MIP: Advanced Data Processing and Analytics for Science and Medicine

Kostas Filippopolitis⋆, Yannis Foufoulas⋆, Minos Garofalakis⋆, Apostolos Glenis⋆, Yannis Ioannidis⋆, Thanasis-Michail Karampatsis⋆, Maria-Olympia Katsouli⋆, Evdokia Mailli⋆, Asimakis Papageorgiou-Mariglis⋆, Giorgos Papanikos⋆, George Pikramenos⋆, Jason Sakellariou⋆, Alkis Simitsis⋆, Pauline Ducouret⋄, Philippe Ryvlin⋄, Manuel-Guy Spuhler⋄ [*]

⋆Athena Research Center, Athens, Greece

⋄Centre Hospitalier Universitaire Vaudois (CHUV), Lausanne, Switzerland

ABSTRACT

We present the Medical Informatics Platform (MIP), an online collaborative platform for the scientific and medical community. It federates de-centralized patient data located in hospitals, helping clinicians, clinical scientists, and researchers identify patterns unique to diseases and provide clear diagnoses and personalized treatments. The platform enables users to access harmonized medical data from pre-processed neurophysiological and medical records, and research cohort datasets without the need to transfer original clinical data. This functionality facilitates exploration and analysis of medical data while preserving the privacy and security of sensitive patient information. The MIP blends data science and machine learning with data technology, and especially data integration, secure computation, decentralized distributed query execution, and low level, efficient execution of science pipelines exploiting features of modern data engines such as vectorization, parallelization, and JIT compilation. The MIP is the result of a multi-disciplinary, multi-year effort among computer scientists, clinical scientists and medical professionals. To date, it has been deployed and used in 40+ hospitals across Europe and another 12 installations are in process.

1 INTRODUCTION

The Human Brain Project (HBP) is one of the European Future and Emerging Technologies (FET) Flagships (to date: 123 partners, 600M Euros funding). It is a long-term and large-scale research initiative that pioneers digital brain research. It aims to gain an in-depth understanding of the complex structure and function of the human brain with a unique interdisciplinary approach at the interface of neuroscience and data technology. HBP scientists employ highly advanced methods from computing, neuroinformatics, and artificial intelligence to carry out cutting-edge brain research. The acquired knowledge is translated into novel applications in medicine and technological advances. The data technology component of HBP that supports HBP scientists in their day-to-day analysis is fueled by the Medical Informatics Platform (MIP).

In this paper, we present the design, architecture, and functionality of the MIP. The MIP is a privacy preserving, federated data analytics software that connects patient anonymized data from hospitals and research cohort datasets without moving them from their original storage. It provides a set of pre-integrated statistical methods as well as predictive machine learning algorithms for patient anonymized data exploration, data modelling, integration and execution of experiments (data analysis methods). It helps investigate and compare harmonized medical data extracted from pre-processed neurophysiological and medical records. It also offers the least invasive approach to connect centers and datasets; it promotes access to augmented knowledge potential through collaborative research. The key design principles of MIP include:

- Data is collected by the hospitals. It is stored on their servers and never leaves the hospital.
- Each hospital keeps its data within its secured server, and the MIP is installed on this server.
- Each analysis is ran by algorithms installed on the server, inside the hospital.
- Only aggregated, encrypted data leaves the hospital.
- The databases are not explorable by users.

Our solution and value proposition includes a new system architecture for decentralized and secure computations, leveraging the efficiency of modern data engines to boost analytics with features such as vectorization, zero-cost copy, data serialization, parallelization, and JIT compilation. The MIP was built from the ground up in synergy with state-of-the-art security and cryptographic technologies to operate on encrypted data without a mediator, ensuring that the data never leaves its location.

Deployment. The MIP is gradually engaging an increasing number of medical centers in Europe, involving thousands of cases with neurological and psychiatric disorders. Current MIP users include CHUV, Fatebenefratelli Brescia, CHRU Lille, Niguarda Ospedale, Mario Negri Institute Bergamo, Karolinska Institute, and Stockholm. The MIP is privacy preserving and GDPR compliant, and currently, its technology readiness level (TRL [12]) is 8.

Example use case: "Federated analyses in Alzheimer's disease". In this scenario, the MIP combines data from memory clinics in Brescia (1960 patients), Lausanne (1032 patients), and Lille (1103 patients), as well as the reference dataset ADNI (1066 patients). The data remains in the respective hospitals but the analysis is performed on the overall caseload. The objectives of the case study are: (a) determine how the brain volumes contribute to diagnosis, (b) increase of diagnosis specificity by introducing 2 key AD biomarkers, Amyloid beta 1-42 and p-Tau, and (c) influence of 2 non-AD etiologies: depression (PSY), vascular damage to cerebral white matter (VA). The method analyzes MRIs from patients utilizing individual T1w images, non-linear registration, neuromorphometric atlas, Shoot-SPM12 Matlab toolbox. The scientific analysis unravel observations related to (a) brain volume repartition across diagnosis, (b) clusters on $A\beta_{42}$, pTau and left entorhinal volume, and (c) influence of 2 non-AD etiologies (see also [11]). This scientific analysis leverages two algorithms provided by MIP: k-means and linear regression.

[*]Equal contribution of the authors, each within their role. The authors are listed alphabetically per institution.

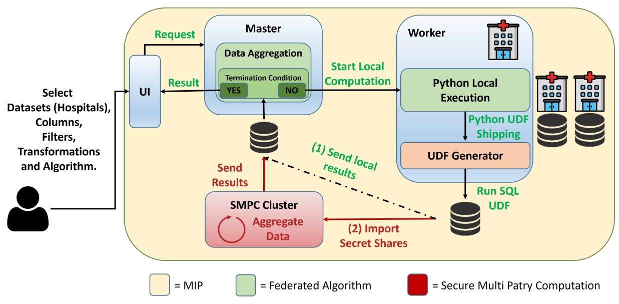


Figure 1: MIP architecture

```
def fit(self, X, y):
  local_transfers = self.local_run(
     func=fit_local,
     keyword_args="x": X, "y": y,
     share_to_global=[True],
   )
   self.global_state, global_transfer = self.global_run(
     func=fit_global,
     keyword_args=dict(local_transfers=local_transfers),
     share_to_locals=[False, False],
   )
   global_transfer_data = get_transfer_data(global_transfer)
   self.coefficients = global_transfer_data["coefficients"]
```

Figure 2: Linear regression fit in MIP (abridged)

2 DEMONSTRABLE FEATURES

A scientist initiates an experiment through interaction with the User Interface (UI). The experiment is sent to the Master node, which determines how the algorithms involved will be executed in a federated fashion. Then, local computation steps start on the Worker nodes, where the clinical data is located. Aggregated results are pushed back to the Master node following a SMPC protocol. The process is iterative and when is done, the analysis results are presented to the scientist via the UI. Figure 1 shows a high-level architecture of MIP. Next, we detail the data and compute flows in MIP.

Federated Algorithm. We write a federated algorithm in three blocks: (a) *Local computation steps* executed on the worker nodes and have access to the primary data. (b) The *algorithm flow* that orchestrates the algorithm execution, e.g., what computation step to run on the workers, how to aggregate and when to return a result. (c) The *algorithm specifications* involving implementation details. The algorithm developer dictates the workers to run a local computation from raw data or another computation result, and specifies how the data should be aggregated. The result of a local computation is kept as a pointer to the actual data. The algorithm may terminate or continue the computation. Currently, the MIP supports Python for algorithm development. Figure 2 shows an abridged example snippet for the linear regression fit. Note that federation, beside privacy (our primary goal), could also handle scalability issues frequently encountered in algorithm iterations and intermediate steps, which could be significant even with small data input.

Master Node. The Master node governs the communication with and among the workers and keeps track of the dataset availability on each worker for efficient algorithm shipping. It also orchestrates the algorithm flow and handles the aggregates returned from the local computations. Finally, it is also possible to perform computations locally as well.

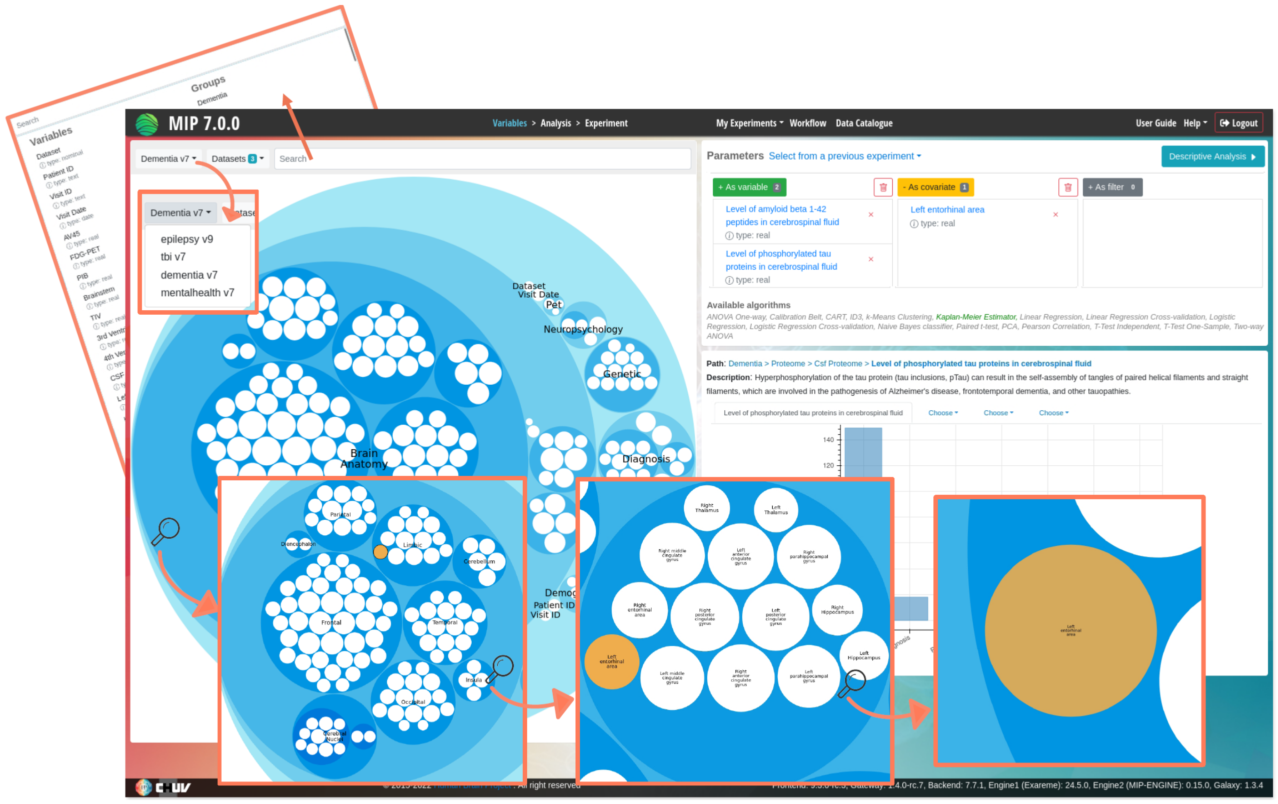
Worker Node. The Worker node hosts sensitive hospital data. It receives an execution request and performs local computations on the data. The request comes as a procedural code defined by the algorithm developer and MIP wraps it as a SQL UDF with the *UDFGenerator* (described next). Executing the algorithm inside a data engine is a strategic choice to leverage all the benefits of performant, in-database analytics, such as zero-cost copy, vectorization, and data serialization. Our implementation choice for the data engine is the open-source, analytics database MonetDB. Note that the source data in each hospital may be stored in a different form (e.g., csv files) or system and MIP provides the required ETL processes to upload it to MonetDB.

UDFGenerator. UDFGenerator follows a UDF-to-SQL approach (see [7, 8]) and JIT translates the procedural Python code to semantically equal declarative SQL code. To deal with the dynamic Python types, the Python functions are wrapped with a decorator that specifies their input/output types. SQL loopback queries, which enable executing SQL in a Python UDF, handle the multiple inputs and outputs of a Python function. These tasks are performed automatically by the UDFGenerator and no action is required by the algorithm developer. Our roadmap includes integrating this process with recent research advancements to in-engine, performant and stateful Python UDF execution using tracing JIT compilation and UDF fusion [1, 9].

Data Aggregation. MIP supports two methods to aggregate the results of the local computations and make them available on the Master node. A first, non-secure transfer, employs remote and merge tables (a MonetDB's feature) to ship local results back to the Master node and perform the aggregation there. (Note that the remote and merge tables are not materialized.) This technique is useful for applications handling non-sensitive data. However, the crown jewel of MIP's algorithm execution is the secure, privacy compliant computation that uses SMPC to collect the aggregated result.

Secure Multi Party Computation. SMPC is a cryptographic primitive where a set of computing nodes perform a computation of some function f over their private set of inputs revealing nothing but the output of the function f. SMPC facilitates precise calculations, guaranteeing that the outcome of any given function f remains consistent, irrespective of whether it is computed via SMPC or an alternative means. SMPC strengthens security and integrity in federated processing especially when combined with differential privacy, as it allows computations on encrypted data without the need for a trusted third-party to collect said data. However, SMPC may include potential computational and communication overheads, particularly for computation types involving extensive multiplications, branching, and comparisons. But it is well-suited and relatively efficient for federated learning, where it is used for the aggregation of gradients or validation metrics across multiple data nodes.

In MIP, the data is converted to secret shares in the Workers to ensure that there is no information leak to a malicious user or breached node. Then, the Master node signals the SMPC cluster, the SMPC nodes import the secret shares from the Workers and run the SMPC protocol. When the SMPC computation finishes, the result is sent to the Master node and the algorithm flow continues. In more detail, when a computation is triggered, it is assigned a global unique identifier, which is used to retrieve results asynchronously and to specify which data should be used for each computation. Worker nodes perform local computations and store these intermediate computation results locally indexed by the corresponding job identifier and are later securely imported through secret sharing to the SMPC nodes so as to be aggregated. The SMPC engine is designed to support aggregation



Path: Dementia > Proteome > Csf Proteome > Level of phosphorylated tau proteins in cerebrospinal fluid

Description: Hyperphosphorylation of the tau protein (tau inclusions, pTau) can result in the self-assembly of tangles of paired helical filaments and straight filaments, which are involved in the pathogenesis of Alzheimer's disease, frontotemporal dementia, and other tauopathies.

ANOVA One-way, Calibration Belt, CART, ID3, k-Means Clustering, Kaplan-Meier Estimator, Linear Regression, Linear Regression Cross-validation, Logistic Regression, Logistic Regression Cross-validation, Naive Bayes classifier, Paired t-test, PCA, Pearson Correlation, T-Test Independent, T-Test One-Sample, Two-way ANOVA

Frontend: 8.0.0-rc.5, Gateway: 1.4.0-rc.7, Backend: 7.7.1, Engine1 (Exareme): 24.5.0, Engine2 (MIP-ENGINE): 0.15.0, Galaxy: 1.3.4

Figure 3: MIP dashboard: domain, datasets, search, parameters, statistics, multi-facets exploration

of vectors which can be used for the implementation of various distributed algorithms (e.g., gradient descent) and statistical computations (e.g. mean).

Our platform supports two security modes: the full threshold (FT) and Shamir's secret sharing schemes. FT is very secure with abort against an active-malicious majority threat model meaning that even if a single SMPC node follows the protocol faithfully the data stays secure. But, computations are slow with FT. Shamir's secret sharing scheme (with $t<n/2$, $t>=n/3$) is much faster, but is secure only against honest-but-curious threat models. The data owners can choose either scheme based on a security-efficiency trade-off.

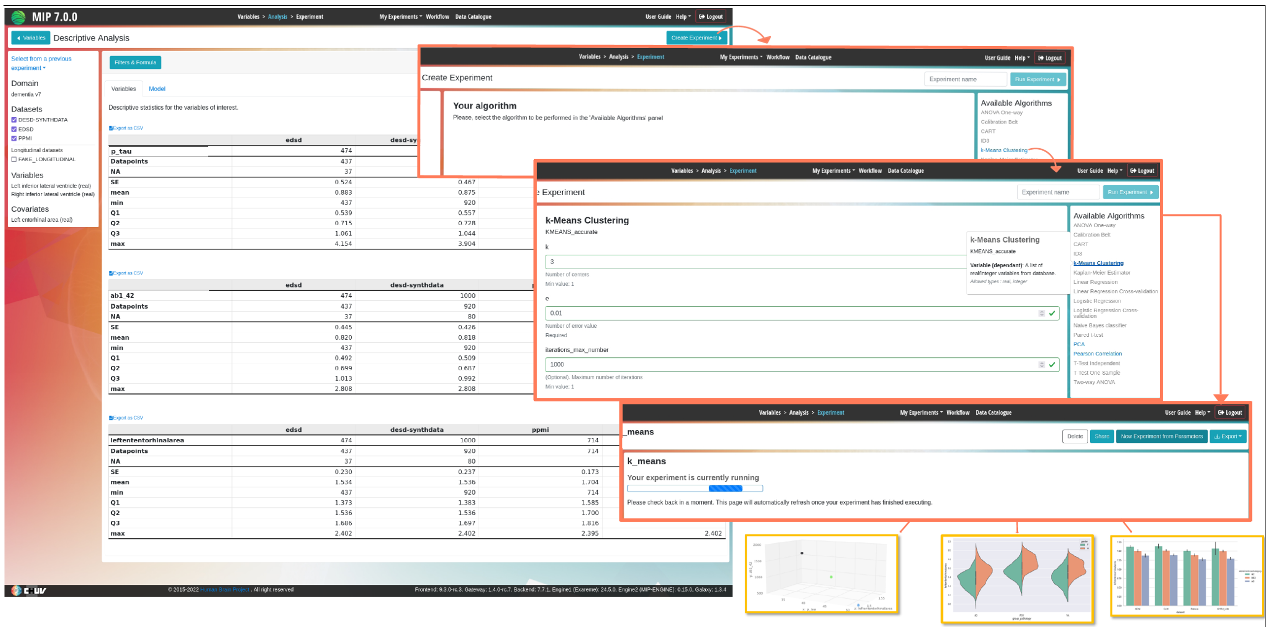
SMPC and Worker nodes are decoupled (different entities). Hence, data import from Worker to SMPC nodes needs to maintain security against the chosen threat model. As such, we are interested in a procedure that will secret share each entry in a dataset between the SMPC nodes. For Shamir secure importation, it is sufficient to secret-share the data on the Workers and send over secure channels the shares to each SMPC node. For FT secret sharing, we need to maintain security in the secure-against-active-malicious threat model and for that, we follow the mechanism in [2].

Training. The federated learning training phase works as follows. The Master sends to Workers (data holders) the current model parameters. Each Worker computes the parameter updates of the model on his local dataset. Next, we have two options: use differential privacy (DP) or secure aggregation (SA). For the local differential privacy (DP) guarantee, the Worker injects noise using Gaussian and sends the result to the Master node for aggregation with other Workers' updates. With SA, instead of injecting noise, the Worker securely imports (secret sharing) local updates to the SMPC cluster. Then, using an SMPC protocol we aggregate the data and inject noise. In both cases, the result is sent to the Master, which updates the model parameters and starts a new cycle. In practice, we have also seen excellent results for model training with other methods too (e.g., [10]).

Implementation. For the algorithm flow we employ NumPy (1.24), SciPy (1.10), Pandas (1.5), scikit-learn (1.2). The inter-node communication is governed by Celery (5.2) on top of RabbitMQ containers. We use a REST API served by Quart (0.18). The MIP engine is deployed with MicroK8s (1.25). The Workers run MonetDB (11.45.13) and RabbitMQ (3.9.14). The SMPC engine has been developed on top of SCALE-MAMBA [2]. Our software runs the SPDZ protocol [3], which speeds up computation by running a lot of the required SMPC computations in an offline phase. SMPC computations are implemented in MAMBA and support aggregation operations such as sum, multiplication, min/max operation and disjoint union. The engine also supports injecting Laplacian and Gaussian noise during the SMPC to the result of the computation.

Current status. The MIP[1] currently integrates 15+ algorithms for data analysis, applicable in subsets to various pathologies (dementia, epilepsy, mental health, traumatic brain injury), such as: k-Means Clustering, ANOVA one/two way, CART, Calibration Belt, ID3, Kaplan-Meier Estimator, Linear Regression, Logistic Regression, Naive Bayes Training, Naive Bayes with Cross Validation, Pearson Correlation, Principal Components Analysis, T-Test Independent, T-Test One-Sample, T-Test Paired. It supports several data types, such as clinical information, regional brain volumes, intracerebral EEG.

[1]More details about MIP can be found at: https://github.com/HBPMedical/mip-docs



Descriptive statistics for the variables of interest.

	edsd	desd-sy
p_tau	474	
Datapoints	437	
NA	37	
SE	0.524	0.467
mean	0.883	0.875
min	437	920
Q1	0.539	0.557
Q2	0.715	0.728
Q3	1.061	1.044
max	4.154	3.904

	edsd	desd-synthdata
ab1_42	474	1000
Datapoints	437	920
NA	37	80
SE	0.445	0.426
mean	0.820	0.818
min	437	920
Q1	0.492	0.509
Q2	0.699	0.687
Q3	1.013	0.992
max	2.808	2.808

	edsd	desd-synthdata	ppmi	
leftententorhinalarea	474	1000	714	
Datapoints	437	920	714	
NA	37	80		
SE	0.230	0.237	0.173	
mean	1.534	1.536	1.704	
min	437	920	714	
Q1	1.373	1.383	1.585	
Q2	1.536	1.536	1.700	
Q3	1.686	1.697	1.816	
max	2.402	2.402	2.395	2.402

Create Experiment

Your algorithm

Please, select the algorithm to be performed in the 'Available Algorithms' panel

k-Means Clustering

KMEANS_accurate

k: 3

Number of centers

Min value: 1

e: 0.01

Number of error value

Required

iterations_max_number: 1000

(Optional). Maximum number of iterations

Min value: 1

k-Means Clustering

KMEANS_accurate

Variable (dependant): A list of real/integer variables from database.

Allowed types : real, integer

k_means

Your experiment is currently running

Please check back in a moment. This page will automatically refresh once your experiment has finished executing.

© 2015-2022 Human Brain Project . All right reserved

Frontend: 9.3.0-rc.3, Gateway: 1.4.0-rc.7, Backend: 7.7.1, Engine1 (Exareme): 24.5.0, Engine2 (MIP-ENGINE): 0.15.0, Galaxy: 1.3.4

Figure 4: MIP analysis: descriptive analysis, experiment design and execution, algorithms, results

3 OUR PRESENTATION

Our presentation script starts with the scenario introduced in Section 1. We will first showcase the analysis from a scientist's perspective through our UI, and then we will demonstrate a step-by-step execution of the two MIP algorithms related to this scenario, k-means and linear regression, inside the MIP engine.

Interactive data analysis. We will present two popular functions: *data exploration* and *descriptive analysis*.

Data exploration. The MIP dashboard (Figure 3) displays the data of the chosen domain (e.g., dementia, epilepsy), the variable and covariate panels, along with descriptions and statistical information. The user sees an overview of the domain, with various data subsets (left side). The user may graphically drill down to various level of detail. In this example, a closer look of the Brain Anatomy section reveals that the data is categorized based on different parts of the brain, and after descending one more level within the limbic system, we see the available data on the different nuclei and subnuclei. There, the user may select a variable and generate views with descriptions and graphs (right side).

Descriptive analysis. The script continues with creating an experiment (Figure 4). The user is presented with a list of algorithms, whose availability depends on the selected variables. After the user configures the algorithm to their needs, they can run the experiment and study the results in tabular forms and appropriate graphs (see bottom-right). In this example, we use the variables amyloid-beta 1-42, the level of phosphorylated tau protein, and the volume of the left entorhinal area as input for a k-means algorithm, and configure it specifying the number of centroids to compute, the number of acceptable error value and the maximum number of iterations.

MIP engine process. Next, our script continues with demonstrating the system internals with canned queries and functions. This includes showing the various operations on Master and Worker nodes, the initialization, federation, and execution of the algorithms, the machinery of data and algorithm shipping, the secret sharing and SMPC-ing of the results, and the return of the aggregate results to the Master node and eventually to the UI. We will showcase specific internal MIP components including the UDF implementation and its seamless integration into MonetDB. Finally, the interested audience may create and execute their own Python UDFs on synthetic medical data.

User interaction. For off-script presentation, we will show three more use cases of applying MIP federation in brain injury [6], dementia [4], and mental health [5]. We will also have various scientific experiments prepared (e.g., data, algorithms, UDFs) and the audience will be able to run experiments on-demand. The interested participant may create a new experiment by exploring various scientific datasets and design ad hoc algorithm flows.

Acknowledgments. This research was supported by the EU Horizon 2020 EBRAINS (No. 945539, Human Brain Project SGA3) and EU Horizon EBRAINS 2.0 (No. 101147319).

REFERENCES

[1] Konstantinos Chasialis, Theoni Palaiologou, Yannis Foufoulas, Alkis Simitsis, and Yannis Ioannidis. 2024. QFusor: A UDF Optimizer Plugin for SQL Databases. In *ICDE*.

[2] Ivan Damgård et al. 2015. Confidential Benchmarking based on Multiparty Computation. *IACR Cryptol. ePrint Arch.* (2015), 1006.

[3] I. Damgard et al. 2011. Multiparty Computation from Somewhat Homomorphic Encryption. Cryptology ePrint Archive, Paper 2011/535. https://eprint.iacr.org/2011/535 https://eprint.iacr.org/2011/535.

[4] EBRAINS. 2023. MIP Federation in Dementia. Available at: https://ebrains.eu/service/medical-informatics-platform/mip-federation-in-dementia.

[5] EBRAINS. 2023. MIP Federation in Mental Health. Available at: https://ebrains.eu/service/medical-informatics-platform/mip-federation-in-mental-health.

[6] EBRAINS. 2023. MIP Federation in Traumatic Brain Injury. Available at: https://ebrains.eu/service/medical-informatics-platform/mip-federation-in-traumatic-brain-injury.

[7] Yannis Foufoulas and Alkis Simitsis. 2023. Efficient Execution of User-Defined Functions in SQL Queries. *PVLDB* 16, 12 (2023), 3874–3877.

[8] Yannis Foufoulas and Alkis Simitsis. 2023. User-Defined Functions in Modern Data Engines. In *ICDE*.

[9] Yannis E. Foufoulas, Alkis Simitsis, Eleftherios Stamatogiannakis, and Yannis E. Ioannidis. 2022. YeSQL: "You extend SQL" with Rich and Highly Performant User-Defined Functions in Relational Databases. *PVLDB* 15, 10 (2022).

[10] Jakub Konečný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. 2016. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *CoRR* abs/1610.02527 (2016).

[11] Arseny A. Sokolov et al. 2020. Greater than the sum: Federated analyses in Alzheimer's disease using the Human Brain Project Medical Informatics Platform (MIP). *Alzheimer's & Dementia* 16, S4 (2020), e045717.

[12] Wikipedia. 2023. Technology readiness level. Available at: https://en.wikipedia.org/wiki/Technology_readiness_level.