✔
Downloadable data, manually and by API.	✔	✔	✔
Uploadable data, manually and by API.	✔	✔	✔
Easy to access.	✖	✔	✔
Secure.	✔	✔	✔
Optional: Previsualization of the data.	✔	✖	✔
Optional: Agrupations of data by topic.	✖	✖	✔
Optional: Agrupations of data by organization.	✔	✖	✔
Optional: Search by content in data, not only by tags.	✖	✔	✖

In more general terms we can compare the pros and cons of each option:

Watson Knowledge	
PROs ● Pre-work data. ● Powerful Searching. ● External support	CONs ● Expensive. ● Loss of ownership of data. ● Not an internal tool. ● They ask for too much information about internal projects. ● Not trustable.

Dremio	
PROs ● Good price. ● Integration with ADLS. ● Good UI.	CONs ● No description of the data. ● Not taggable by API.

CKAN	
PROs ● Free. ● Internal tool. ● Has most of the features. ● Adaptable UI. ● Good enough search system.	CONs ● It must be developed from 0. ● Support only in forums.

After taking into account all the things mentioned before we ended up choosing CKAN as our Data Catalog. It was the most complete option after all. We decided that it will be running in a Virtual Machine in Azure. In this way, we can adapt the machine according to our needs and it is running in the cloud.

4.2 CKAN Overview

"In this section, we will explain how does CKAN works from a user/admin point of view, all of the available actions that can be performed from the UI. "

We find out a template and we modified some files to change the visualization of the catalog to make it more Zurich friendly. We used MobaXterm to access the data of the web via ssh. We will focus more on the functionalities rather than the visual aspect since this is not our scope in this project.

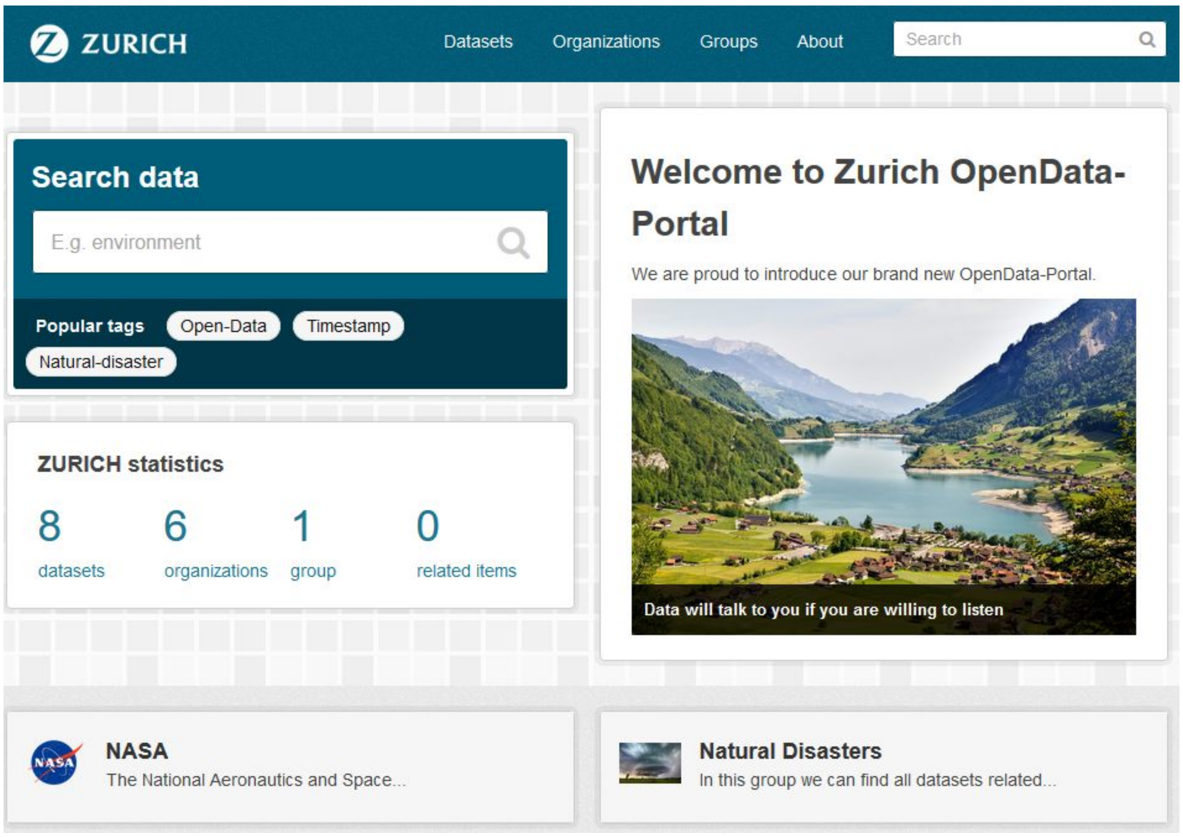
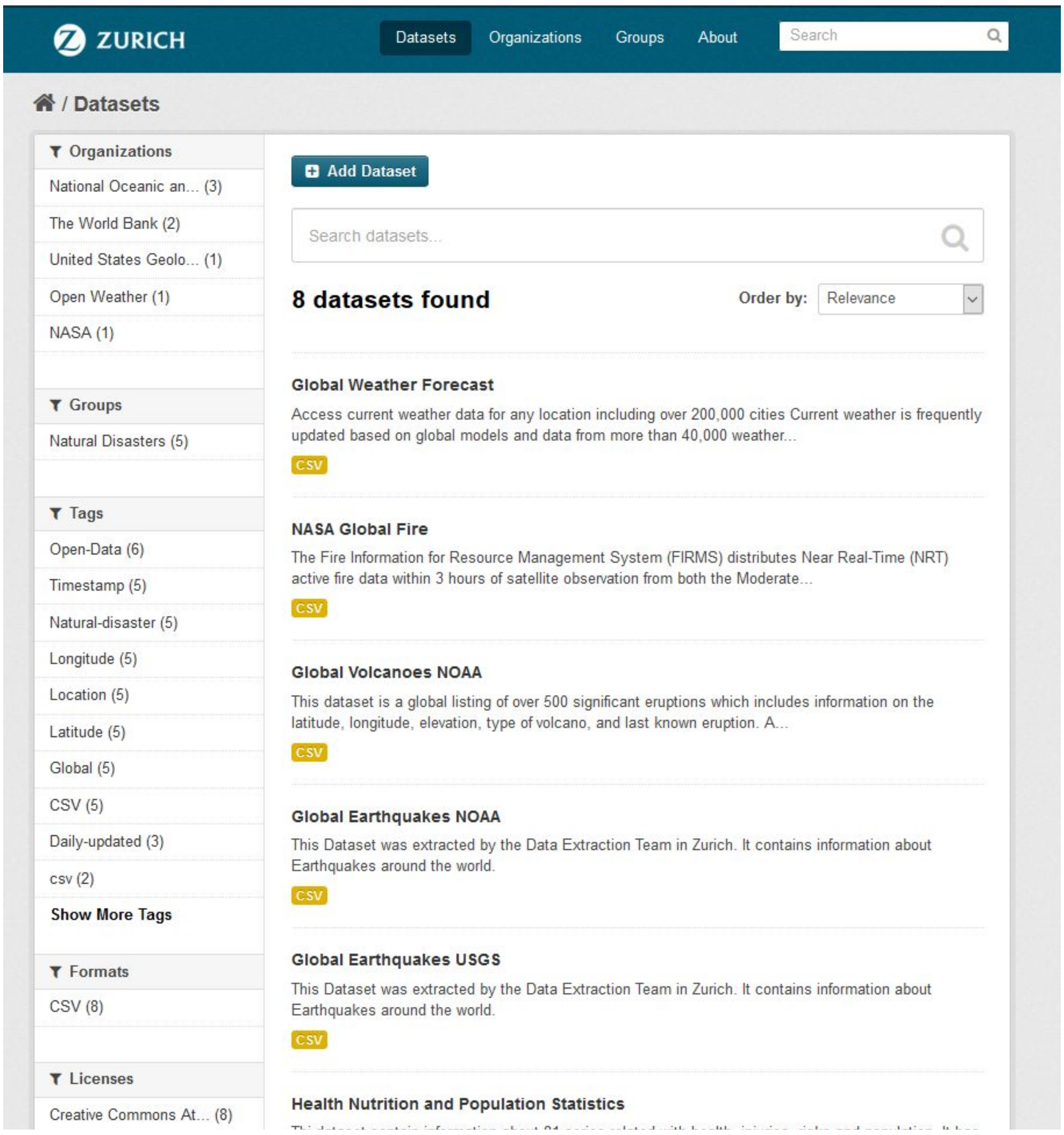

ZURICH
Datasets
Organizations
Groups
About
Search
Search data
E.g. environment
Popular tags
Open-Data
Timestamp
Natural-disaster
ZURICH statistics
8 datasets
6 organizations
1 group
0 related items
Welcome to Zurich OpenData-Portal
We are proud to introduce our brand new OpenData-Portal.
Data will talk to you if you are willing to listen
NASA
The National Aeronautics and Space...
Natural Disasters
In this group we can find all datasets related...

Fig 35: Homepage of the Data Portal

As we can see, the homepage is summarizing the content and features of our Data Catalog. We can see that we have some datasets, something called Organizations and Groups. We have a searcher and the most popular tags in the catalog. We will focus first on the main part of a data catalog, the datasets.


ZURICH

Datasets | Organizations | Groups | About | Search

/ Datasets

Organizations
- National Oceanic an... (3)
- The World Bank (2)
- United States Geolo... (1)
- Open Weather (1)
- NASA (1)

Groups
- Natural Disasters (5)

Tags
- Open-Data (6)
- Timestamp (5)
- Natural-disaster (5)
- Longitude (5)
- Location (5)
- Latitude (5)
- Global (5)
- CSV (5)
- Daily-updated (3)
- csv (2)
- **Show More Tags**

Formats
- CSV (8)

Licenses
- Creative Commons At... (8)

Add Dataset

Search datasets...

8 datasets found Order by: Relevance

Global Weather Forecast
Access current weather data for any location including over 200,000 cities Current weather is frequently updated based on global models and data from more than 40,000 weather...
CSV

NASA Global Fire
The Fire Information for Resource Management System (FIRMS) distributes Near Real-Time (NRT) active fire data within 3 hours of satellite observation from both the Moderate...
CSV

Global Volcanoes NOAA
This dataset is a global listing of over 500 significant eruptions which includes information on the latitude, longitude, elevation, type of volcano, and last known eruption. A...
CSV

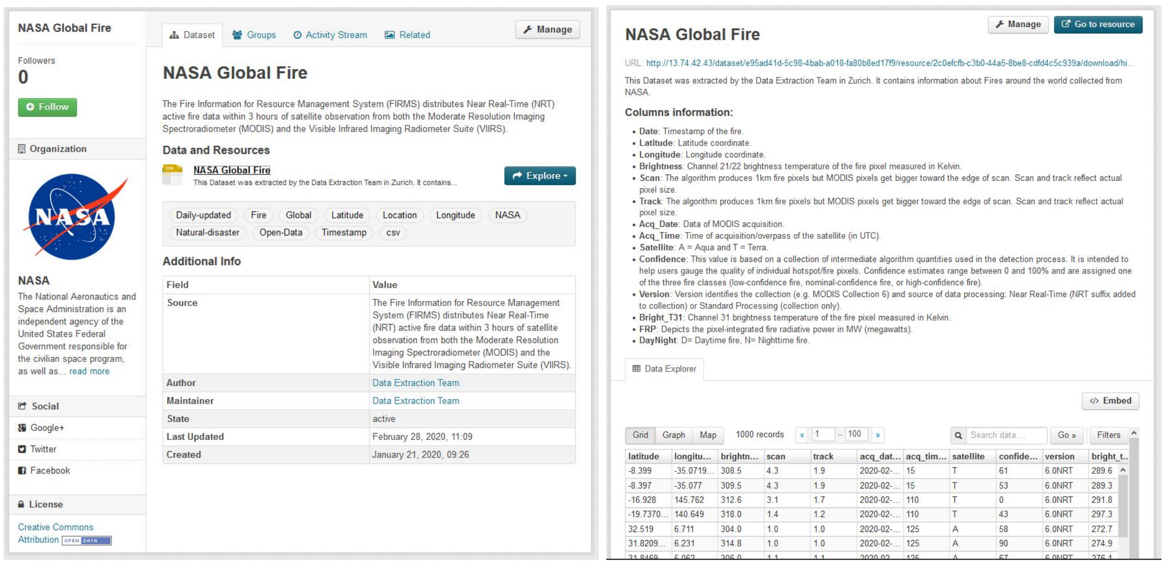
Global Earthquakes NOAA
This Dataset was extracted by the Data Extraction Team in Zurich. It contains information about Earthquakes around the world.
CSV

Global Earthquakes USGS
This Dataset was extracted by the Data Extraction Team in Zurich. It contains information about Earthquakes around the world.
CSV

Health Nutrition and Population Statistics

Fig 36: Dataset Search

Here we can see some of the datasets available. They show the title, the description and the format of the data. We can search using free text or by filtering by tags, groups or organizations on the left part. A very interesting thing about the search system is that if we search for a word that is in the description of the dataset or in the resource description (the file itself) it will be displayed too, even if the word is not in the tags. We will use this for content search. We will put as description a list with the columns, so if someone is looking for some concrete column it will appear in the search.



NASA Global Fire

Followers
0

Follow

Organization

NASA
The National Aeronautics and Space Administration is an independent agency of the United States Federal Government responsible for the civilian space program, as well as... read more

Social
- Google+
- Twitter
- Facebook

License
Creative Commons Attribution

Dataset | Groups | Activity Stream | Related | Manage

NASA Global Fire

The Fire Information for Resource Management System (FIRMS) distributes Near Real-Time (NRT) active fire data within 3 hours of satellite observation from both the Moderate Resolution Imaging Spectroradiometer (MODIS) and the Visible Infrared Imaging Radiometer Suite (VIIRS).

Data and Resources

NASA Global Fire
This Dataset was extracted by the Data Extraction Team in Zurich. It contains...
Explore

Daily-updated · Fire · Global · Latitude · Location · Longitude · NASA · Natural-disaster · Open-Data · Timestamp · csv

Additional Info

Field	Value
Source	The Fire Information for Resource Management System (FIRMS) distributes Near Real-Time (NRT) active fire data within 3 hours of satellite observation from both the Moderate Resolution Imaging Spectroradiometer (MODIS) and the Visible Infrared Imaging Radiometer Suite (VIIRS).
Author	Data Extraction Team
Maintainer	Data Extraction Team
State	active
Last Updated	February 28, 2020, 11:09
Created	January 21, 2020, 09:26

Manage | Go to resource

NASA Global Fire

URL: http://13.74.42.43/dataset/e95ad41d-5c98-4bab-a018-fa80b8ed17f9/resource/2c0efcfb-c3b0-44a5-8be8-cdfd4c5c939a/download/hi...

This Dataset was extracted by the Data Extraction Team in Zurich. It contains information about Fires around the world collected from NASA.

Columns information:

- **Date**: Timestamp of the fire.
- **Latitude**: Latitude coordinate.
- **Longitude**: Longitude coordinate.
- **Brightness**: Channel 21/22 brightness temperature of the fire pixel measured in Kelvin.
- **Scan**: The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
- **Track**: The algorithm produces 1km fire pixels but MODIS pixels get bigger toward the edge of scan. Scan and track reflect actual pixel size.
- **Acq_Date**: Data of MODIS acquisition.
- **Acq_Time**: Time of acquisition/overpass of the satellite (in UTC).
- **Satellite**: A = Aqua and T = Terra.
- **Confidence**: This value is based on a collection of intermediate algorithm quantities used in the detection process. It is intended to help users gauge the quality of individual hotspot/fire pixels. Confidence estimates range between 0 and 100% and are assigned one of the three fire classes (low-confidence fire, nominal-confidence fire, or high-confidence fire).
- **Version**: Version identifies the collection (e.g. MODIS Collection 6) and source of data processing: Near Real-Time (NRT suffix added to collection) or Standard Processing (collection only).
- **Bright_T31**: Channel 31 brightness temperature of the fire pixel measured in Kelvin.
- **FRP**: Depicts the pixel-integrated fire radiative power in MW (megawatts).
- **DayNight**: D= Daytime fire, N= Nighttime fire.

Data Explorer

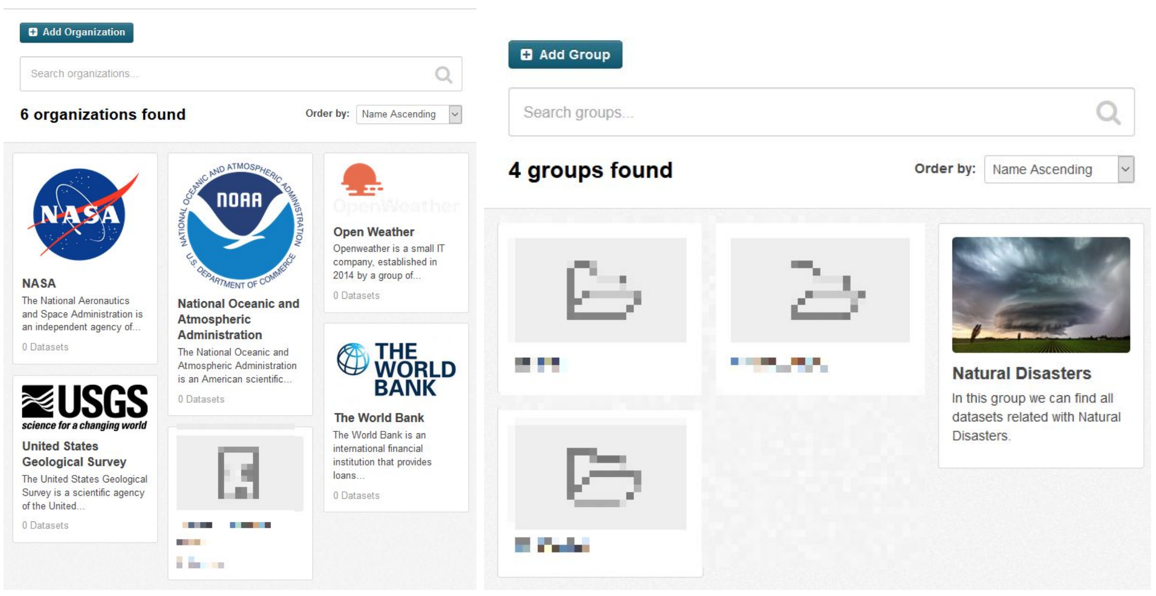
Embed

Grid | Graph | Map   1000 records   « 1 – 100 »   Search data ...   Go »   Filters

latitude	longitu...	brightn...	scan	track	acq_dat...	acq_tim...	satellite	confide...	version	bright_t..
-8.399	-35.0719...	308.5	4.3	1.9	2020-02-...	15	T	61	6.0NRT	289.6
-8.397	-35.077	309.5	4.3	1.9	2020-02-...	15	T	53	6.0NRT	289.3
-16.928	145.762	312.6	3.1	1.7	2020-02-...	110	T	0	6.0NRT	291.8
-19.7370...	140.649	318.0	1.4	1.2	2020-02-...	110	T	43	6.0NRT	297.3
32.519	6.711	304.0	1.0	1.0	2020-02-...	125	A	58	6.0NRT	272.7
31.8209...	6.231	314.8	1.0	1.0	2020-02-...	125	A	90	6.0NRT	274.9

Fig 37: Dataset and Resource Example

Here we can see the difference between dataset and resource. The dataset is on the left and it can belong to an organization and group. Every dataset can have associated more than one resource, which are the data files itself. Every dataset has also extra info like the source, the maintainer, the author, the time updated and when was created. The Resource is the data itself, it has a format, a description, a link where you can find it and previsualization. You can also download it directly from the page by clicking the go-to source button.



Add Organization

Search organizations...

6 organizations found   Order by: Name Ascending

NASA
The National Aeronautics and Space Administration is an independent agency of...
0 Datasets

National Oceanic and Atmospheric Administration
The National Oceanic and Atmospheric Administration is an American scientific...
0 Datasets

Open Weather
Openweather is a small IT company, established in 2014 by a group of...
0 Datasets

United States Geological Survey
The United States Geological Survey is a scientific agency of the United...
0 Datasets

The World Bank
The World Bank is an international financial institution that provides loans...
0 Datasets

Add Group

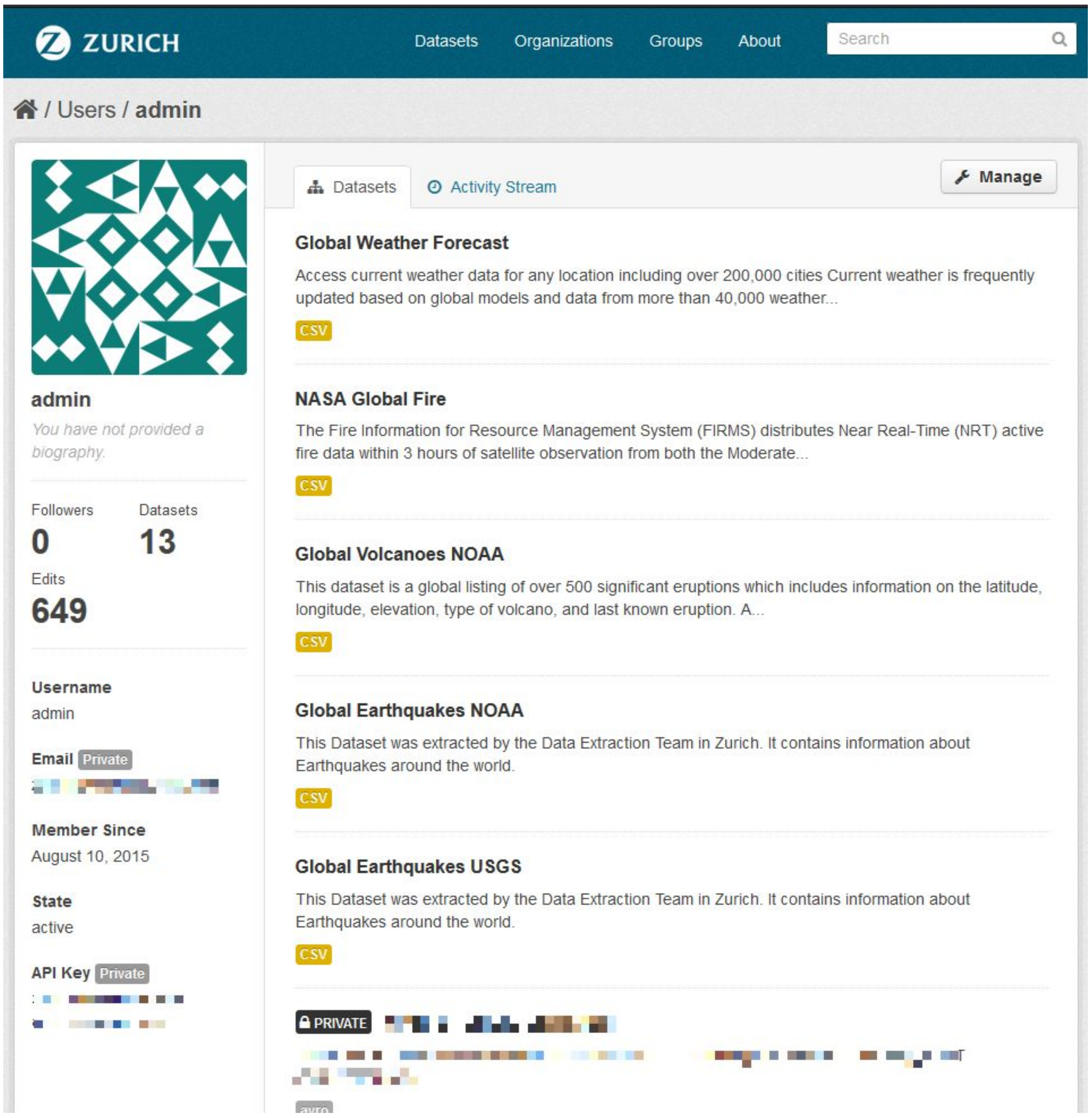
Search groups...

4 groups found   Order by: Name Ascending

Natural Disasters
In this group we can find all datasets related with Natural Disasters.

Fig 38: Organizations and Groups Example

This is how the organizations and groups are displayed. Some of them are pixelated since we were testing some functions involving private data. The governance system is involved in this part. If any of these groups or organizations are private they will not appear in the general search. We can see it because we are admins, but even admins will not find direct private data when browsing datasets in general. If we want to see any of this dataset we must enter into the organization itself and there we will find a searching page similar to the one in Figure 47 with all the private datasets.



admin

You have not provided a biography.

Followers 0
Datasets 13
Edits 649

Username
admin

Email Private
[redacted]

Member Since
August 10, 2015

State
active

API Key Private
[redacted]

Global Weather Forecast

Access current weather data for any location including over 200,000 cities Current weather is frequently updated based on global models and data from more than 40,000 weather...

CSV

NASA Global Fire

The Fire Information for Resource Management System (FIRMS) distributes Near Real-Time (NRT) active fire data within 3 hours of satellite observation from both the Moderate...

CSV

Global Volcanoes NOAA

This dataset is a global listing of over 500 significant eruptions which includes information on the latitude, longitude, elevation, type of volcano, and last known eruption. A...

CSV

Global Earthquakes NOAA

This Dataset was extracted by the Data Extraction Team in Zurich. It contains information about Earthquakes around the world.

CSV

Global Earthquakes USGS

This Dataset was extracted by the Data Extraction Team in Zurich. It contains information about Earthquakes around the world.

CSV

PRIVATE [redacted]

[redacted]

avro

Fig 39: User Example

Every user can track what datasets they manage and, depending on the user, has certain rights to edit, visualize or delete them. As we can see here, the admin user owns 13 datasets even though that on the homepage only appears 8. This is because of the private configuration that we mentioned before.

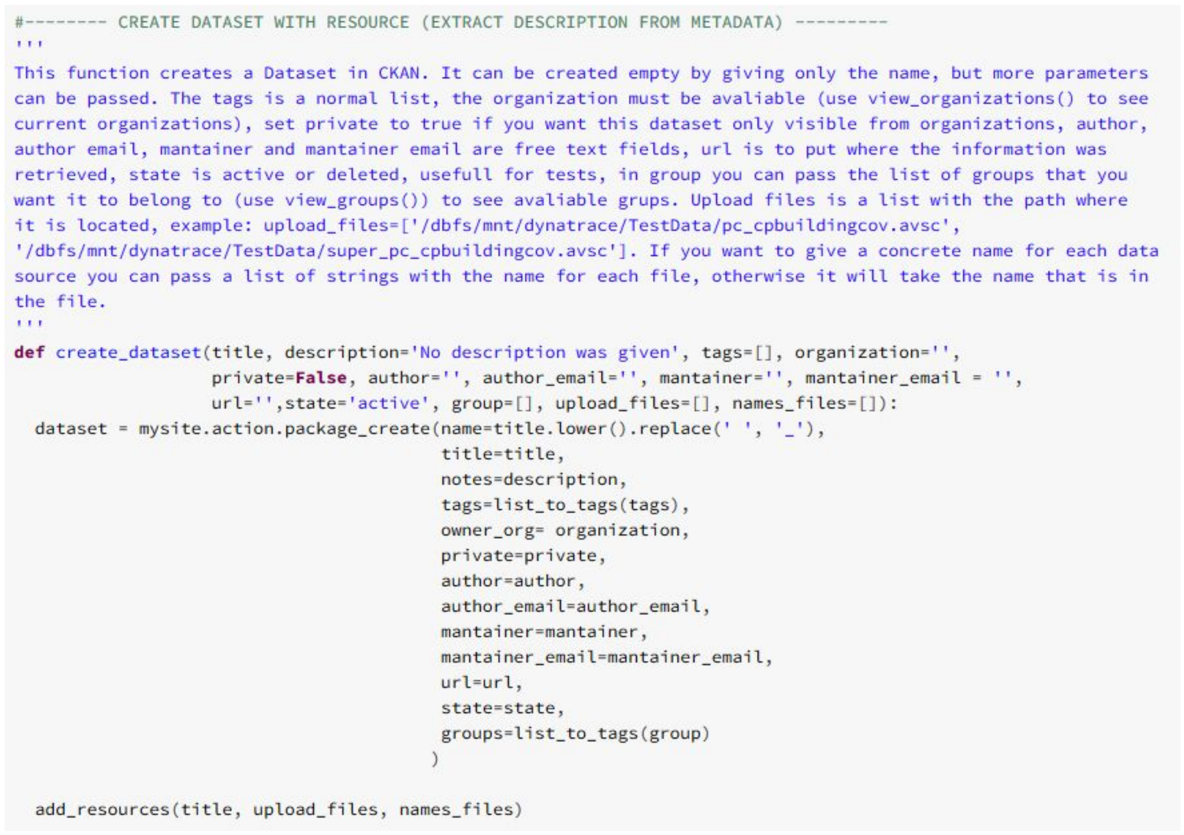
4.3 Uploading Data to CKAN

"In this section, we will see how we upload our Open Data to CKAN. To do this we have created a new Python Library based on the CKAN API."

It is great to have a portal that allows manual uploads but we want our data to get updated at the same time that it does in our ADLS. To do so we want that our data is automatically updated, it was one of the most important requisites that we had when choosing a portal.

We have developed a Python library to use the CKAN API intuitively. We have used an external library to support our version.

First of all, we need to log in with our admin credentials and indicate which site are we working with. Then we can start using the commands. In the API we can find how the page manages their content. It has what they call package that corresponds to our dataset. You can create an empty dataset, package, that only had a name, but from the API you can add much more information. We've created a function that uses the API call to simplify.



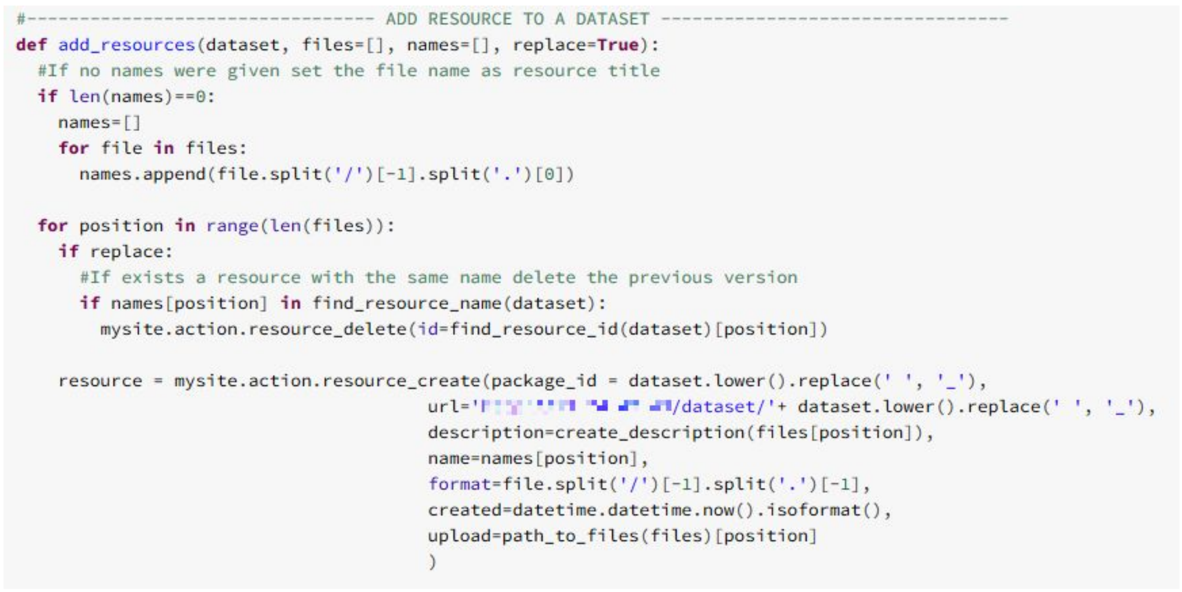
```
#-------- CREATE DATASET WITH RESOURCE (EXTRACT DESCRIPTION FROM METADATA) ---------
'''
This function creates a Dataset in CKAN. It can be created empty by giving only the name, but more parameters
can be passed. The tags is a normal list, the organization must be avaliable (use view_organizations() to see
current organizations), set private to true if you want this dataset only visible from organizations, author,
author email, mantainer and mantainer email are free text fields, url is to put where the information was
retrieved, state is active or deleted, usefull for tests, in group you can pass the list of groups that you
want it to belong to (use view_groups()) to see avaliable grups. Upload files is a list with the path where
it is located, example: upload_files=['/dbfs/mnt/dynatrace/TestData/pc_cpbuildingcov.avsc',
'/dbfs/mnt/dynatrace/TestData/super_pc_cpbuildingcov.avsc']. If you want to give a concrete name for each data
source you can pass a list of strings with the name for each file, otherwise it will take the name that is in
the file.
'''
def create_dataset(title, description='No description was given', tags=[], organization='',
                   private=False, author='', author_email='', mantainer='', mantainer_email = '',
                   url='',state='active', group=[], upload_files=[], names_files=[]):
  dataset = mysite.action.package_create(name=title.lower().replace(' ', '_'),
                                         title=title,
                                         notes=description,
                                         tags=list_to_tags(tags),
                                         owner_org= organization,
                                         private=private,
                                         author=author,
                                         author_email=author_email,
                                         mantainer=mantainer,
                                         mantainer_email=mantainer_email,
                                         url=url,
                                         state=state,
                                         groups=list_to_tags(group)
                                        )

  add_resources(title, upload_files, names_files)
```

Fig 40: Create Dataset Function

As we can see, using this function we do not need to worry about any parameter since it is preset to empty. We also handle other possible issues with the API. The labels and groups must be passed as a list of dictionaries with a concrete format. In this function, we only need to enter a normal list and we handle the transformation. There is an internal name for the dataset apart from the title itself, we managed it to be automatically set based on the given title. We can also pass a list with the path where our data is stored and it will be added as a resource. If we want to give a concrete name to our resource we can also pass them a list of names.

A resource is what we have called the data itself from the dataset. One dataset can have several resources but we will try to not have too many of them. The reason is that when we search on CKAN we will obtain as a result the datasets that match our parameters. As we explained, this search affects also the data in the datasets, the resource, so if we have too many resources in a single dataset we could worsen the search system.

E.g. Search for "fire" and find that a dataset of wildfires had more than 100 resources but the word fire only appears in one of them, now we need to find which one of this 100 is the one we are looking for.



```
#-------------------------------- ADD RESOURCE TO A DATASET --------------------------------
def add_resources(dataset, files=[], names=[], replace=True):
  #If no names were given set the file name as resource title
  if len(names)==0:
    names=[]
    for file in files:
      names.append(file.split('/')[-1].split('.')[0])

  for position in range(len(files)):
    if replace:
      #If exists a resource with the same name delete the previous version
      if names[position] in find_resource_name(dataset):
        mysite.action.resource_delete(id=find_resource_id(dataset)[position])

    resource = mysite.action.resource_create(package_id = dataset.lower().replace(' ', '_'),
                                  url='[redacted]/dataset/'+ dataset.lower().replace(' ', '_'),
                                  description=create_description(files[position]),
                                  name=names[position],
                                  format=file.split('/')[-1].split('.')[-1],
                                  created=datetime.datetime.now().isoformat(),
                                  upload=path_to_files(files)[position]
                                  )
```

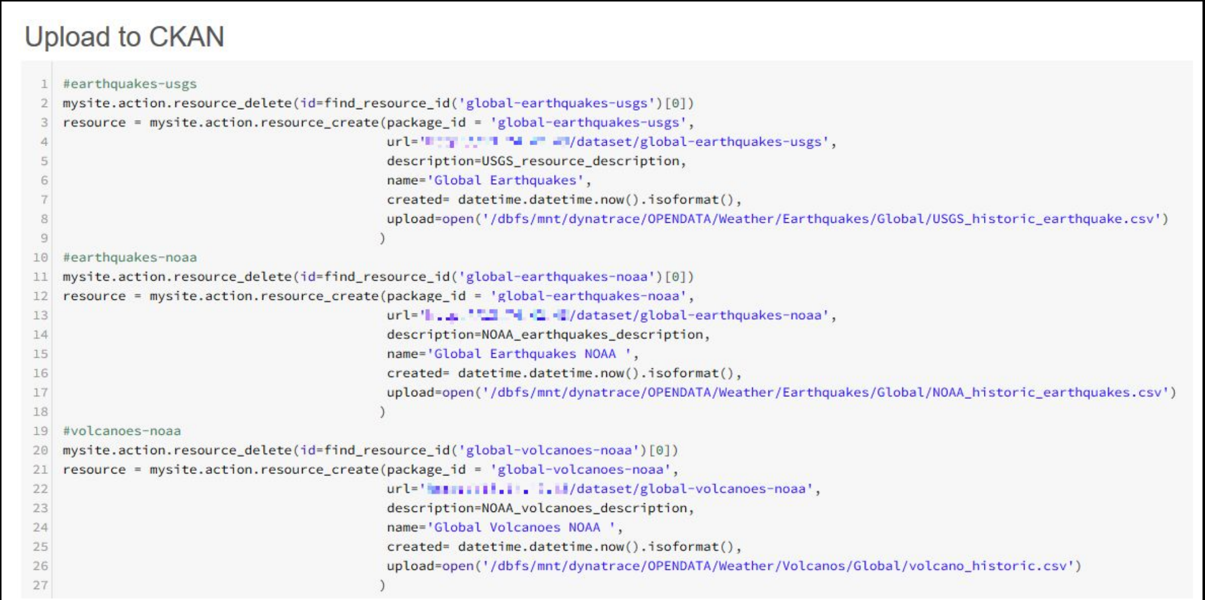
Fig 41: Add Resource to a Dataset

With this function, we handle several things. It can be called manually if someone needs to add some resources to any dataset but it is automatically called when we create a dataset with data attached. In this case, we see that if no names are given we will assign the file name. Also, the fact that if exist any resource with the same name we will delete the previous one. It is not forbidden to have two resources with the same name but it is usually not desirable. We had also to handle some issues when locating the resources and when deleting since the API does not provide an effective method to delete. It changes the state of the resource to *deleted* but it is still stored in the database, which causes some trouble later.

We created this little framework to simplify the process because, this project, more concretely the data catalog development, has drawn the attention of other departments. So we were required to optimize the process and make it understandable for developers and data scientists. In our case, we used a more basic and customized version since the library was not fully developed when we were asked to upload this data. We created the organizations from each dataset (NASA, OpenWeather, NOAA, and USGS) and the natural-disaster group. Since we will only update our datasets, not deleting or creating new, every time we extract data we will only regenerate the resource, not the whole dataset.

We have the create description function that automatically generates descriptions from the resource using the spark schema of the data. We have also a version when they want to update only metadata files, such as Avro schemas (.avsc). But for our case, we decided to handwrite this information to give a more human perspective of each column and what they mean. Note that this can also be done through the API.

The only change that we need to do to our ETL code is to upload the data to CKAN. We added a cell at the end of the notebook that only does this.



Upload to CKAN

```
#earthquakes-usgs
mysite.action.resource_delete(id=find_resource_id('global-earthquakes-usgs')[0])
resource = mysite.action.resource_create(package_id = 'global-earthquakes-usgs',
                                          url='[redacted]/dataset/global-earthquakes-usgs',
                                          description=USGS_resource_description,
                                          name='Global Earthquakes',
                                          created= datetime.datetime.now().isoformat(),
                                          upload=open('/dbfs/mnt/dynatrace/OPENDATA/Weather/Earthquakes/Global/USGS_historic_earthquake.csv')
                                         )
#earthquakes-noaa
mysite.action.resource_delete(id=find_resource_id('global-earthquakes-noaa')[0])
resource = mysite.action.resource_create(package_id = 'global-earthquakes-noaa',
                                          url='[redacted]/dataset/global-earthquakes-noaa',
                                          description=NOAA_earthquakes_description,
                                          name='Global Earthquakes NOAA ',
                                          created= datetime.datetime.now().isoformat(),
                                          upload=open('/dbfs/mnt/dynatrace/OPENDATA/Weather/Earthquakes/Global/NOAA_historic_earthquakes.csv')
                                         )
#volcanoes-noaa
mysite.action.resource_delete(id=find_resource_id('global-volcanoes-noaa')[0])
resource = mysite.action.resource_create(package_id = 'global-volcanoes-noaa',
                                          url='[redacted]/dataset/global-volcanoes-noaa',
                                          description=NOAA_volcanoes_description,
                                          name='Global Volcanoes NOAA ',
                                          created= datetime.datetime.now().isoformat(),
                                          upload=open('/dbfs/mnt/dynatrace/OPENDATA/Weather/Volcanos/Global/volcano_historic.csv')
                                         )
```

Fig 42: Code to upload Open Data to CKAN

We first have written the descriptions of each resource in markdown format. Since the description is always the same, it explains each column of the dataset, there is no need to update it as long as the format of the table does not change. We can see in *Figure 48* how does it look like.

This is not the only way to upload data from ADLS to CKAN. Since we ended up creating a library for the rest of the teams we also developed a method that allows scanning an already mounted ADLS (see 1.4). In this case, we input as a parameter the file path and we scan the whole Data Lake. If there is any supported type file it will be uploaded to our Data Catalog. This function is highly customizable, but by now it saves every data file in a separate dataset with only one source, it uses as name the name of the file itself and the file where it is contained. We put every file accessed from the root as a tag to facilitate the search. This method is tough to bulk several files at the same time.

We could use this method also to upload all our open data to CKAN since we have got stored our information in an ADLS file called Open Data. As a part of developing this library, we also created new functionalities such as deleting all data from an organization or locating files with given filters. E.g. locate datasets, modify datasets, modify the group of a dataset, add datasets to an organization, create and remove datasets and organizations, download data from resources, etc. In general, we have created more functions for more elaborate tasks but simplifying all the process.

5. Use case

"In this part, we will analyze all the Open Data datasets that we have stored in our Data Catalog and try to extract useful insights from them. We will use our library to access the data."

At this point, we have extracted the data from the source, transformed it to fit our made-up standards, and loaded it to a catalog that allows an optimal search. All this data management is great but we need to be able to get something from all this data to ensure that all that we have done is worth it.

To illustrate what we want to do, we will use the concept of the pyramid of knowledge also known as the DIKW pyramid. It stands for Data, Information, Knowledge, Wisdom. Right now we only have data, even though that it is labeled and has some description. We will consider that we have information when we have some sort of human-understandable concept from the data.

Our first approach will be to transform all the data into a simple sentence given a location. It means that we will input a location (longitude and latitude coordinates) and we will output a full report related to all the Natural disasters that we have been extracting.

To do so, we need to first extract the most useful insights from each table based on the location given. The first point is to reduce the rows of data in our dataset keeping only those that are more significant to us. In our case, we want to keep only those rows that hold information from our city, or in case that our city is not in the dataset the closest cities.

To select those rows we first take the given location point in longitude-latitude format and check if it is in our database. If we do not find the point we expand the radius of search applying some transformations to convert from coordinates to distance. We will increase this radius until we find a row that contains information from a city inside the area.

Once we find it we will extract from each dataset the most interesting, and human-readable, features and output them in a structure that will store this information as long as the radius that we had to search until we found it. In the cases where our city is between more than one city, we will make the average between them. In these cases, we should notify this in the output.

We did this for each extracted dataset. Then we created a function that given the coordinates apply the corresponding function for each dataset and output a human-readable report. In our example, we used the coordinates from Sarasota, Florida, USA. (Latitude: 27.3364, Longitude: -82.5308). Here what we get:

OUTPUT

```
Hurricanes: This area (Your city and 23 km around) has been visited 2 times by a
tropical Storm, 0 times by a Cyclone and 0 times by a Hurricane.
Volcanoes: You have 4 volcanoes near (1650 km). They last erupted on average the
year 1830.
Wildfires: You had 53 wildfires close (19 km around) with 351 brightness on
average and 76% confidence on average.
Earthquakes: The most dangerous area is 1011 km near and it has earthquakes of
magnitude 4.2 on average.
Weather: Using information from 1 closest cities (53 km): Highlands.
	Humidity: 94.0%
	Pressure: 1014.75hPa
	Wind Speed: 2.98m/s
	Minimum Temperature: 20.25ºC
	Temperature: 21.48ºC
	Maximum Temperature: 22.5ºC.
```

With this explanation of the data, the Data Science team could create a formula to evaluate the risk of every coordinate based on the parameters described in this text. E.g.

$$\text{Risk(Lat, Long)} = \frac{\textit{Number of disasters} * \textit{damage caused by the disaster}}{\textit{distance to the disaster}}$$

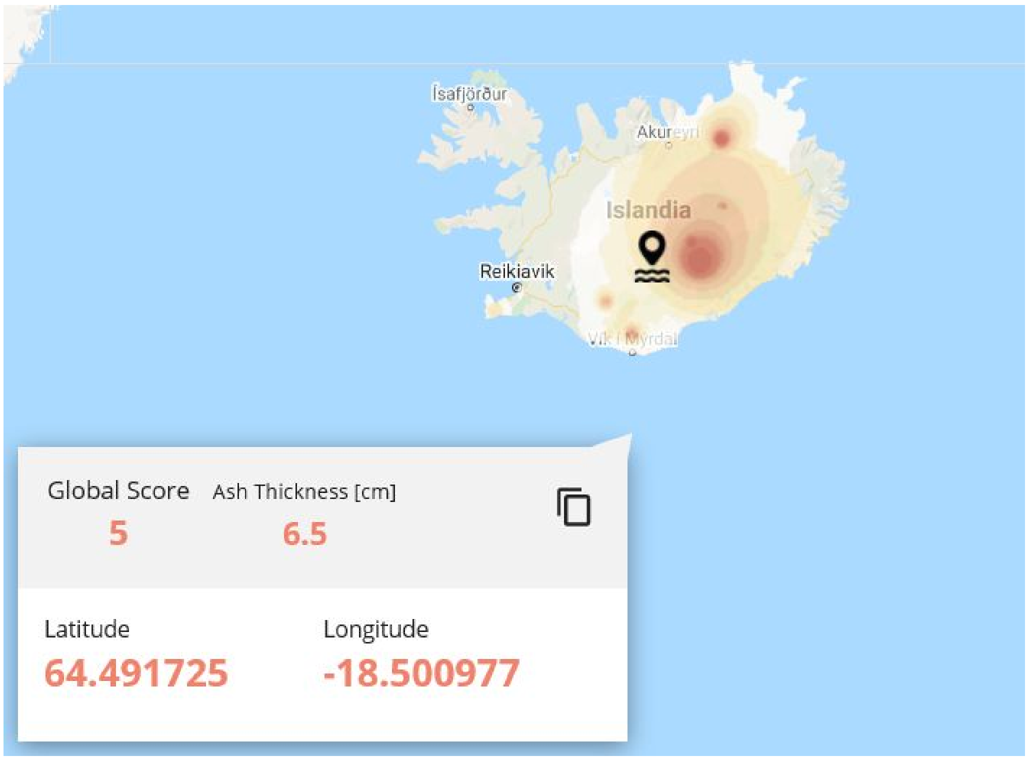
This is a simplification of a more complex formula that could be used in the future to determine the natural disaster risk related to a certain property. Once this formula is well defined it could be applied to all coordinates in a map, from -180 to 180 for longitude and from -90 to 90 for latitude, and plot the risk. So we could generate a map of the whole planet with the risk per zone in it.

6. Conclusions

"In this part, we will show the impact of the project. We will separate it in two categories, the usability of the Open Data extracted and the acceptance of the chosen CKAN Data Catalog inside the company."

6.1 Open Data

Currently, the preprocessed data is already being used in several projects. Despite the current version is very preliminar, a web-based interactive hazards map has been developed as an internal tool that uses the data. In addition, this information is also being used by a team that integrates it for risk assessment. We can thus conclude that the company is being benefited from the TFG in terms of Open Data usability.



Global Score 5
Ash Thickness [cm] 6.5
Latitude 64.491725
Longitude -18.500977

Figure 43: Risk in a concrete point shown in the map

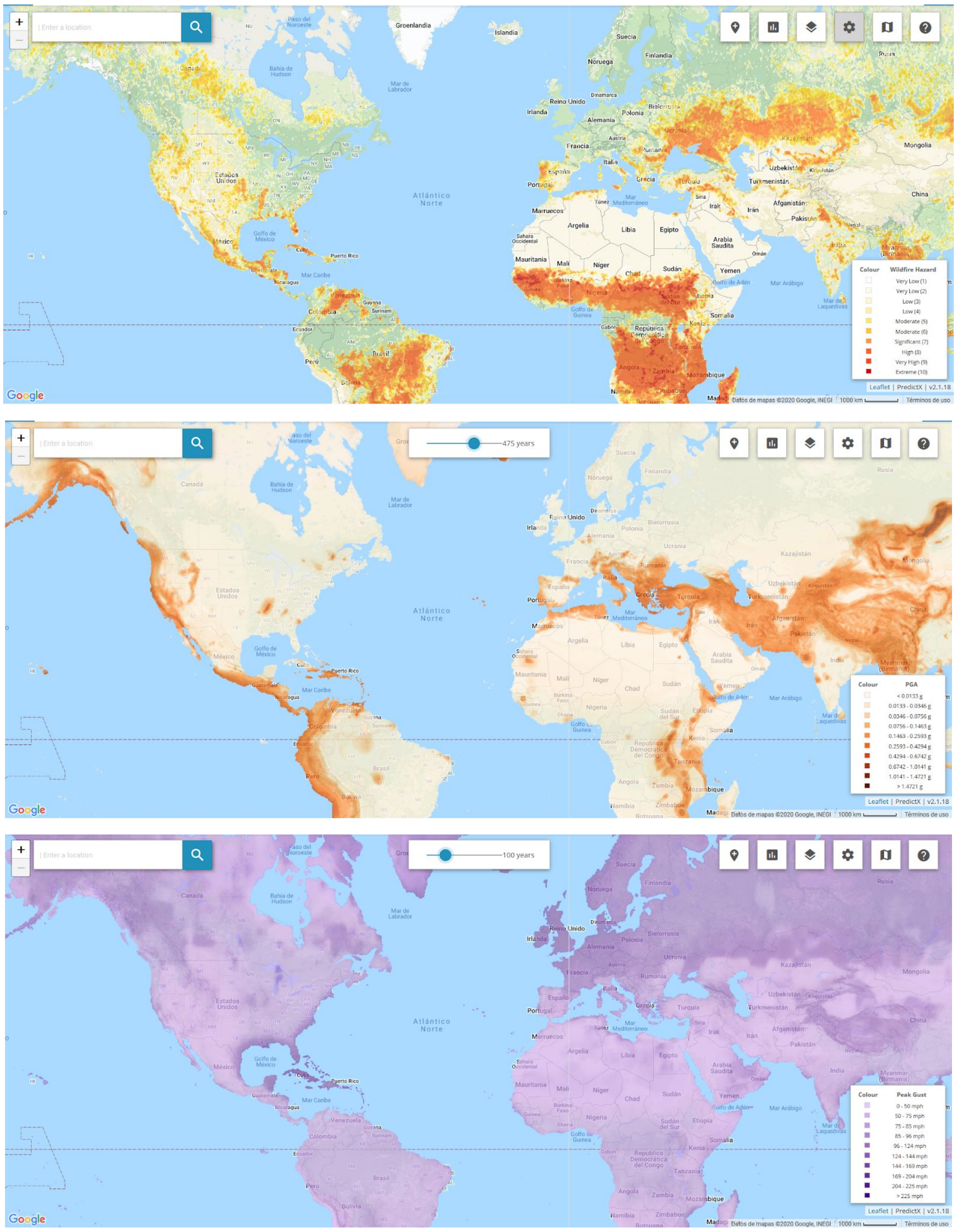
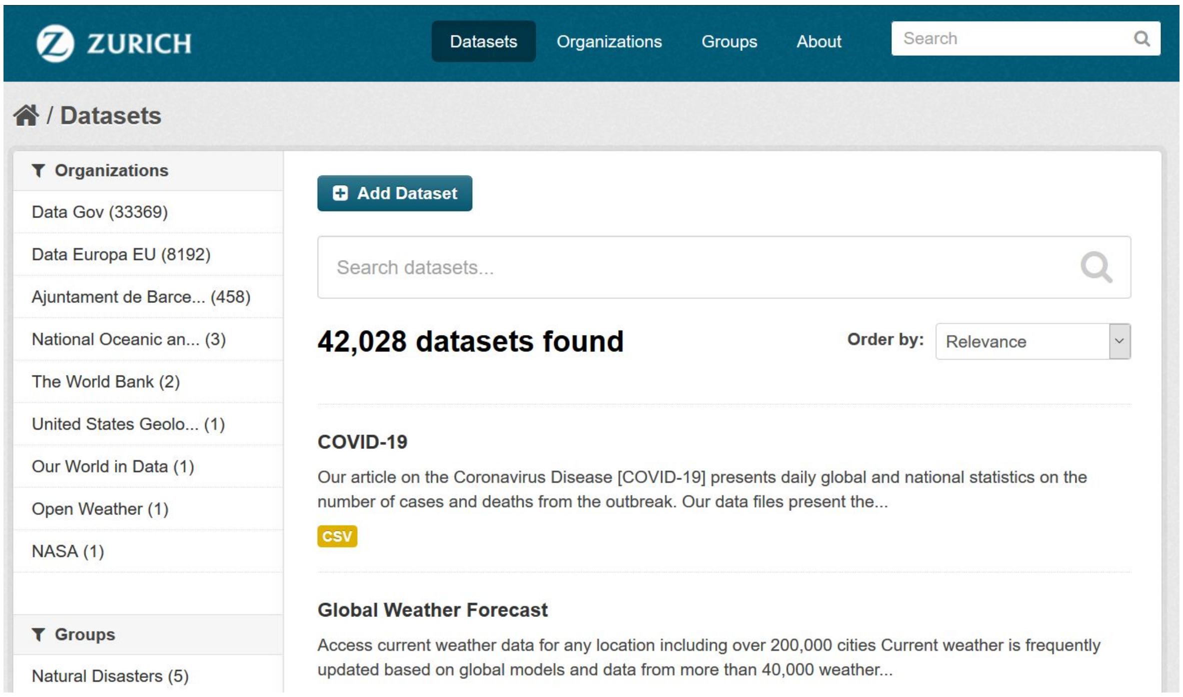


Figure 44: Wildfires, earthquakes and storms maps in the internal tool

6.2 Data Catalog

For the data catalog acceptance inside the company we can declare it as a success. We had a first version in the web (http://13.74.42.43/), now only used to test experimental features from our CKAN Library. This system really liked here and in Germany. They asks us to manage a private organization inside CKAN with part of their own private data. They also wanted to do a bulk upload to store all the information from their ADLS.



42,028 datasets found

COVID-19

Our article on the Coronavirus Disease [COVID-19] presents daily global and national statistics on the number of cases and deaths from the outbreak. Our data files present the...

Global Weather Forecast

Access current weather data for any location including over 200,000 cities Current weather is frequently updated based on global models and data from more than 40,000 weather...

Figure 45: State of the catalog one month later after release (Showing only public datasets)

We add this functionality to our library and we managed to successfully upload that data. Our CKAN Library was explained to the developers team inside Zurich and they are now working with it to manage and modify the CKAN structure.

They created a new version in their Zurich Intranet for private data. All this new version was created using only our library. New dominions were requested by Zurich to have some versions of our data catalog.

1. opendata.zurich.com
2. externaldata.zurich.com
3. data.zurich.com

Bibliography

[1] Insurance Information Institute. (2020). “Facts + Statistics: Global catastrophes.” Retrieved from: https://www.iii.org/fact-statistic/facts-statistics-global-catastrophes

[2] Bhageshpur, K. (2019, Nov 15). “Data Is The New Oil -- And That's A Good Thing”, *Forbes*. Retrieved from: https://www.forbes.com/sites/forbestechcouncil/2019/11/15/data-is-the-new-oil-and-thats-a-good-thing Retrieved on: 2020, May 12

[3] Marr, B. (2019, September 5). “How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read”, *Forbes.* Retrieved from https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#9fd6d5660ba9 Retrieved on: 2020, May 12

[4] Rakwal, A. (2012, April 9). “Create a Windows Azure Subscription”. Retrieved from https://blogs.msdn.microsoft.com/arunrakwal/2012/04/09/create-windows-azure-subscription

[5] Gillies, S. et al. (2007). “Shapely: manipulation and analysis of geometric objects”. *toblerity.org*. Retrieved from: https://github.com/Toblerity/Shapely

[6] Lloyd, Stuart P. "Least squares quantization in PCM." Information Theory, IEEE Transactions on 28.2 (1982): 129-137.

[7] Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O.,et al. (2011). *Scikit-learn: Machine learning in Python. Journal of Machine Learning Research*, 12(Oct), 2825–2830.

[8] Kaufman, L., & Rousseeuw, P. J. (2009). *Finding groups in data: an introduction to cluster analysis* (Vol. 344). John Wiley & Sons.