PROJECT DELIVERABLE REPORT

Project Title:

Advanced personalised, multi-scale computer models preventing osteoarthritis

SC1-PM-17-2017 - Personalised computer models and in-silico systems for well-being

Deliverable number	**6.4**
Deliverable title	Computational intelligence models development
Submission month of deliverable	42
Issuing partner	CETRI
Contributing partners	CERTH
Dissemination Level (PU/PP/RE/CO):	CO
Project coordinator	University of Nicosia (UNIC)
Tel:	**+357 22 841 528**
Fax:	**+357 22 357481**
Email:	**felekkis.k@unic.ac.cy& giannaki.c@unic.ac.cy**
Project web site address	www.oactive.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777159

Revision History

Version	Date	Responsible	Description/Remarks/Reason for changes
1.0	25/03/2021	CETRI	Initial draft of D6.4
2.0	28/03/2021	CERTH, UNIC	Second Draft of D6.4

Table of Contents

Table of Figures

1 Summary

The present document reports on the OACTIVE cloud platform that includes the web-enabled database (OACTIVE DB), the AI/ML cloud computing infrastructure (OACTIVE AI), and the APIs providing a secure connection, storage, and accessibility to all the data collected and generated in the OACTIVE project.

The key features of the OACTIVE platform are provided below:

- a robust and scalable implementation using a well-established platform (Amazon AWS).
- data import and export via dedicated web clients or APIs.
- data protection & GDPR compliance.
- different access levels and permissions.

One of the core components of the OACTIVE platform is the OACTIVE DB, where the primary clinical data and the outcomes of the research activity are stored. The database is based on PostgreSQL v.11 as it considered a well-supported and scalable Relational Database Management System (RDBMS). The database instance is hosted in the Amazon Relational Database Service (RDS). The detailed specification of the OACTIVE DB is reported in Section 2.1.

Once the data has been set up, it can be accessed via API or Web Clients, allowing its use in various other applications (e.g., S3 cloud Storage, Jupyter Lab, R Statistics, Python, Webpg). API access is secured by tokens that can only be issued from the Amazon AWS policy service. The API is the primary interaction method with the OACTIVE DB backend and includes various methods to act on the data. A complete API reference is available in Sections 2.2 to 2.3.

The AI/ML infrastructure is hosted in Amazon Linux 2 server hosted in Amazon's AWS Cloud Computing service EC2. Users can access AI/ML computing resources through web applications, such as Jupyter Lab[1]and MLFlow[2]. All web applications are running on a NGINX[3] web server, hosted on the same Cloud computing service. A detailed examination of the AI/ML infrastructure is provided in Sections 3.1 to 3.2.

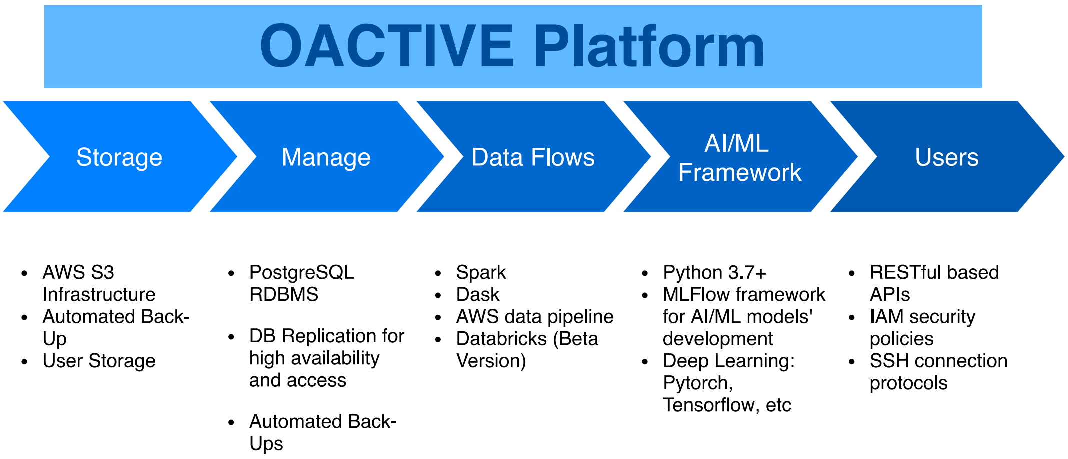
2 OACTIVE Cloud Platform Structure

2.1 Platform Overview

The OACTIVE platform provides a collection of services that allows project partners involved with the clinical trials to interact with the clinical data (raw, pre-processed data and metadata). All services provided by the OACTIVE platform is stored and managed on the AWS Cloud Infrastructure. A detailed view of the services implemented on AWS Cloud is presented in Fig. 1. The main advantages of Amazon's AWS Cloud services are:

1. **Scalability**: Allocate computes resources when it is needed.
2. **Accessibility**: RESTful based APIs allows users to access clinical data & data sources.
3. **Security**: Adopt Amazon's AWS security schemas.
    a. IAM policies and roles to grant access to users with different privileges and permissions.
    b. SSH secure based APIs for uploading and downloading data.

[1] https://jupyter.org
[2] https://mlflow.org
[3] https://www.nginx.com



OACTIVE Platform

Storage
- AWS S3 Infrastructure
- Automated Back-Up
- User Storage

Manage
- PostgreSQL RDBMS
- DB Replication for high availability and access
- Automated Back-Ups

Data Flows
- Spark
- Dask
- AWS data pipeline
- Databricks (Beta Version)

AI/ML Framework
- Python 3.7+
- MLFlow framework for AI/ML models' development
- Deep Learning: Pytorch, Tensorflow, etc

Users
- RESTful based APIs
- IAM security policies
- SSH connection protocols

Figure 1. Overview of services in OACTIVE cloud Platform

A detailed view of the OACTIVE's platform architecture is presented in Fig. 2. The OACTIVE platform infrastructure is based exclusively on Amazon's AWS cloud services. These services are:

- Relational Database Service, where the OACTIVE DB instance is hosted along with OACTIVE DB's replication.
- EC2 Computing instance, where the Web Server, the AI/ML services, and the MLFlow service are hosted.
- S3 Cloud storage, where the Metadata and the MRIs are stored.
- Amazon's API Gateway

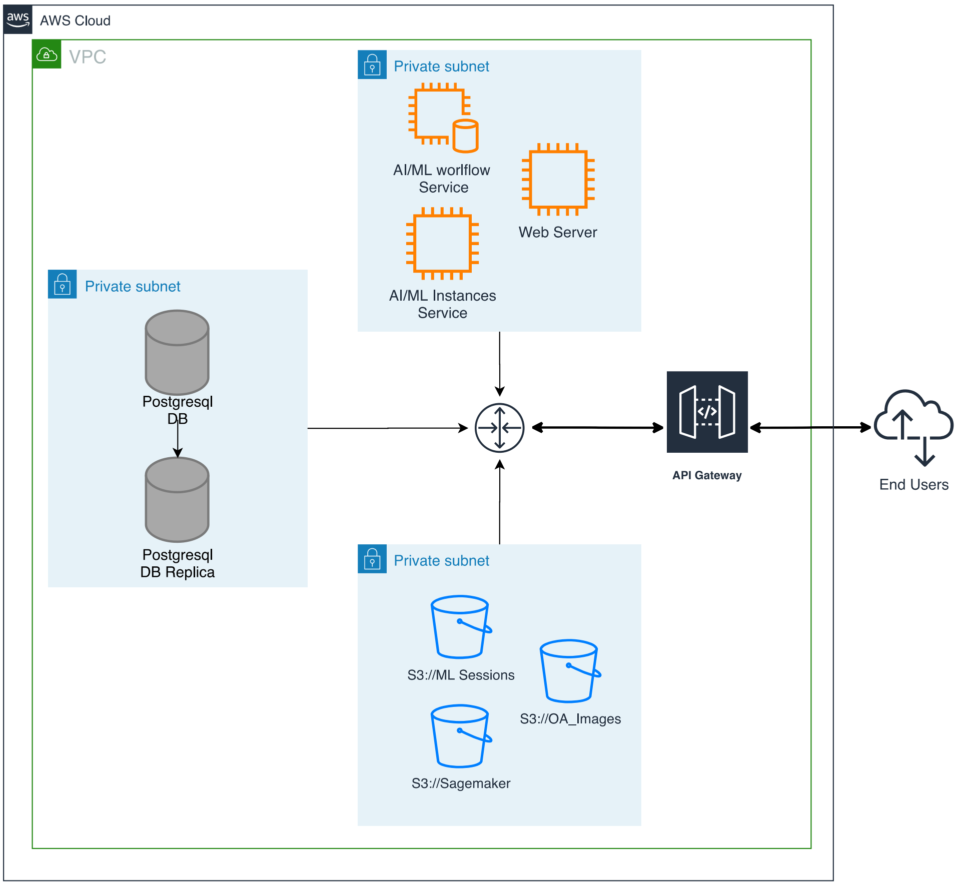


Figure 2. OACTIVE platform architecture

2.2 OACTIVE Database

Raw data from clinical trials and process data, along with their Metadata, are stored and organised through a relational database. PostgreSQL selected as a Relational Database Management System (RDBMS). RDBMS provide a logical data modelling schema that guarantees data availability, reliability, currency, consistency, flexibility, and efficiency. The main components of a logical data modelling schema are:

- Entities represent objects, things, and persons (e.g., Demographics)
- Attributes are the properties of entities (e.g., ID, Sex, Age, Occupation, Income, etc.)
- Relationships describe how entities related to each other.

A view of OACTIVE's relational database schema is presented in Fig. 3.

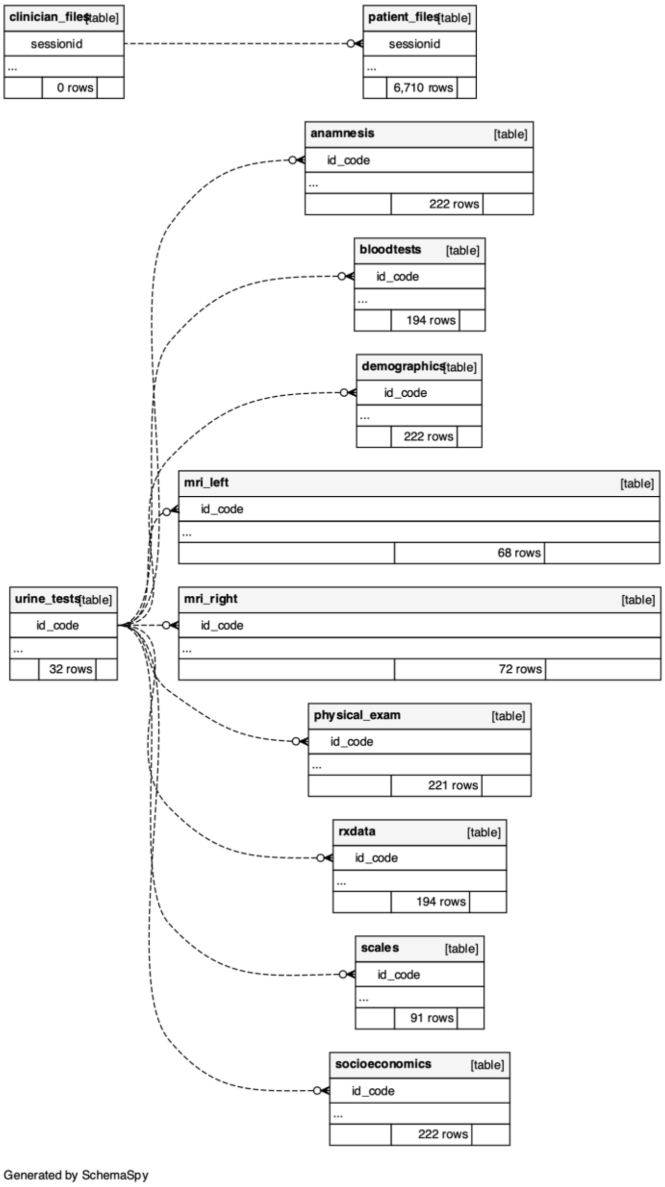
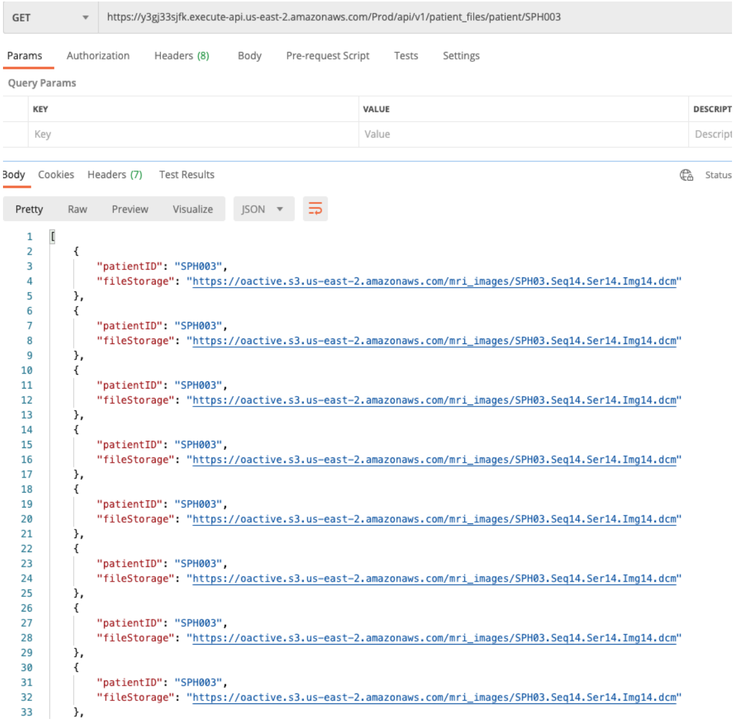


Figure 3. OACTIVE's Relational Database schema

Other data objects, such as images and Metadata, are stored into the Amazon Simple Storage Service (Amazon S3). The main advantages of Amazon S3 is that offers scalability, data availability, security, and instant access to the stored objects. Moreover, every data object can be associated with every instance in the OACTIVE database (Fig. 4). This functionality is crucial for the researchers as it can give instant access to patients' profiles.



GET https://y3gj33sjfk.execute-api.us-east-2.amazonaws.com/Prod/api/v1/patient_files/patient/SPH003

KEY	VALUE	DESCRIPT
Key	Value	Descript

```
[
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
    {
        "patientID": "SPH003",
        "fileStorage": "https://oactive.s3.us-east-2.amazonaws.com/mri_images/SPH03.Seq14.Ser14.Img14.dcm"
    },
```

Figure 4. Example of MRI objects in OACTIVE DB

2.3 Access Data via RESTful APIs

Application Programming Interfaces (APIs) are used to exchange data between services or between services and users, e.g., a user requests a chunk of data stored in a relational database or images from an S3 storage service. The OACTIVE APIs are based on RESTful architecture that defines various data handling methods, such as GET, POST, PATCH, and DELETE.

- Method GET to return a particular product (e.g., list of data from a database)
- Method PUT to update the full content of a particular product (e.g., upload new images on the S3 folder)
- Method PATCH to update a sub-content (set of fields) of a particular product (e.g., add new observations in database tables)
- Method DELETE to delete a particular product (e.g., remove images from a specific folder)

Lambda Function is an AWS service that provides an intermediate layer between RESTful APIs and AWS Services (e.g., PostgreSQL & S3 Storage). The Lambda layer scope is two-fold; first, it helps to integrate APIs and AWS services better, and second, it allows to incorporate AWS built-in access permissions. This AWS, built-in access permission, is managed through Identity and Access Management (IAM) security & policy schema. Thus, an API can connect to a specific AWS service and execute a RESTful method if the attached IAM policy permits it. API Gateway is an AWS service that facilitates developers creating, publishing, maintaining, and monitoring their APIs. In the adopted communication and security schema (Fig. 5), API Gateway's role is to connect APIs with IAM policies and Lambda functions and access specific users.

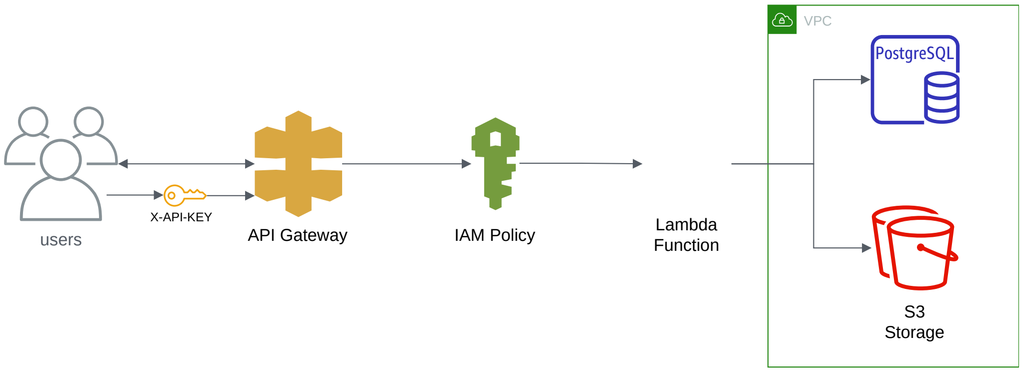
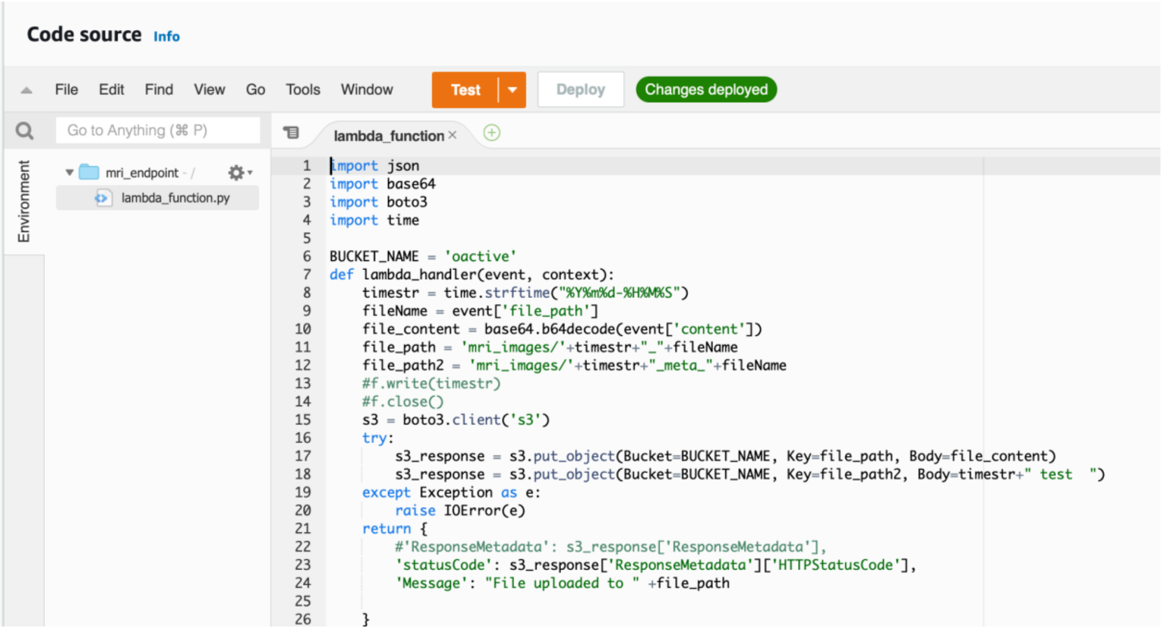


Figure 5. API Communication and Security schema

AWS Lambda layer is essentially a serverless compute service that manages APIs' calls to the OACTIVE platform. In Lambda service, APIs' backend can be written in Python or Java. A Python-based API that handles the upload functionality of the OACTIVE platform is presented in Fig. 6. The Python code, in this instance, manages the calls for the uploading of MRIs and their metadata in the S3 cloud storage.



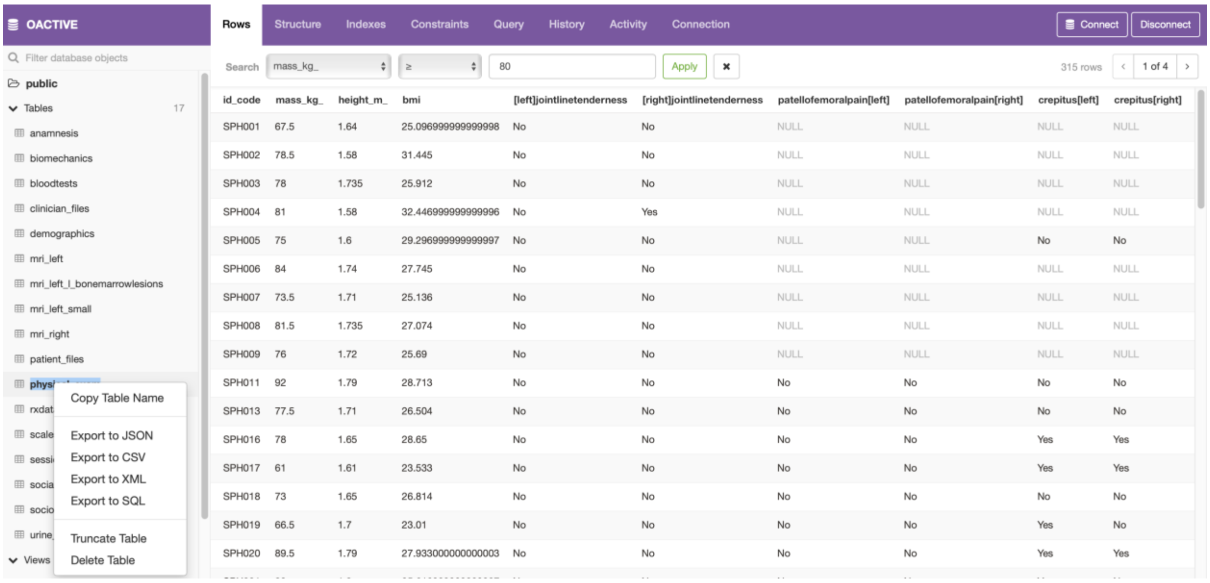
```python
import json
import base64
import boto3
import time

BUCKET_NAME = 'oactive'
def lambda_handler(event, context):
    timestr = time.strftime("%Y%m%d-%H%M%S")
    fileName = event['file_path']
    file_content = base64.b64decode(event['content'])
    file_path = 'mri_images/'+timestr+"_"+fileName
    file_path2 = 'mri_images/'+timestr+"_meta_"+fileName
    #f.write(timestr)
    #f.close()
    s3 = boto3.client('s3')
    try:
        s3_response = s3.put_object(Bucket=BUCKET_NAME, Key=file_path, Body=file_content)
        s3_response = s3.put_object(Bucket=BUCKET_NAME, Key=file_path2, Body=timestr+" test  ")
    except Exception as e:
        raise IOError(e)
    return {
        #'ResponseMetadata': s3_response['ResponseMetadata'],
        'statusCode': s3_response['ResponseMetadata']['HTTPStatusCode'],
        'Message': "File uploaded to " +file_path

    }
```

Figure 6. Lambda function API Backend

2.4 Access Data via Web Client

PgWeb[4] is an open-source web-based database client that allows users to explore and interact with the OACTIVE DB. The PgWeb is running locally on the user's PC, providing a simple and easy to operate GUI. The connection to the DB is made using the secure protocol SSH, along with user's credentials. Among the functionality provided by the PgWeb is the ability to download every DB table separately from the user. The downloaded files can be saved in four formats, CSV, XML, JSON, and SQL (Fig. 7).

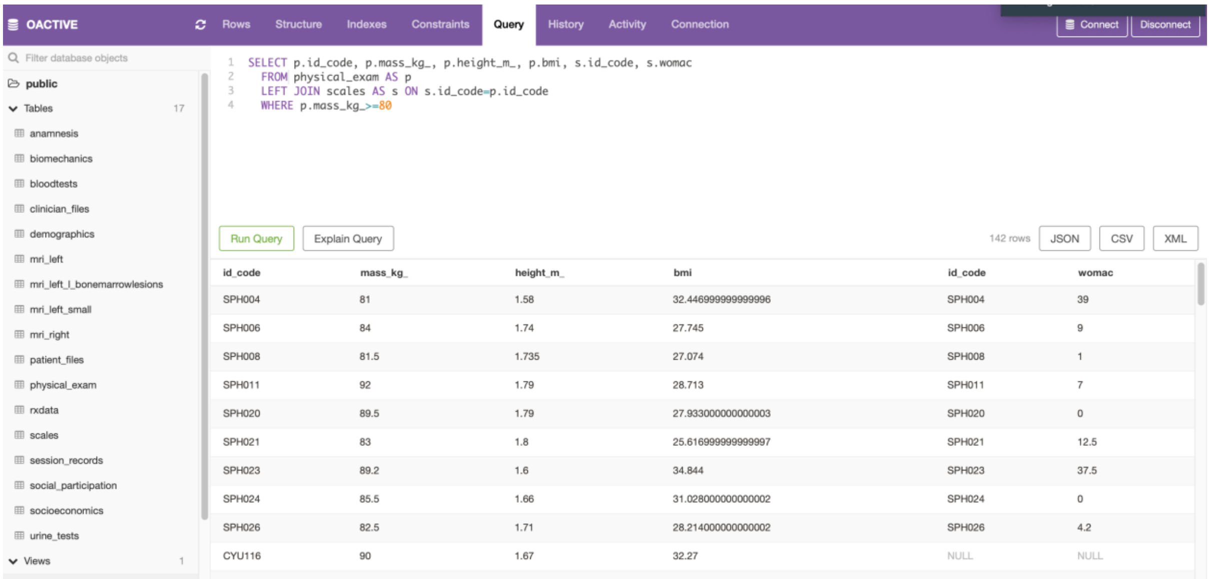
[4] https://sosedoff.github.io/pgweb/



id_code	mass_kg_	height_m_	bmi	[left]jointlinetenderness	[right]jointlinetenderness	patellofemoralpain[left]	patellofemoralpain[right]	crepitus[left]	crepitus[right]
SPH001	67.5	1.64	25.096999999999998	No	No	NULL	NULL	NULL	NULL
SPH002	78.5	1.58	31.445	No	No	NULL	NULL	NULL	NULL
SPH003	78	1.735	25.912	No	No	NULL	NULL	NULL	NULL
SPH004	81	1.58	32.446999999999996	No	Yes	NULL	NULL	NULL	NULL
SPH005	75	1.6	29.296999999999997	No	No	NULL	NULL	No	No
SPH006	84	1.74	27.745	No	No	NULL	NULL	NULL	NULL
SPH007	73.5	1.71	25.136	No	No	NULL	NULL	NULL	NULL
SPH008	81.5	1.735	27.074	No	No	NULL	NULL	NULL	NULL
SPH009	76	1.72	25.69	No	No	NULL	NULL	NULL	NULL
SPH011	92	1.79	28.713	No	No	No	No	No	No
SPH013	77.5	1.71	26.504	No	No	No	No	No	No
SPH016	78	1.65	28.65	No	No	No	No	Yes	Yes
SPH017	61	1.61	23.533	No	No	No	No	Yes	Yes
SPH018	73	1.65	26.814	No	No	No	No	No	No
SPH019	66.5	1.7	23.01	No	No	No	No	Yes	No
SPH020	89.5	1.79	27.933000000000003	No	No	No	No	Yes	Yes

Figure 7. PgWeb GUI and functionality

For advanced filtering options, PgWeb provides an SQL editor where user can type complex SQL queries to view and download filtered portions of the data stored into the database (Fig. 8).



```
SELECT p.id_code, p.mass_kg_, p.height_m_, p.bmi, s.id_code, s.womac
  FROM physical_exam AS p
  LEFT JOIN scales AS s ON s.id_code=p.id_code
  WHERE p.mass_kg_>=80
```

id_code	mass_kg_	height_m_	bmi	id_code	womac
SPH004	81	1.58	32.446999999999996	SPH004	39
SPH006	84	1.74	27.745	SPH006	9
SPH008	81.5	1.735	27.074	SPH008	1
SPH011	92	1.79	28.713	SPH011	7
SPH020	89.5	1.79	27.933000000000003	SPH020	0
SPH021	83	1.8	25.616999999999997	SPH021	12.5
SPH023	89.2	1.6	34.844	SPH023	37.5
SPH024	85.5	1.66	31.028000000000002	SPH024	0
SPH026	82.5	1.71	28.214000000000002	SPH026	4.2
CYU116	90	1.67	32.27	NULL	NULL

Figure 8. PgWeb SQL query editor

2.5 Access Management Levels

To enhance the security in the OACTIVE platform, five different access layers with a different set of permissions have been defined. The scope of each access layer is described below:

Administrator: Project administrators can perform any operation in the project, including removing the project and managing other users' roles. At least one user in a project must have the role of administrator.

Maintainer: Maintainers have complete control of DB configuration and data within the project. They can create or remove tables in the OACTIVE DB and change any entity's properties in the DB.

Operator: Operators can perform data operations on the items in a project. This will include updating and exporting data from DB.

Observer: The observer role allows someone to view project items and data but not change anything.

None: Changing a user's role to None will remove them from the project.

3 AI/ML Cloud Infrastructure

3.1 AI/ML Framework Overview

The primary function of OACTIVE AI/ML framework is to help AI/ML researchers and developers to develop, validate and put in production AI/ML models. The AI/ML framework provides a toolchain to accomplish these steps. The main components of the AI/ML framework are:

- **Data pipelines**: Collect data, cleaning data, standardize, feed AI/ML models, evaluate results.
- **Developing environments**: Python, R Statistics, Jupyter Lab.
- **AI/ML tools**: PyTorch, TensorFlow, Mxnet, CUDA.
- **MLOps**: MLFlow library for AI/ML for managing developing cycles.

The OACTIVE platform supports a three-phase developing cycle schema (Fig. 9). The first phase of the developing schema is focused on preparation of the data and in testing of different AI/ML models. The second phase of the development is devoted in extended out-of-sample test and integration tests with third-party applications. The third phase of the developing cycle is the last step before the application goes into production.

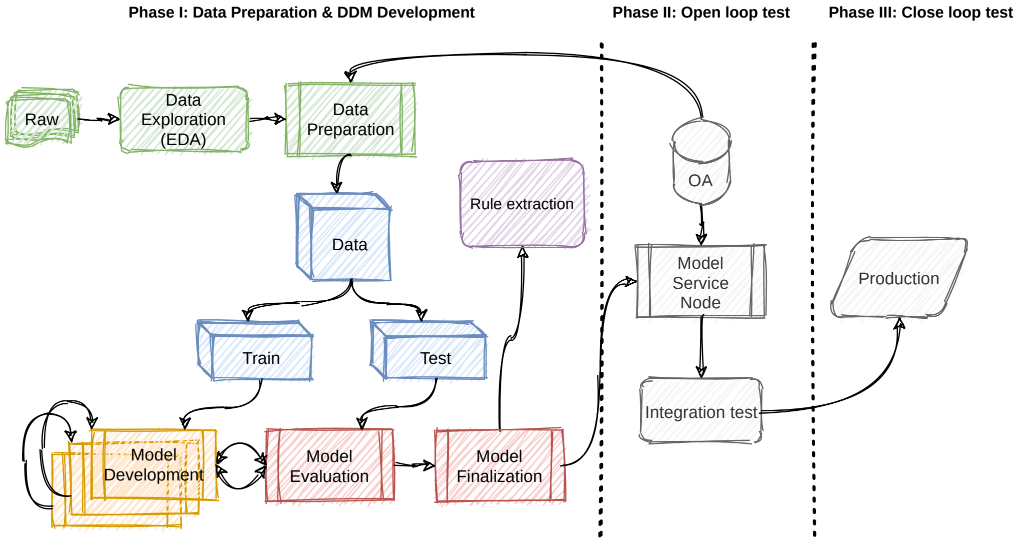


Figure 9. OACTIVE AI/ML developing schema

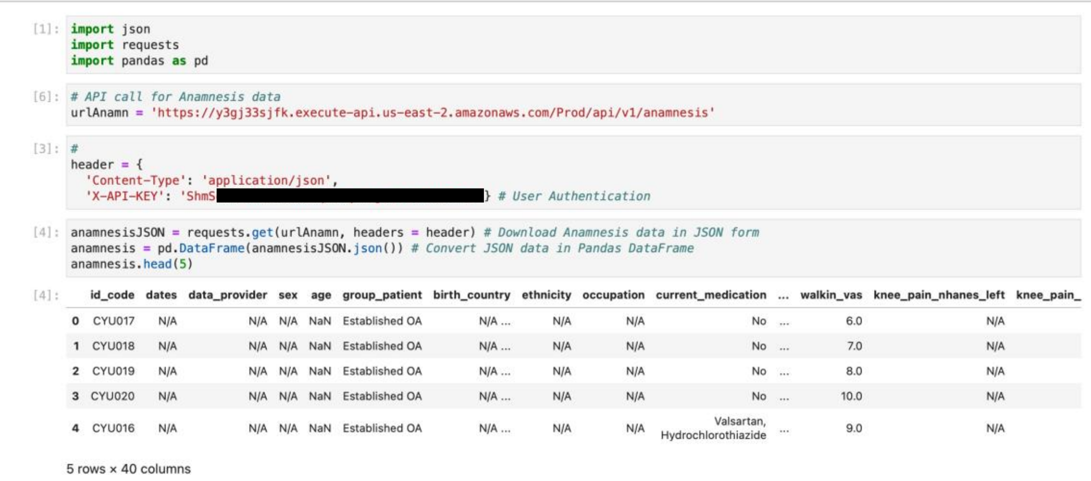
3.2 Data Pipelines

Data pipelines is a basic procedure in the developing of the AI/ML models and includes various methods adequate for different implementations. OACTIVE AI/ML framework offers various pre-installed frameworks and software packages to help researchers and developers to create their own data pipelines based on their needs. Databricks[5] and Apache Airflow[6], and Apache Spark[7] are well-established frameworks for End-to-End development of AI/ML applications.

Besides the pre-installed frameworks in the EC2 Linux instance, the OACTIVE AI/ML framework is equipped with numerous Python environments for tailored data pipelines. Pandas and Numpy python packages are only some of the libraries that can help researchers to build their own data pipeline

[5] https://databricks.com
[6] https://airflow.apache.org
[7] https://spark.apache.org

procedures. Furthermore, researchers and developers can use their user credentials to directly fetch data from OACTIVE DB into the Python environment (Fig. 10).



```
[1]: import json
     import requests
     import pandas as pd

[6]: # API call for Anamnesis data
     urlAnamn = 'https://y3gj33sjfk.execute-api.us-east-2.amazonaws.com/Prod/api/v1/anamnesis'

[3]: #
     header = {
       'Content-Type': 'application/json',
       'X-API-KEY': 'ShmS[redacted]} # User Authentication

[4]: anamnesisJSON = requests.get(urlAnamn, headers = header) # Download Anamnesis data in JSON form
     anamnesis = pd.DataFrame(anamnesisJSON.json()) # Convert JSON data in Pandas DataFrame
     anamnesis.head(5)
```

[4]:

	id_code	dates	data_provider	sex	age	group_patient	birth_country	ethnicity	occupation	current_medication	...	walkin_vas	knee_pain_nhanes_left	knee_pain_
0	CYU017	N/A	N/A	N/A	NaN	Established OA	N/A ...	N/A	N/A	No	...	6.0	N/A	
1	CYU018	N/A	N/A	N/A	NaN	Established OA	N/A ...	N/A	N/A	No	...	7.0	N/A	
2	CYU019	N/A	N/A	N/A	NaN	Established OA	N/A ...	N/A	N/A	No	...	8.0	N/A	
3	CYU020	N/A	N/A	N/A	NaN	Established OA	N/A ...	N/A	N/A	No	...	10.0	N/A	
4	CYU016	N/A	N/A	N/A	NaN	Established OA	N/A ...	N/A	N/A	Valsartan, Hydrochlorothiazide	...	9.0	N/A	

5 rows × 40 columns

Figure 10. Python API for OACTIVE DB

3.3 Web-based Integrated Development Environment

The main web Integrated Development Environment (IDE) provided in the OACTIVE AI/ML framework is the Jupyter[8] Lab which offers an easy-to-use programming environment for different programming languages (e.g., Python and R Statistics). The main advantage of Jupyter Lab is that can be accessed both from the Web or by running a local Jupyter Lab (Fig. 11) instance in a PC. For more demanding developer tasks, researcher and developers can connect their personal choice IDE (e.g., VSCode) to the Python instance through a SSH connection.

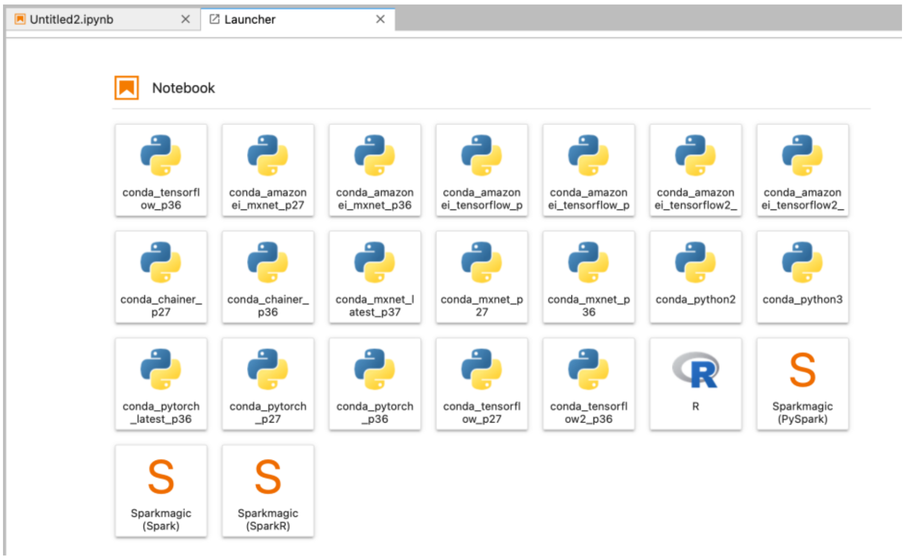


Figure 11. Jupyter Lab IDE Environment

[8] https://jupyter.org/about

3.4 AI/ML tools

The OACTIVE AI/ML framework offers to researchers and developers a vast variety of AI/ML libraries and frameworks. The list of these libraries and frameworks are presented below:

- Python 3.6+ programming environment
    - AI/ML Python packages:
        - Numpy 1.19
        - Pandas 1.1.4
        - Sklearn 0.23.2
        - PySpark 3.1
        - Scipy 1.5.4
        - Tensorflow 2.2
        - Pytorch 1.7.0
        - CUDA 9.2
        - Mxnet 1.7.0
- R Statistics 4.0.3
- Databricks
- Spark
- Apache Airflow

3.5 AI/ML workflows

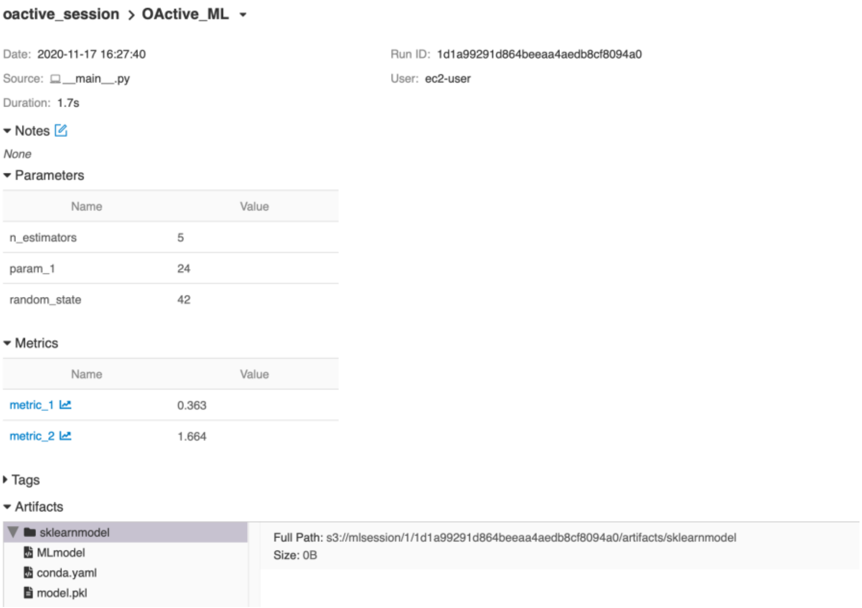
OACTIVE AI/ML framework provides MLflow[9] suite to assist researchers to implement the three-phase developing cycles, as introduced in Section 3.1. The Mlflow suite is hosted in the EC2 Amazon Linux 2 instance and can be accessed through the web by using username and password.

The methods that MLflow imports into the development workflow of the AI/ML models are:

- A tracking API and GUI for logging parameters, code versions, metrics, and artifacts.
- Project creation method that defines programming environment, metadata, input data, and packages or libraries used to the project.
- Model methods offer a convention for packaging machine learning models in multiple flavors, and a variety of tools for deployment.
- Registry offers a centralized model store, set of APIs, and GUI, to collaboratively manage the full lifecycle of an MLflow Model.

An example of using MLflow in OACTIVE is presented in Fig. 12 & 13, where two Support Vector Machine (SVM) models are tested.

[9] https://mlflow.org/docs/latest/concepts.html



oactive_session > OActive_ML

Date: 2020-11-17 16:27:40

Source: __main__.py

Duration: 1.7s

Run ID: 1d1a99291d864beeaa4aedb8cf8094a0

User: ec2-user

Notes

None

Parameters

Name	Value
n_estimators	5
param_1	24
random_state	42

Metrics

Name	Value
metric_1	0.363
metric_2	1.664

Tags

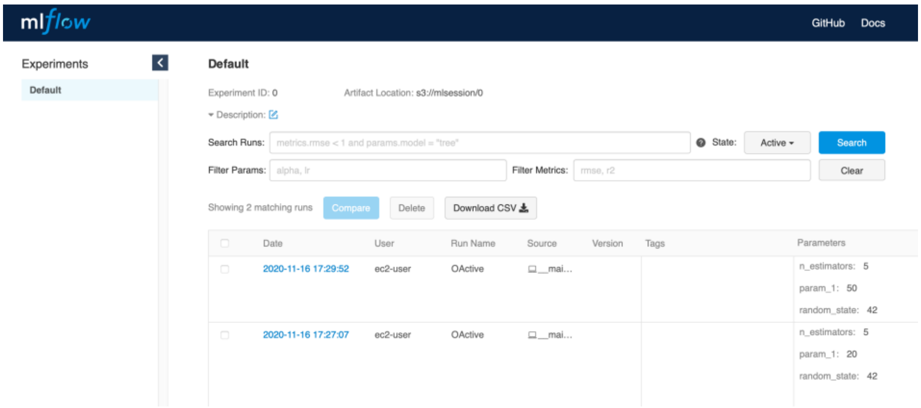
Artifacts

- sklearnmodel
  - MLmodel
  - conda.yaml
  - model.pkl

Full Path: s3://mlsession/1/1d1a99291d864beeaa4aedb8cf8094a0/artifacts/sklearnmodel

Size: 0B

Figure 12. MLflow model definition and settings



Default

Experiment ID: 0

Artifact Location: s3://mlsession/0

Description:

Search Runs: metrics.rmse < 1 and params.model = "tree"

State: Active

Filter Params: alpha, lr

Filter Metrics: rmse, r2

Showing 2 matching runs

□	Date	User	Run Name	Source	Version	Tags	Parameters
□	2020-11-16 17:29:52	ec2-user	OActive	__mai...			n_estimators: 5 param_1: 50 random_state: 42
□	2020-11-16 17:27:07	ec2-user	OActive	__mai...			n_estimators: 5 param_1: 20 random_state: 42

Figure 13. MLflow project definition