Deliverable 3.3

Final design and preliminary version of the data valuation tools and trading engine

H2020-EU-2.1.1: **PIMCity**

Project No. **871370**

Start date of project: 01/12/2019

Duration: 33 months

Deliverable delivery: 15/06/2021

Deliverable due date: 31/05/2021

Document Information

Document Name: Final design and preliminary version of the data valuation tools and trading engine

WP3 – Title: Understanding the new data economy

Tasks 3.2 and 3.3

Revision: 01

Revision Date: 26-05-2021

Authors: UC3M, WIBSON, IMDEA

Dissemination Level

Project co-funded by the EC within the H2020 Programme		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Approvals

	Name	**Entity**	**Date**
Author	Daniel Fernandez	WIBSON	13/05/2021
Author	Rodrigo Irarrazaval	WIBSON	13/05/2021
Author	Javier Calvo	WIBSON	13/05/2021
Author	Pedro Reveriego	UC3M	13/05/2021
Author	Angel Cuevas	UC3M	13/05/2021
Author	Nikolaos Laoutaris	IMDEA	13/05/2021
Author	Santiago Andrés	IMDEA	13/05/2021
WP Leader	Rubén Cuevas	UC3M	13/05/2021
Coordinator	Marco Mellia	POLITO	13/05/2021

Document history

Revision	Date	Modification
Version 1	31 March 2021	
Version 2	15 Apr 2021	
Version 3	12 May 2021	

List of abbreviations and acronyms

Abbreviation	Meaning
DVTMP	Data Valuation Tool from Market Perspective
DVTUP	Data Valuation Tool from end User Perspective
TE	Trading Engine
DVT	Data Valuation Tool
CPM	Cost Per Mille
CPC	Cost Per Click
WTP	Willingness-to-pay
WTA	Willingness-to-accept
WP	Work Package

Executive Summary

Deliverable 3.3 describes the final design of tools devoted to understanding the value of users' data (Data Valuation Tools) and the tool for trading users' data with data buying entities (Trading Engine). It also delivers the preliminary version of the software implementing the Data Valuation Tools and the Trading Engine.

This document is part of the PIMCity Project, funded from the Horizon 2020 Program (ICT-13-2018-2019) under Grant Agreement number 871370. In this document, we provide the final detailed design as well as the first version of the software implementing the tools included in the WP3 of the PIMCity Project.
In particular, the *Data Valuation Tools from the Market Perspective (DVTMP), the Data Valuation Tools from User Perspective (DVTUP) and the Trading Engine (TE).* The technical decisions and design choices have been carefully discussed in plenary meetings as well as specific meetings of WP3 members. Based on these decisions, the partner responsible for the development of each tool has taken care of complete the design and initial implementation description which has been subject to reviews by other members of the WP3.
The result of this process is presented in this document.

Index

1 Introduction

This deliverable aims at presenting the final version of the design of the tools developed as part of the Tasks 3.2 and 3.3 within the WP3. It also delivers the preliminary version of the software implementing such tools, namely:

- *Data Valuation Tools from a market perspective (DVTMP)*
- *Data Valuation Tools from an end-user perspective (DVTUP)*
- *An open Trading Engine (TE)*

Moreover, this deliverable also describes how these tools will be integrated together to provide a consistent service and provide a first implementation of the APIs enabling such integration.

The TE is considered a key component of a PIM, responsible for trading the data of users registered/handled by the PIM with interested buyers. Hence, the TE serves as a communication interface between the PIM backend and the data buyers. There is a myriad of data types that can be sold. For coherence and concreteness in PIMCity, and the design of the tools referred to in this deliverable, we will focus on the two most common types of data being sold: 1) Bulk data and 2) Real-time audience data.

Bulk data is typically bought in a non-real-time manner to receive information from a (typically large) group of users. Some examples include: a health insurance company may be interested in people's medical records; a car insurance company may be interested in the people's mobility data to understand who drives through tough roads or at higher speeds; a mortgage-issuing company may be interested in people's financial records, etc. None of these data need to be traded in real-time, and typically data buyers try to buy it in bulk. Due to the relevance of the selling of bulk data for a large number of businesses, the TE will be designed and developed to enable the trading of bulk data to PIMs implementing it.

An audience is a term used in marketing to refer to a specific group of the population defined by three parameters: location (where the user is located); demographic information (age, gender, etc.); interests (a list of interests, e.g., outdoor activities, sports, science, automobile, etc.). Most of these parameters are typically extracted from well-known taxonomies such as the one offered by IAB (which is a de-facto standard in digital marketing)[1] Online advertising (a.k.a. digital marketing) is arguably one of the most important businesses exploiting data utilization, specifically audience data, to deliver targeted ads to users in real-time. Due to the importance of this business, the TE will be designed and developed to allow PIMs to implement it to trade audiences' data.

One of the main problems that PIMs face, and the fundamental reason for the failure of some of them so far, is their inability to assess the value of data. To address this issue in Task 3.2, we will work on developing two types of Data Valuation Tools (DVTs). In the

[1] https://iabtechlab.com/standards/audience-taxonomy/

context of the PIMCity architecture, the DVTs are meant to assist the TE in estimating the value of the data the TE is trading. As indicated above, WP3 will develop two types of DVTs:

On the one hand, the DVTMP will obtain audiences' value from some of the most important online advertising platforms such as Facebook, Instagram, or LinkedIn. The information collected by the DVTMP will assist the TE to estimate the value of audiences based on the market price of these on the aforementioned advertising platforms so that the trading price is well aligned with the market price of the audience. This enables the PIM to implement a combination of the DVTMP and TE to compete in the market of audiences purchasing/selling (e.g., the online advertising market). On the other hand, the DVTUP will analyze the value of data of individual users within a large pool of users. It will apply a value-based fairness principle, which is a novel principle to be used in the context of the data economy. Each user data contributes different data with a relative value compared to the data provided by other users. Let us consider the next example: A company willing to predict the traffic in a city. Professional drivers from the city can share their GPS data so that the aggregate data from a large pool of drivers will help modelling the city's traffic. However, the sample of data provided by each professional driver has not equal value. The driver providing data from the most crowded area where many other drivers are also providing data is not of much value since he provides redundant information. However, a driver providing data from less crowded areas where few other drivers provide data is much more worthy. Armed with this novel fairness-based data valuation, the DVTUP will enable the TE to estimate the relative value of different data sources in the context of bulk data trading for specific purposes.

Hence, following the modular design concept of PIMCity that will allow a PIM to use the whole PIMCity architecture or to implement just a few of its modules, WP3 implements two independent DVT modules which address the two considered data trading scenarios: <DVTUP – bulk data purchase scenario> and <DVTMP – audience-based purchase scenario>. Based on this design concept, integrating the two DVTs will be done independently with the TE.

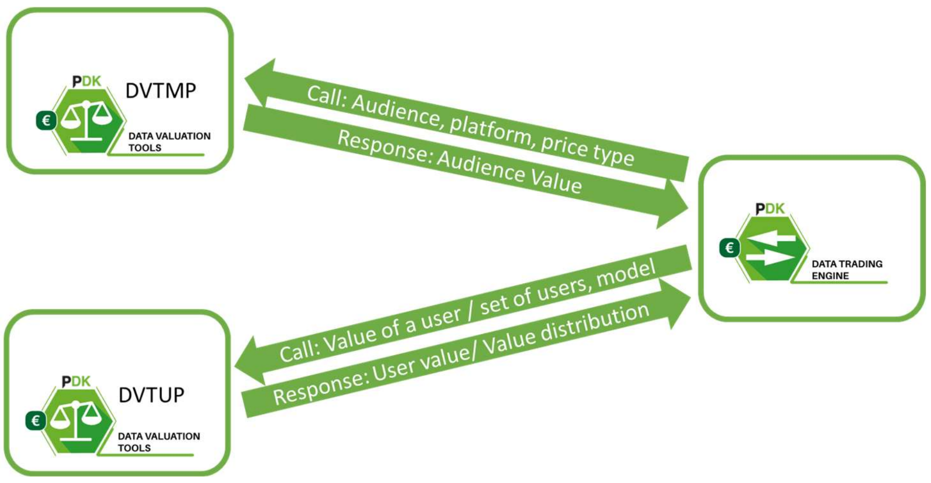

PDK
DVTMP
DATA VALUATION TOOLS
Call: Audience, platform, price type
Response: Audience Value
PDK
DATA TRADING ENGINE
Call: Value of a user / set of users, model
Response: User value/ Value distribution
PDK
DVTUP
DATA VALUATION TOOLS

Figure 1 Summary of modules and their interactions

Figure 1 shows a scheme for the high-level integrated design of the tools covered by this deliverable. The final design and the first version of the software implementing each of the three mentioned modules (DVTMP, DVTUP, and TE) will be described in the rest of the document. In addition, the details of the integration of the DVTMP-TE and DVTUP-TE and the first version of the APIs implementing it will be also presented.

Note that these modules' design and implementation have been done taking into account the requirements detailed in D1.1 from PIMCity. We suggest the reader interested in knowing the details to refer to such deliverable. This document presents the final design of the technical modules from WP3. However, the delivered software along with this deliverable should be considered a preliminary version and will be further improved in the next months of the project. The final version of the software implementing the DVTMP, DVTUP and TE will be delivered in D3.4

2 Deliverable Objective

The first goal of this deliverable is to provide the final description of the design of the technical modules to be developed as part of Task 3.2 and Task 3.3; these are the Data Valuation Tools from Market Perspective (T3.2), Data Valuation Tools from End-User Perspective (T3.2) and the Trading Engine (T3.3). Moreover, this deliverable provides a detailed description of the integration of these modules. In particular, as discussed in the Introduction, the integration will be done independently between the DVTMP and the TE and between the DVTUP and the TE.

The second goal of this deliverable is to deliver a preliminary version of the software implementing the DVTMP, DVTUP and TE as well as their associated APIs enabling the communication between these modules as well as the communication of the TE with modules developed in other WPs

Next, we provide a list of the specific objectives of this deliverable:

- Provide the final comprehensive description of the design of the DVTMP module (Section 3)
- Provide the final comprehensive description of the communication interface and integration of the DVTMP and the TE (Section 3)
- Deliver a preliminary version of the DVTMP software available in the PIMCity Gitlab repository (Section 3)
- Deliver a preliminary version of the DVTMP’s API that allow communications between the TE and the DVTMP modules (Section 3)
- Provide the final comprehensive description of the design of the DVTUP module (Section 4)
- Provide the final comprehensive description of the communication interface and integration of the DVTUP and the TE (Section 4)

- Deliver a preliminary version of the DVTUP software available in the PIMCity Gitlab repository (Section 3)
- Deliver a preliminary version of the DVTUP's API that allow communications between the TE and the DVTMP modules (Section 3)
- Provide the final comprehensive description of the design of the TE module (Section 5)
- Provide the final comprehensive description of the communication interface offered by the TE to other modules within the PIMCity infrastructure (Section 5)
- Provide the final comprehensive description of the communication interface offered by the TE to third parties for data purchase (Section 5)
- Deliver a preliminary version of the TE software available in the PIMCity Gitlab repository (Section 3)
- Deliver a preliminary version of the TE's API that allow communications between the TE and other modules within the PIMCity infrastructure and third parties (Section 3)

This deliverable partially addresses the objectives of WP3 described in the Grant Agreement, which we remind in the following:

- Implement tools to provide estimations (from the market and the user perspective) of the value associated with individual users' data.
- Design and implement a system, referred to as the trading engine, that integrates the data broker into the data market ecosystem allowing it to sell end-users' data to interested third parties.

Finally, it is worth mentioning that as a secondary but still very relevant objective, we have designed and implemented the different technical modules using as reference the requirements described for each of them in Deliverable D1.1.

3 Data Valuation Tools from Market Perspective

In the last decade, complex tracking infrastructure has been developed on top of the internet and global telecom networks. This tracking infrastructure leverages both the web and mobile ecosystems, so it can track not only our online activity (what websites we visit or which products we buy online) but also our offline activity (what places a user visits can be extracted from her mobility data obtained, for instance, through mobile apps). This tracking infrastructure collects data from users and profiles. Arguably, the most relevant business driving this tracking ecosystem is online advertising. Companies can know individual users' habits and preferences, thus delivering personalized advertisements or offers to each user.

Many of the most popular online services follow this business model, i.e., they define different techniques to profile users and provide them with personalized ads, products, offers, or services. Among them, we list Facebook, Instagram, Google, and LinkedIn. Some of these companies provide web platforms or APIs to allow advertisers to define their advertising campaigns. In many cases, part of the information they offer to advertisers through those interfaces includes an approximation of the end-user value. They also offer advertisers estimations of the Cost Per Mille (CPM), Cost Per Click (CPC), etc. Therefore, an advertiser may configure its target end-user profile (a.k.a. audience) in the service platform (e.g., Facebook). Advertisers will get in return the cost of showing one thousand ads to users meeting such profile (CPM) and the cost of getting a click in the ad to bring the user to the advertiser's landing page (CPC), etc. Data Valuation Tools from the market perspective (DVTMP) developed in PIMCity will leverage the existing online advertising platforms to estimate the value of hundreds to thousands of audiences (i.e., end-user's profiles) representing an important portion of the population. This will meet the specific objective defined in this WP in this regard.

3.1 Objective

The Data Valuation Tools from the market perspective (DVTMP) module developed in PIMCity will leverage some of the most popular existing online advertising platforms to estimate the value of hundreds to thousands of audiences. The DVTMP module aims to provide the monetary value of audiences traded on the main online advertising platforms. This will serve any PIM deciding to implement the DVTMP module to have a realistic estimation of audiences' value to be traded. PIMs which implement a more accurate value estimation module will have a competitive advantage in the market. Since the information about values collected from these advertising platforms is based on aggregated historical pricing data, we can assert that DVTMP provides full-privacy guarantees. Moreover, a given audience's value can be obtained in real-time from the referred online advertising platform. In particular, the design of the DVTMP pursues the following objectives:

1. Crawling data value of audiences from Facebook, Instagram, and LinkedIn
2. Process, clean, and curate the collected data
3. Store processed data
4. Provide access to the data through an API

3.2 Background and state of the art

We can find a large body of literature studying the value of information assets and privacy from a macroscopic economic perspective [1]. However, only recently, researchers have addressed the monetary value of personal information from a microscopic point of view in the context of online services. The most adopted methodology to capture the value of personal information has been retrieving directly from users the value they assign to personal data through surveys, interviews, etc. This methodology measures users' willingness-to-pay (WTP) to protect/recover their personal information and their willingness-to-accept (WTA) to sell their personal data. Then, it applies different mechanisms (e.g., conjoint analysis) to obtain the monetary value of personal information [2][3][4]. There are two main reasons why this approach is not valid: first, there is an existing prominent market which is already trading personal information and establishing market prices for personal data; second, recent interviews with advanced Internet users in the context of Facebook show a clear lack of societal awareness regarding the value of personal information, and thus users answers are too random to derive solid conclusions. Therefore, asking users is useful to understand their perception about the monetary value of their personal information, but not to understand the actual value of online personal data, which is established by the market irrespectively of Internet users' perception.

Another approach is the manipulation of aggregated market cap (revenue, net income, etc.) of companies exploiting personal information to quantify the monetary value of personal data records (e.g., dividing the revenue of a company by the number of active users to get the average monetary value per user) [5]. This methodology tries to find a unique answer valid for all users ignoring the fact that the value of personal information differs from user to user. Each user generates different revenue for an online service depending on the market price and his data and service activity. A comprehensive personal data valuation methodology should be able to provide personalized feedback per user and per online service.

A third methodology uses the economic cost of data breaches per data entry to estimate personal information's average monetary value. As happened in the first methodology described above, this methodology ignores that there already exists a market that trades personal information and defines prices associated with personal information transactions. Furthermore, a data breach has economic implications that go far beyond standard personal data trading. Deriving personal data value in this way will very likely overestimate actual market prices and will lead to wrong conclusions. [21]

The last approach tries to use actual market prices to estimate personal information's monetary value. The most recent works in the literature use this technique. In particular, DVTMP follows this approach to provide a reasonable estimation of PIMCitizens' data value. DVTMP proposes a link between the value of personal information to the revenue that internet users generate in real-time for online services based on the commercial exploitation of their personal information and the users' activity within those online services. Two specific implementations of this approach are the FDVT [1] and Pricom (integrated into the portal testyourprivacy.org) platforms implemented by the same UC3M team participating in PIMCity in a previous EU H2020 Project (TYPES). This team will use the knowledge acquired to implement these tools to design and develop the DVTMP. Note that there are some fundamental differences between FDVT/Pricom with DVTMP since the former was meant as informative standalone tools for citizens, whereas DVTMP is designed as an integrated module within the PIMCity architecture meant to interact with businesses.

3.3 Technical Design

The DVTMP module's goal consists of developing a software module that extracts audiences' prices from different popular advertising platforms, stores them, and delivers them to the TE under request. To obtain a realistic market price of audiences, the DVTMP queries the Marketing API of popular advertising platforms (Facebook, Instagram, and LinkedIn) to obtain the reported price advertisers pay to deliver ads to specific audiences. This module queries the marketing APIs specifying the parameters of the audience being targeted and obtains in return an estimated price of the audience in the form of standard price metrics such as the price for showing 1000 ads to users from that audience (CPM) or the price of a click (CPC). The obtained market value of each queried audience is locally stored in a database (Cache). The DTVMP is designed to obtain, store, and update the information from hundreds to thousands of audiences on each of the considered platforms. Third parties can access the information stored by the DTVMP module through its API interface. In PIMCity, the only module communicating with the DTVMP using this API is the TE. Note that if the DTVMP module does not have updated information on the requested audience's price, it can obtain in real-time and answer the third party requesting the information. Finally, it is worth noting that the DVTMP has been designed with a modular architecture to give the technology versatility to add new data sources of audiences' prices simply.

3.4 Internal Operation

Figure 2 shows the technical design of the DVTMP detailing its building blocks and processing structure:

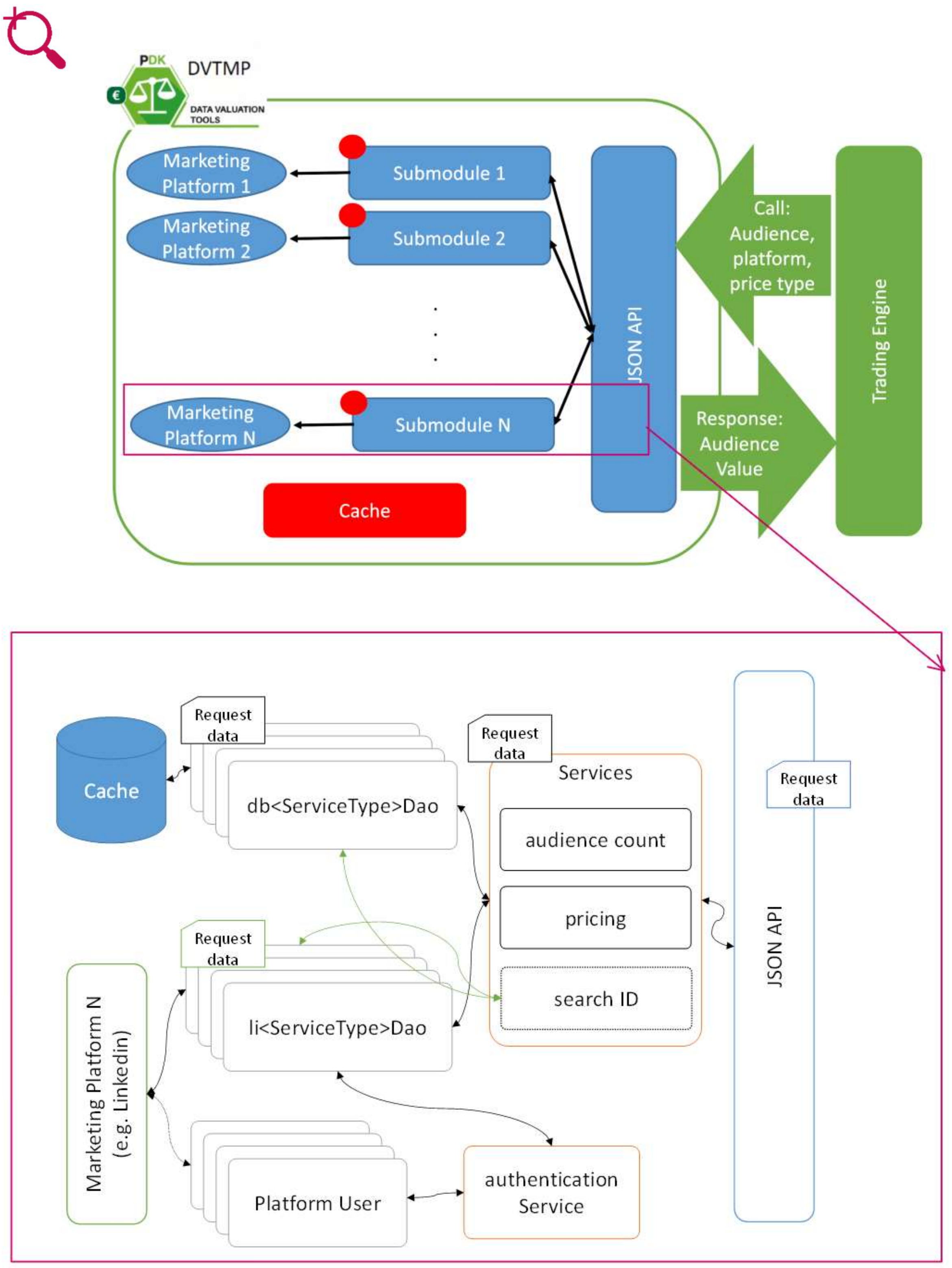


Figure 2 DVTMP architecture[2]

[2] Dao refers to Data Access Object

The building blocks are described below:

- **JSON API block**: this block implements the communication interface between the DVTMP module and the TE. In particular, it implements an API that receives queries with a predefined format. Those queries are processed and sent to the Services block for its subsequent processing. Details of the JSON API block's specific operation are provided in the next subsection (3.3.2).
- **Services block**: there are three different sub-blocks in this building block, with each one performing a different task. The functionality of each block is provided below:
  - **Search ID sub-block**: when the API receives a call, it includes a set of interests, locations, and demographic information (keywords). This call is string-based. However, each advertising platform associates each of these keywords to an internal ID for its marketing platform. This sub-block aims to retrieve the specific ID associated with the different keywords in the targeted platform. To this end, the Search ID sub-block first searches for the keyword's ID in the cache. In case the information is not available locally (note that the ID usually does not change over time), it launches a query to the targeted online platform to gather the keyword's ID. For example, "Football" as a keyword is associated with an ID on Facebook (6003107902433). In that case, the Search ID sub-block sends a request to Facebook and returns "6003107902433". If "Football" is available in the cache, this sub-block returns this ID without sending a Facebook request.
  - **Pricing sub-block**: This block is in charge of obtaining pricing information. Once the module receives a call asking for an audience price, it specifies the platform and the set of interests, location, demographics, skills, etc., which define the targeted audience. Each parameter is associated with an internal ID. For example, in the audience "*Women in France interested in Science or Technology and skilled in Blockchain*"; the marketing platform links each parameter (Women, France, Science, Technology, Blockchain) to a different ID. Following the previous example, the *Search ID* sub-block would have previously searched each parameter ID. The *Pricing* sub-block first checks whether the pricing information for the considered combination of IDs (Demographics: Women, Location: France, Interests: Science, Technology, Skill: Blockchain) is stored in the cache. In this case, the pricing information of the audience is retrieved from the cache. If the pricing information is not available locally, the Pricing sub-block launches a request to the considered marketing platform to retrieve the pricing information.
    In particular, the Pricing sub-block retrieves pricing information from two well-known and extensively used pricing metrics in the online advertising ecosystem:
    - CPM (Cost Per Mille impressions). The value of one thousand (1000) impressions.
    - CPC (Cost Per Click). The amount charged to the advertiser each time a user clicks on the ad.

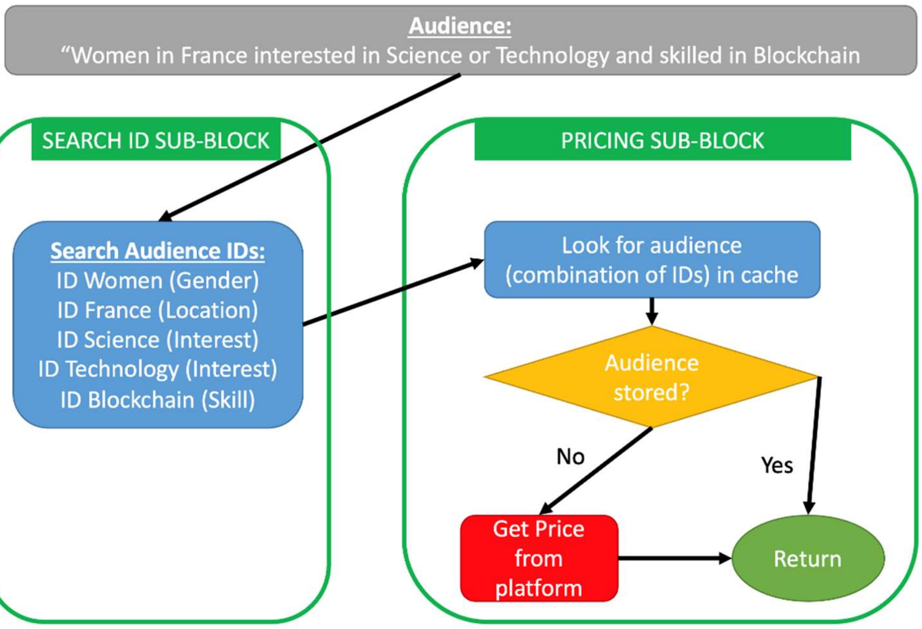


Figure 3 Services Block

- **Audience count sub-block**: Some marketing platforms also estimate the targeted audience's size in addition to the pricing information. While this information is not needed for the price estimation, it might be relevant for future use since it allows us to understand the targeted audience's popularity (i.e., how many users fit the targeted audience). This sub-block operates similarly to the Pricing sub-block, using the cache: it first looks for the cached information, and if the information is not available locally, it sends a request to the considered marketing platform. Note that the information about the audience's price and size can be retrieved together from a single query for some marketing platforms.

- **Authentication service block**: Most of the considered marketing platforms require to have an account logged in the associated service to access the marketing API (e.g., an account logged in Facebook to launch queries to its marketing API). The Authentication service block is designed with a group of preconfigured accounts for each considered marketing platform ("Platform User" in Figure 2). These accounts are provided by the PIM provider and do not involve PIM users. The credentials will be used to automatically log into the different systems and query their associated marketing APIs. Note that this block is initialized by the Services blocks so that when the latter ones are executed, they can launch queries to the marketing APIs.

- **Marketing platform block**: This block reads the parameter referring to the targeted platform received through the JSON API block and will set up this as the platform to be used by the Authentication and Services blocks.

- **Cache block**: This block implements a database where the collected information from the different marketing services is stored. This database serves as a local cache that speeds up the response to queries from third parties whenever the requested information is available locally. The stored information will include a timestamp, and it will only be returned if it is not outdated (too old). In that case, the marketing API is queried, and the local information is updated.

The flow chart shown in **Figure 4** summarizes the conceptual execution process of the DVTMP module for a specific advertising platform. Note that there is an independent submodule taking care of each advertising platform. This design provides flexibility to easily extend the DVTMP to include new data sources from other advertising platforms.

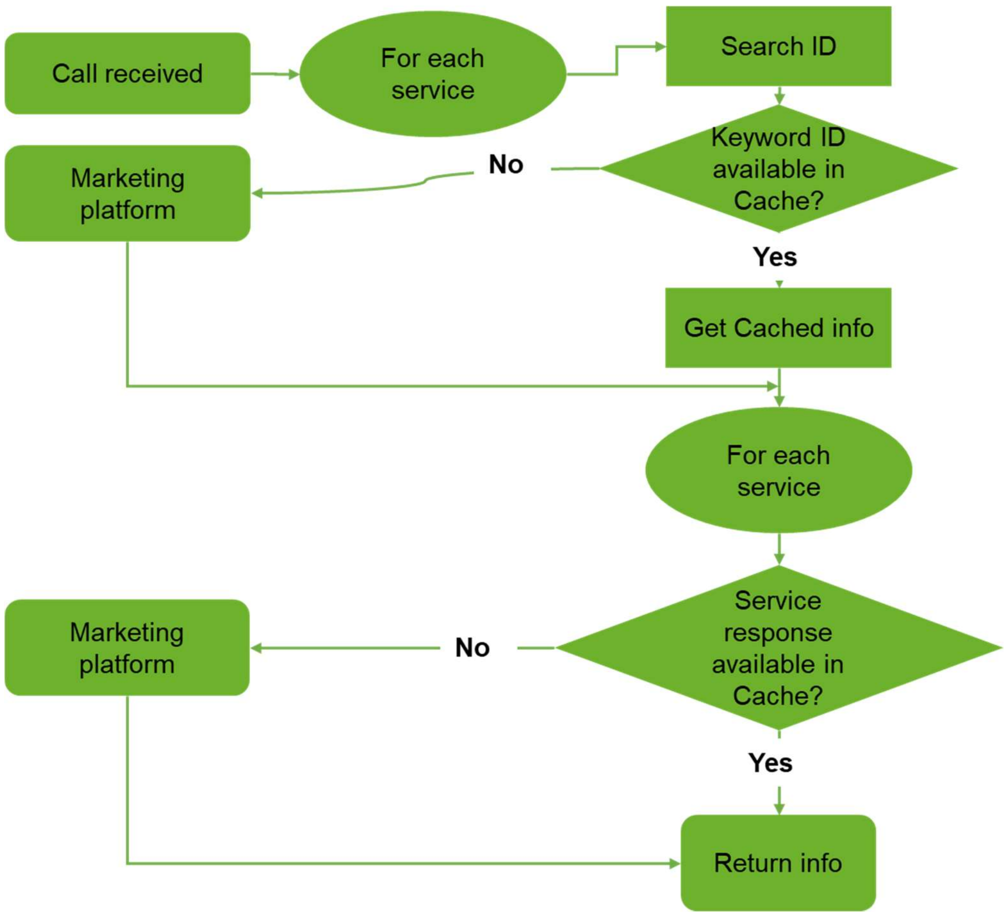


Figure 4 Workflow of each platform submodule

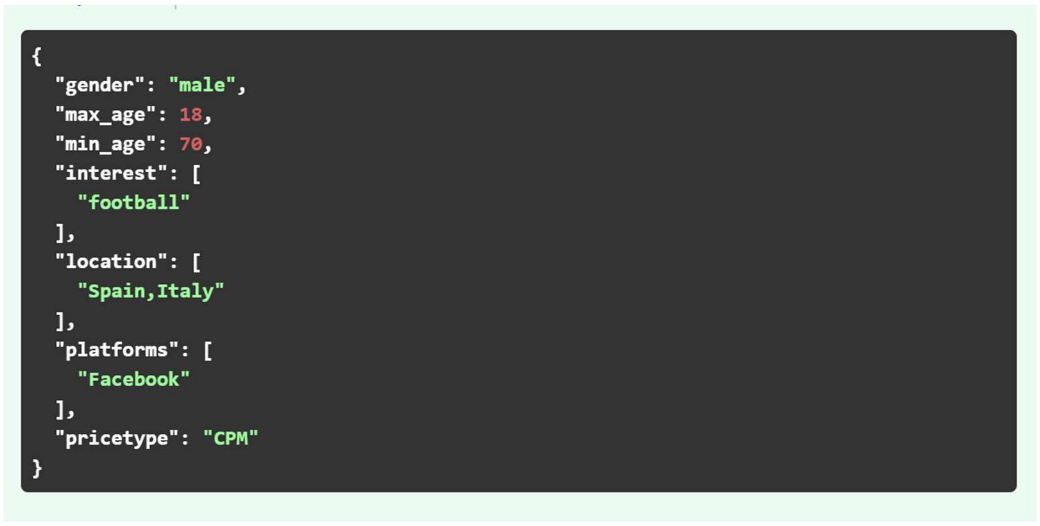
3.5 Interfaces

This section describes the JSON API block, which will serve as a communication interface between the DVTMP module and the Trading Engine. We define a basic request/response procedure, which on the one hand, simplifies the integration process of DVTMP with third parties and, on the other hand, makes it easy to extend the type of calls to be used. The JSON API block receives a query that specifies the required parameters for the DVTMP module's operation: the keywords (location, demographic information, interests, etc.) and the targeted platform. Upon the query's reception, the REST API triggers the sequence of procedures described in **Figure 3**. Once the DVTMP module's operation is completed and the information is ready, it generates a JSON file as the response for delivering to the third party (the TE in the case of PIMCity architecture).

Once our module receives a request, the JSON API analyses it. The JSON file will include the platforms for which the module will extract the information (Facebook, LinkedIn, Instagram). The module will first identify the marketing platform targeted in the request. Then, the corresponding platform submodule analyses the request's parameters (**Figure 4**). Since each marketing platform has its peculiarities regarding the offered information and how to access it (e.g., search for ids, audiences, or prices), their submodules are also independent. Finally, once the submodule(s) retrieves the requested information, this is sent to the JSON API block, creating a JSON object that will respond to the received request.

Figure 5 presents the OpenAPI documentation for a POST request in Swagger[3] format. This documentation is uploaded in the PIMCity online repository:

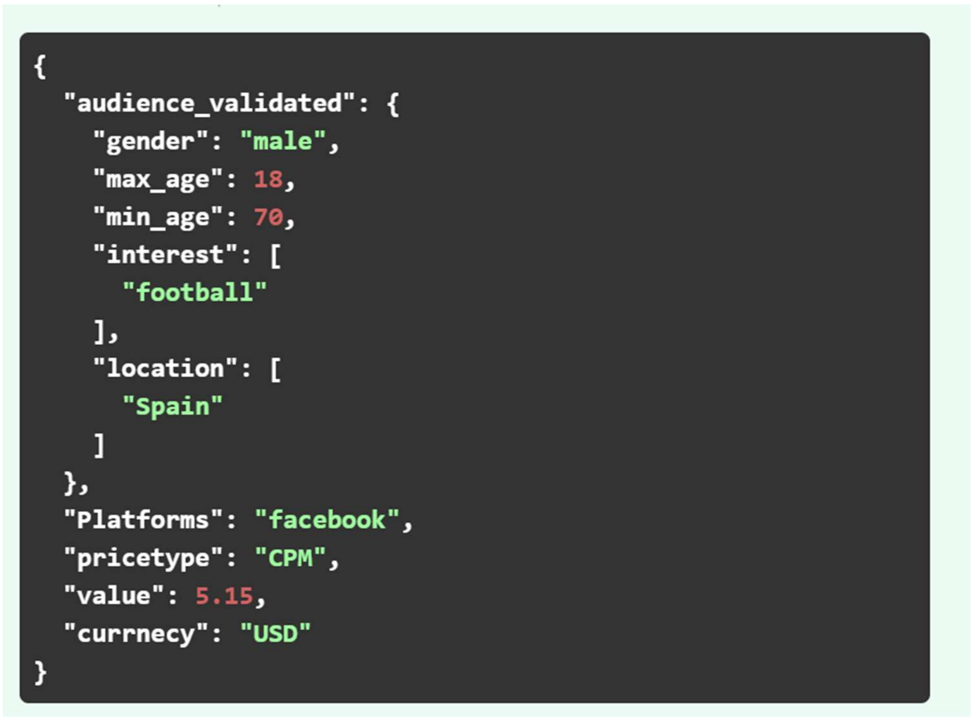
https://gitlab.com/PIMCity/wp5/open-api/-/blob/master/WP3/DVTMP.yaml



```
{
  "gender": "male",
  "max_age": 18,
  "min_age": 70,
  "interest": [
    "football"
  ],
  "location": [
    "Spain,Italy"
  ],
  "platforms": [
    "Facebook"
  ],
  "pricetype": "CPM"
}
```

Figure 5-a. Sample POST request body to the module

[3] https://swagger.io/



```
{
  "audience_validated": {
    "gender": "male",
    "max_age": 18,
    "min_age": 70,
    "interest": [
      "football"
    ],
    "location": [
      "Spain"
    ]
  },
  "Platforms": "facebook",
  "pricetype": "CPM",
  "value": 5.15,
  "currnecy": "USD"
}
```

Figure 5-b. Sample response to the previous POST request

3.6 Configuration and Use

The preliminary software implementation of the Data Valuations Tool from Market Perspective can be found in the following directory of the PIMCity's Gitlab in the following address:

https://gitlab.com/pimcity/wp3/dvtmp/-/tree/master/DVTMP/Build

3.7 Dependencies

- Docker Engine: v20.10.x
- Docker Compose: v1.27.x

3.8 Configuration

The project configurations can be found in this address:

https://gitlab.com/pimcity/wp3/dvtmp/-/blob/master/Build/request_handler/config.env

The required configuration parameters are as following:

NODE_ENV= Node js the environment (e.g., "development")

PORT= server port

DATABASE_LOCAL= Mongo database address (e.g. "mongodb://<DB_address>:27017")

FB_TOKEN = Facebook test account access token

FB_AC_ID = Facebook test account ID

IG_TOKEN= Instagram test account access token

IG_AC_ID= Instagram test account access token

LINKEDIN_USER = LinkedIn test account information

LINKEDIN_PASS = LinkedIn test account password

Two images will be created and linked through a network by setting the configurations and composing the dockers.

- DVTMP image
- MongoDB image

3.9 Execution

For starting the data valuation service, the following script should be executed in the repository root:

```
$docker-compose up
```

4 Data Valuation Tools from an End-User Perspective

4.1 Objective

The objective of the Data Valuation Tools from an End-User Perspective (DVTUP) module is to provide estimated valuations of end-users' data for the dataset they are selling through the marketplace as bulk data. In particular, DVTUP will provide tools for the TE to:

i. Provide buyers with a hint of how valuable a piece of data is for a certain type of model or even for a specific task.

ii. Calculate a fair breakdown of data transaction charges by seller, looking forward to rewarding each user proportionally to the value that each piece of data from different sellers brings to the buyer for a specific task.

In the first case, the output will be the expected accuracy the buyer will get from a dataset if purchased from the marketplace. In the second case, the output will estimate the percentage of a transaction value that corresponds to each seller, and a log of data and results obtained to justify rewards paid to different sellers. For that purpose, different methodologies and algorithms will be designed and implemented to allow data marketplaces to calculate such breakdown in different ways, namely: i) using traditional heuristics such as data volume or the number of sources, ii) using a value-based data evaluation framework to test data on the specific task the buyer is intended to use it in, and breaking the payment according to a value function (e.g., accuracy yielded by a given model or algorithm) provided by the buyer.

4.2 Background and state of the art

According to our survey (see Deliverable D3.1), existing data marketplaces just put in touch data sellers and buyers and leave data pricing to sellers or be negotiated between both parties. Some marketplaces provide data valuation (Dawex[4]) services or data monetization and marketing (Battlefin[5]) services to data sellers. In practice, current data transactions involve one seller and one buyer, and existing data marketplaces usually leave enriching and combining data to Internet service providers. In fact, traditional data providers have made good business out of enriching and combining data, although they cannot be considered data marketplace platforms. Instead, they decouple data sourcing, which they handle directly when needed with their data providers, from the data-driven services they sell to the end-users. Sourcing is usually done through an opaque partnership or specific data acquisition agreements. On the contrary, a data marketplace is a two-sided platform that transparently puts in touch sellers with buyers by matching data owned by the former to solve the latter's data sourcing needs. The research community has made different efforts to define a value-based data valuation framework that allows buyers and data marketplaces to discriminate between data pieces. Some vision papers provide an overview of data marketplace challenges and propose a high-level architecture to solve most of them [14]. Actually, some authors are proposing theoretical platforms that leverage a data valuation framework [7] [12] to trade with "accuracy" rather than with data.

From the users' perspective, the main concerns (and challenges faced by data marketplaces according to D3.1) are whether their data is under control and getting a fair reward for their data. Regarding the first problem, data marketplaces deal with consent management, anonymization, and data replication issues to gain data sellers' trust, and these topics are out of the scope of this deliverable. So, we concentrate on the second problem, particularly how users perceive their data's value. Different methodologies have been used to measure the value of personal data [13]. Traditionally, such a problem has been treated in the context of preserving privacy from the online advertising industry. Different studies have tried to determine the value of data based on surveys conducted among people to determine how much they value their privacy [6]. Other studies refer to economic experiments about how users evaluate different types of their Personal Information (PI) being shared while being online [11]. However, our goal is to set up a framework that is useful to work in other domains and applications. For instance, how much is a user's location information for a Business Intelligence (BI) demand prediction algorithm? In that general setting, it is generally accepted that users have no idea of their data's value.

As a result, their main concern in terms of value is whether it is valued accordingly to similar data from other users when shared as part of a transaction. In other words, in terms of value, and assuming that the market is setting prices to close data transactions, users are

[4] https://www.dawex.com/
[5] https://www.battlefin.com/

more concerned by the "relative value" of their data in comparison to other users that fulfil the same requirements, and the transparency of the market in reporting these rewards. A value-based data valuation framework is being proposed for revenue allocation in such theoretical marketplaces. Researchers resort to well-known concepts of game theory to split the revenue among. Some papers propose using the Shapley value [16] for such a task [7], whereas others propose to use *the core* [17]. As its exact calculation requires an exponential processing time as the number of sources increases, different papers devoted to building approximations to Shapley value in different settings [9] [10] allow users to trade computing time and accuracy.

Regarding revenue allocation in practice, existing data marketplaces tend to use heuristics to estimate data's value. It is common to price datasets or data services depending on the volume of data downloaded, and PIMS usually charge information based on the number of shared users, regardless of the value of each user. Leave-one-out (LOO) has been used for "denoising" datasets by omitting pieces of data that reduce the accuracy of machine learning [7]. On the one hand, these methodologies are simple to calculate and implement. On the other hand, simple heuristics are not necessarily tied to the utility of data [15], and consequently could be considered unfair by sellers and provide the market with the wrong incentives. Not only will a value-based data valuation make sellers happier, but incentivize the provision of better-quality data to the market, as well.

4.3 Technical Design

DVTUP has been developed under the following assumptions:

1. The value of a piece of data is closely tied to the specific buyer, and the purpose of traded data is intended to be used. Consequently, neither buyers nor the marketplace can estimate the value of data on their own if such a context is not considered.
2. The marketplace will play a mediation role between buyers and sellers, providing buyers with suitable information that fits their needs by combining, processing, or just reselling data that users share with the marketplace platform.
3. Once a transaction is closed, and data is delivered, the buyer will make a payment to the marketplace, and the marketplace will share this income (partially or wholly) with users providing data for such bulk transactions.
4. Sellers will be concerned about getting a fair reward for their data, which is justified and according to the rewards of other users whose data has also been traded.

This module provides the TE with a series of methods to understand the value of data from different contributors for a specific task or model, either selected from a catalog or implemented by the buyer to work with data from the marketplace. In the second case, DVTUP can tailor the results to the task the buyer intends to use data for. At a high level, the following **figure** describes how DVTUP works.

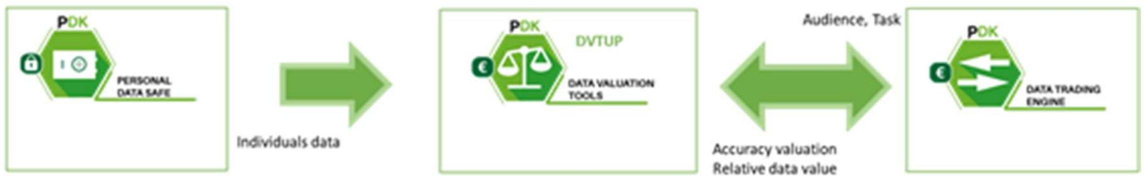


Figure 5 High-level design of DVTUP

DVTUP provides the data trading engine with valuations of the marginal accuracy that data from individuals or users bring to a specific model. This allows the TE:

1. To report such accuracy valuation to buyers **before a data purchase decision is made** by the buyer, and effectively implement a Try-Before-You-Buy scheme as described in Deliverable D3.1
2. To calculate the **value-based distribution of rewards** among individuals that contributed to a certain traded dataset **after a transaction is closed**

For that purpose, the DVTUP will need to have access to the P-DS to test the data in the PIMS with the specific algorithm in the catalog or either provided by the buyer. As TE filters individuals who consented to provide their data for such a task when building the audience, the DVTUP will not have access to data from users who are not willing to exchange their data. The following sections describe the internal operation of the DVTUP, the interface, and the functions it provides for the TE.

4.4 Internal Operation

Figure 6 summarizes the internal technical design of the DVTUP tool. It is founded on the value-based data valuation framework (hereinafter, vbdeFramework), which defines a series of abstractions needed to calculate data's relative value in a general setting. Specific implementations of each application's framework interfaces are needed to tailor to buyers' specific use cases. The framework will also be providing some classic AI/ML models, which can also be selected by buyers to be trained with data from the marketplace to evaluate the different outcomes. In addition to this central module, it will be required:

- An intermediate layer exposing the API to convert API requests from the TE to vbdeFramework orders
- A module to get data from the marketplace users, or information sellers
- An internal database to store logs and information generated during the operation of DVTUP

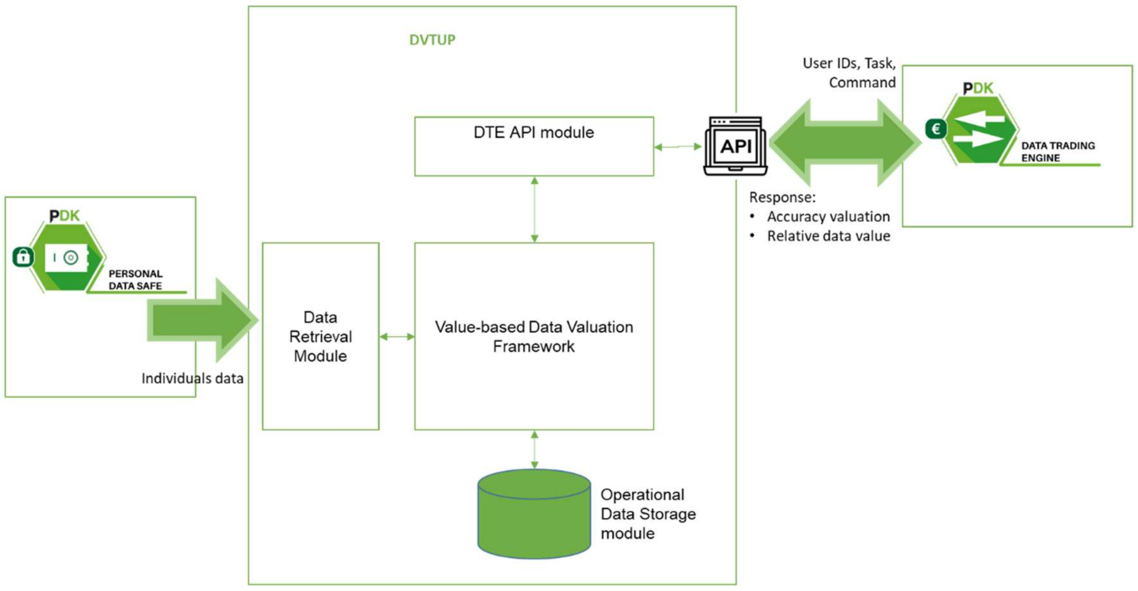


Figure 6 General design of DVTUP

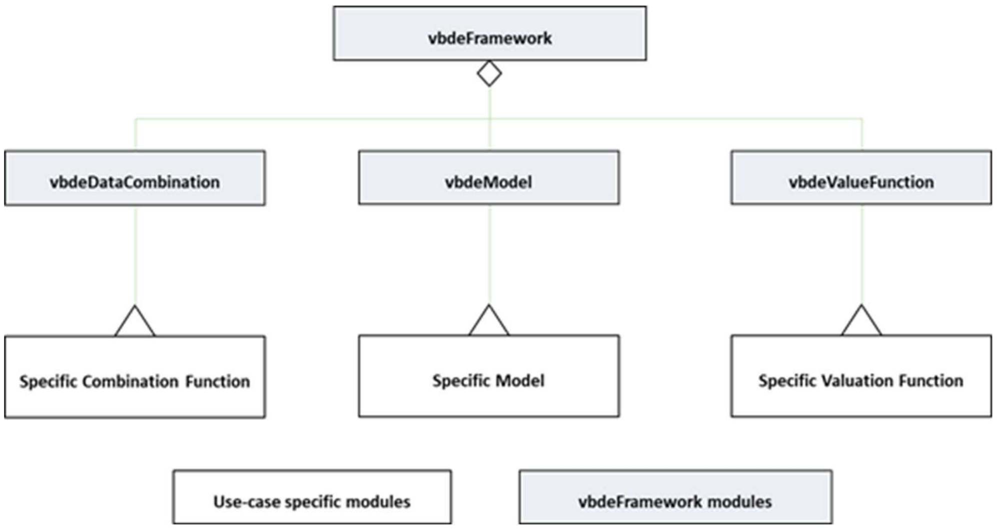


Figure 7 Internal design of the value-based data evaluation framework

The framework defines a series of interfaces that models and algorithms must implement to use the sandbox framework. If such interfaces are implemented to respect some restrictions that will be stated later, the framework's functions will be able to operate the model. The following assumptions are made:

- We know the list of potential sources to be used in the specific problem: S,
- We can retrieve and combine data from a combination of these sources d(S).
- We know, and we are able to run a certain model (M) and an accuracy valuation function (a) in a neutral sandbox. Potential buyers provide both "M" and "a" to check specific data pieces' value to their specific processing tasks.
- "M" and "a" must implement the abstract interfaces defined in the framework for them to be plugged into the solution

- The combiner is provided by the marketplace and implements an abstract interface to be plugged into the overall framework. The combiner is in charge of injecting suitable data from the marketplace for the framework to feed the model and get the corresponding results.

4.4.1 Value-based data valuation framework

Figure 8 summarizes the functional operation of a value-based data valuation framework. The design was intentionally agnostic from the underlying model or algorithm.

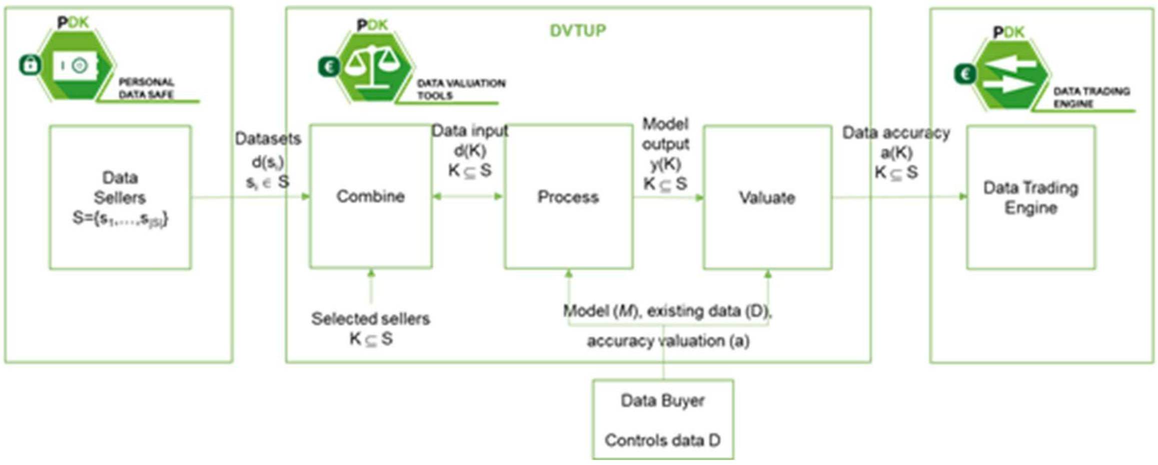


Figure 8 Data valuation framework

It assumes that a buyer who requires the marketplace to provide data for a certain data-driven task shares with the marketplace platform:

1. an algorithm or model (M) to be optimized that takes a series of inputs and provides an output y,
2. Any existing information already under the buyer's control to be plugged in the model (if not embedded in the model parameters),
3. an accuracy valuation function (a) that is able to measure how good or bad the outcome of the model. We will assume it is a similarity function a: y in [0,1].

We assume the marketplace controls, identifies, and selects a set of data sources (S), each contributing a dataset d(s) for all sources in set S whose data is suitable for the specific task. Using the framework, the marketplace will be able to process the model M for combinations of data inputs K from sellers d(K) = U d(k), for all k in K, and calculate the accuracy such inputs yield (a(K)). We assume that the buyer knows or is able to calculate the value of data for the specific task based on such accuracy. For example, eBay estimated that a 15% improvement on the recommender system translated into a 6% increase in revenues (i.e., an increment of $0.54 billion in 2016) [8].

We intend to enable a series of functions in data marketplaces leveraging the three former steps (combine, process and evaluate) of this data valuation framework so that the marketplace will be able to reward sellers participating in a transaction based on the accuracy data brings to a specific model. Users will be able to have transparent access to the relative valuations of their data. In order to accommodate the framework to work with different types of models and tasks, the following abstract components are needed:

- **vbdeDataCombination**: Abstract combiner interface defining functions to i) get a list of potential sources S, and ii) get a combined input d(K) for a subset K of S
- **vbdeModel**: Abstract model interface defining functions to i) initialize the model, ii) return the output of the model y(K) given an input d(K)
- **vbdeValueFunction**: Abstract value function interface that takes as an input y(K) and returns a float representing the accuracy (hence, the value) that sources in the set K bring to the model M.

The framework imposes the following restrictions:

- All input and output parameters of those functions are abstract serializable data,
- S and K are a set (list format) of unique source identifiers, be it numbers or strings, that corresponds to a single user, or is possible to be uniquely associated to the user or seller that provided the information,
- Classes implementing those interfaces must be compatible and interoperable, which means that:
    a. The output d(K) of the combiner can be plugged into the model as an input,
    b. The output of the model y(K) can be plugged into the valuation function.

The framework (**vbdeFramework**) will be used to encapsulate the concepts in **Figure 9,** according to the following UML diagram:

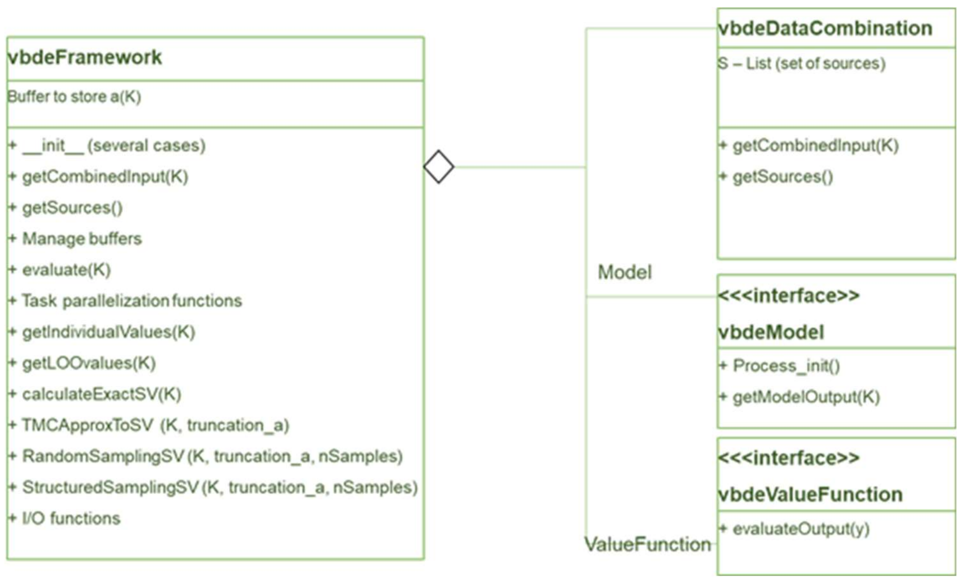


Figure 9 Design details of the value-based data valuation framework

4.4.2 Payment distribution function

Once a certain data transaction is closed by a data marketplace, involving a combination of data from different sources in a certain set P (d(P)), the resulting dataset is shared with the buyer, who is charged the amount of money agreed to close the transaction. At this time, it is needed functionality to break the corresponding amount of money and reward the sources that contributed to it. The payment division function's objective is to find a value assignment method, m_i, that captures the relative importance of the data originating from source s_i to the predictions of the model or algorithm (M). The payment division function would thus be a function of:

- the datasets used in d(P), $d(s_i)$ for i in P,
- the Model (M) together with the function that calculates the accuracy M yields to the buyer for different combinations of inputs, a(K), K subset of 2^S.

The sum of relative contributions of data sources in P must be equal to 1 and related to the accuracy that each data source brings to the buyer in that transaction's specific problem subject. The framework will implement different methodologies to calculate this breakdown based on the value of data, namely the following:

- Proportional to individual value,
- Proportional to LOO value,
- Proportional to Shapley value, including different methods to calculate or approximate to it.

Moreover, a payment distribution function based on a heuristic (f) is implementable by using the former framework, as well. For instance, one could think of distributing payments based on the number of individuals contributing to each dataset, or based on the average km per day their location information covers. In this case, a specific model must be available in the marketplace that calculates $f(s_i)$ based on data from individuals $d(s_i)$, and provides this $f(s_i)$ as the accuracy so to divide the payment based on the heuristic. This way the DVTUP model allows the TE to accommodate more simple use cases where the relative value of data can be approximated by certain characteristics of datasets shared with the buyer (e.g., volume, number of sources, or even equitable share).

According to the former design, the framework (vbdeFramework) will encapsulate all the functions needed to calculate the payment division function. Consequently, the framework needs to implement the following set of methods:

- Initialization of the framework (constructor) which will require the user to provide classes implementing all **vbdeDataCombination**, **vbdeModel** and **vbdeValueFunction** interfaces.
- Methods to manage an optional buffer to store a(K) and avoid repeated execution of the model for the same set of inputs. This buffer, which is specific to a certain task (model + value function), will be stored in the DVTUP server.
- Execute the whole value-based data valuation pipeline to get a(K), optimizing the processing time by previously checking the buffer.
- Parallelize the execution of works in different threads and processes to speed up the execution of algorithms.

- Retrieve information about the valuation of data sources, namely any of the following:
  - Get the value of a specific source s_i in S, $a(s_i)$,
  - Get the value of a set of sources K (subset of S), a(K),
  - Get an individual valuation of all the sources in S, by running the model using each of them exclusively and returning $a(s_i)$, s_i in S,
  - Get a leave-one-out valuation of sources in a set K, by running |S| times the model, using in execution i all the information except $d(s_i)$. The leave one out value of (LOO_i) is defined as: $LOO_i = a(S) – a(S – s_i)$,
  - Calculate the exact Shapley value of a source s_i in a set of source S' subset of S ($f(s_i)$), defined as the average marginal accuracy added by each source to each combination of the rest of sources in S':

$$\phi(s_i) = \sum_{K \subseteq S' - \{s_i\}} \frac{|K|!\,(|S'| - |K| - 1)!}{|S'|!} \cdot [a(K \cup s_i) - a(K)], s_i \in S' \qquad 4\text{-}1$$

  - Get the maximum value that the model could achieve by combining sources in a subset S' of S, which not necessarily requires using all the information:

$$a^* = \max_{K \subseteq 2^{S'}} a(K) \qquad 4\text{-}2$$

  - Get an approximation to the Shapley value using the following methods:
    - (Truncated) Montecarlo [10] [18]
    - Random sampling, [9]
    - (Truncated) Structured sampling [15]
- Store all the results of the execution to be shared or consulted by the users or the data management platform to explain the rewards a user got as a result of a specific transaction or a set of them. The results of the execution will include:
  - A unique identifier for the valuation (valuation_ID), that will be incrementally assigned by the DVTUP tool
  - The transaction ID, which refers to a key number that allows the TE to identify the transaction that generated the valuation, will be used to track valuation activity and relate it to operations made by the TE,
  - An identifier of the model used to calculate the payment division,
  - The set of sources s for which a relative valuation is provided (S’) ,
  - The method used to calculate the payment distribution, and

  - Final drivers used to calculate the payment distribution percentage among the set of sources S'

- Import and export methods to store or retrieve buffers or results from the storage used to keep all the data generated by transactions.

4.5 Interfaces

4.5.1 P-DS or PPA (input)

DVTUP will require using information from users or sellers to feed the task or model and get the value of every user a task or model. For that purpose, it will use the standard interfaces of the Personal Data Safe (P-DS), if personal information is directly used by the buyer's task or model, or the Privacy-Preserving Analytics (PPA) module, if just aggregated anonymized data from the PIMS user is required. Such standard interfaces ensure that DVTUP will receive data from users/sellers who consented their data to be used for the specific purpose of the (potential) data transaction. Besides, the combiner or the buyers' models may be implemented to use external data or processing to enrich the platform's information.

4.5.2 TE (input-output)

DVTUP is intended to be used by the Trading Engine. This section describes the JSON API block, which will serve as a communication interface between the DVTUP module and the Trading Engine.

The DTE API Module provides a REST API wrapper to the functionality of the value-based data evaluation framework, and manages a log of the operations of DVTUP. This module will act as the server of an internal REST API exposed to the data trading engine, that allows the API users:

1. Check the accuracy value of users, or sets of users, in the PIMs for a certain task
2. Launch valuation tasks
3. Check the situation and results of such valuation tasks

For that purpose, DVTUP exposes a REST API allowing the TE to execute the following commands:

Table 1 API REST interface calls between DVTUP and TE

Call	Description	Request body	Response
POST /accuracy	It responds with the accuracy achieved in the specific model by data from a specific set of users	JSON object with the following properties: (1)Transaction ID (2) Model ID (3) Set of users S’	Success status code (OK / Error) Body: accuracy valuation (float) [0,1]

POST /value	It triggers a work to determine the relative accuracy achieved in a specific model by each data source or aggregation of data sources by using the methodology stated in the parameter	JSON object with the following properties: (1) Transaction ID (2) Model ID (3) Set of users S' (4) Method to be used to calculate the relative value (5) Truncation value (6) Depth parameter (7) Stop condition	Success status code (OK / Error) Body: {value_id} of the valuation task, required to poll the server for results
GET /value/{value_id}	Polls the server for results of valuation task {value_id}. In case the task is over, it responds with the relative accuracy achieved in the specific valuation task by each data source or aggregation of data sources.	-	Success status code (OK / Error) Body: 1. {value_id} 2. Status 3. Model ID 4. Set of users S' 5. Method used 6. Truncation value 7. Depth parameter 8. Stop condition 9. Set of relative value of users in S

Notes regarding the inputs of REST API endpoints:

1. The transaction ID, which is passed as an input by the TE, will not be used nor processed by the DVTUP model, but just stored as part of the information about valuations for tracking purposes. Thus, a transaction may have several related valuations, each of them with its specific valuation ID.
2. The set of users is a JSON array including the ids of PIMCity users whose data has been traded as a result of the transaction. DVTUP API module will check that they

correspond to valid ids in the P-DS whose data can be collected in order to carry out the valuation. Should any user id not be valid, the API will return an 'invalid user' error as a response.

3. Model ID (integer) refers to the identification of the preloaded model the data will be used for. The valuation algorithms will use this model to calculate the value of users. In this version, the models include are the following (in parentheses the code to be user to call this model):

   a. Average value (0)

   b. Max value (1)

   c. Record count (2)

   d. Forecasting SARIMA model (3)

   DVTUP API module will check that the model ID corresponds to a valid model in the system. Otherwise, the API will return an 'invalid model' error as a response.

4. Method (in parentheses the code to be used to call this method, please refer to internal operation to get more information about each of them): In a first version, the DVTUP will support the following methodologies to retrieve the value of users' data:

   a. individual-value (0),

   b. LOO value (1),

   c. Exact Shapley value (2),

   d. (Truncated) Montecarlo (MC) approximation to Shapley value (3),

   e. (Truncated) Structured sampling (SS) approximation to Shapley value (4),

   f. (Truncated) Random sampling (RS) approximation to Shapley value (5)

   DVTUP API module will check that the model ID corresponds to a valid model in the system. Otherwise, the API will return an error as a response.

5. Truncation value (MC, SS, RS) specifies an accuracy threshold (float in [0,1]) that, if exceeded while evaluating a permutation of users s', will make the DVTUP stop the processing of incremental information, and assume that the rest of users in the permutation are adding no value to those in s'.

   If truncation value is equal to 1, non-truncated versions of MC, SS or RS algorithms will be run in order to get the value of user data.

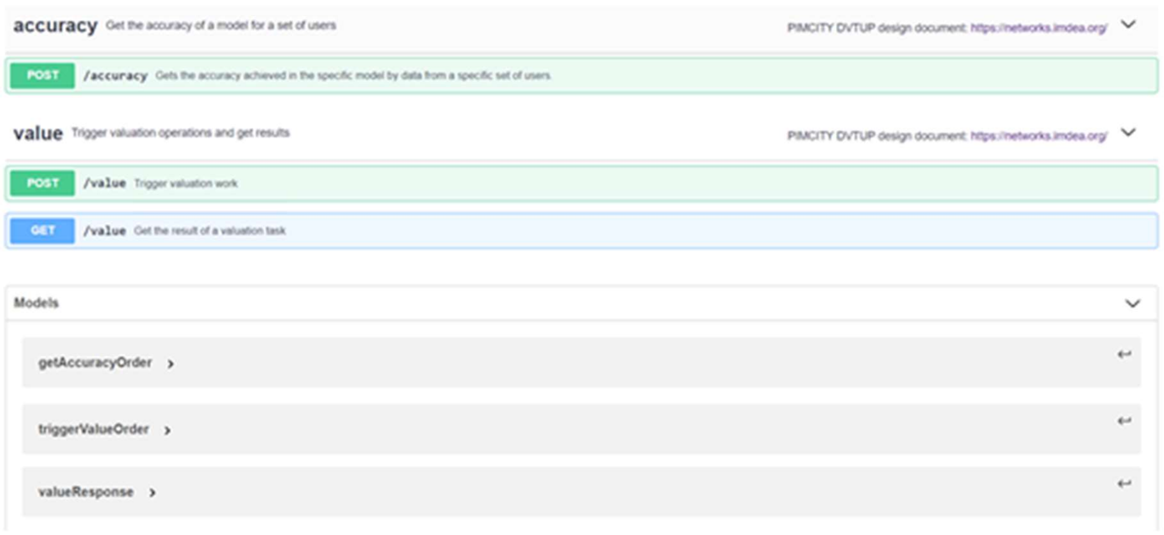
6. Depth parameter (SS, RS) defines the number of permutations (integer above 0) used to approximate Shapley value in sampling algorithms. In the case of using the SS, this value will be automatically rounded to the closest multiple of the total number of individuals in S'.

7. Stop condition (MC) defines the minimum percentage variation of Shapley value (float in [0,1]) in the last executions required for the DVTUP to continue evaluating new permutations.

The precision of any Shapley value approximation algorithm (MC, SS, RS) is given by the combination of the last three parameters. The lower truncation value, the shallower depth or the higher percentage in the stop condition will lead to coarser approximations.

4.5.3 DVTUP API REST Online Documentation Website

Even though it is an internal API, DVTUP documentation is available on an online interactive documentation website for the services it provides[6]. By visiting it, the reader can quickly understand and try out the API functionalities. It includes more detail, such as JSON schemas for inputs and outputs of all endpoints and services.



accuracy Get the accuracy of a model for a set of users — PIMCITY DVTUP design document: https://networks.imdea.org/

POST /accuracy Gets the accuracy achieved in the specific model by data from a specific set of users

value Trigger valuation operations and get results — PIMCITY DVTUP design document: https://networks.imdea.org/

POST /value Trigger valuation work

GET /value Get the result of a valuation task

Models

getAccuracyOrder

triggerValueOrder

valueResponse

Figure 10 Online documentation service

4.6 Configuration and Use

The preliminary software implementation of the Data Valuation Tools from End User Perspective can be found in the following directory of the PIMCity's Gitlab[7]. It includes examples (Jupyter notebooks) to show how to deal with the different APIs and components of the DVTUP.

4.7 Dependencies

DVTUP module is implemented in Python3, and requires that the following Python libraries are installed:

Numpy version 1.19.1 or above

[6] See https://gitlab.com/pimcity/wp3/dvtup/-/tree/master/DVTUP
[7] See https://gitlab.com/pimcity/wp3/dvtup/-/tree/master/DVTUP

Pandas version 1.1.1 or above

Dask version 2.25.0 or above

Flask version 1.1.2 or above

Statsmodels version 0.12.1 or above

Markupsafe version 1.1.1 or above

4.8 Configuration

All the aforementioned libraries are available to be installed by using pip ("pip install <library_name>").

4.9 Execution

To run the server, the following command must be executed from the folder where the source code is stored:

```
python DVTUPServer.py -p <PORT_NUMBER>
```

Where <PORT_NUMBER> specifies the port number the server will listen to. By default, DVTUP server will use port 5000. Consequently:

python DVTUPserver.py - starts a DVTUP server in localhost:5000

python DVTUPserver.py -p 5005 - starts a DVTUP server in localhost:5005

The server will check any dependencies from external PIMCity modules during the initialization process and prompt you if further action is needed. In this version, the module will work with preloaded models and data to show the feasibility of executing these valuations. Refer to the online documentation to test API calls to the server and use its functionality.

5 Trading Engine

5.1 Objective

In this section, we introduce PIMCity's Trading Engine, also known as TE. Its primary objective is to execute all transactions within the platform to exchange data for value in a secure, transparent, and fair-for-all way. Moreover, its key requirement is to be fully GDPR compliant. It must be able to receive Data Offers from Data Buyers and fulfil them with users from the PIMCity platform that fits within the target data sought and have proactively consented to share that data with that company for a specific purpose. The Trading Engine is presented as a REST API exposed to external Data Buyers and internal components from the PIMCity platform. Data Monetization over the internet is still brand new as regulations define new rights and responsibilities, opening new opportunities to include individuals in the value proposition. Therefore, there is no standard way of performing these transactions. With PIMCity's Trading Engine, we aim at:

- Defining a standard interface to carry out transactions where data is exchanged for a value between Data Buyers (i.e., companies and organizations) and End Users.
- Providing a REST API as an implementation of such a standard interface.
- Implementing state-of-the-art techniques using standard community-approved software libraries.
- Putting an integration flow which eases the deployment and allows for quick upgrades to the production environment tool.
- Making the component efficient, fast, and scalable to accommodate a huge number of transactions for a very large set of users and Data Buyers.

5.2 Background and state of the art

The trading engine is the backbone of the data monetization system. Different companies, especially start-ups, have been trying and are continuously testing different protocols to monetize their personal data. Three main engines are working on with data monetization, each offering a different solution reaching the same objective: making a data and advertisement ecosystem more transparent and balanced for the real owner of the data (people).

5.2.1 Wibson decentralized data marketplace

Wibson protocol [19] is a blockchain-based, decentralized data marketplace that provides individuals a way to securely and anonymously sell the information in a trusted environment. The combination of the Wibson token and blockchain-enabled smart contracts allows Data Sellers and Data Buyers to transact with each other directly while providing individuals the ability to maintain anonymity as desired. Wibson protocol intends that its data marketplace will provide infrastructure and financial incentives for individuals to sell personal information without sacrificing personal privacy securely. Data Buyers receive information from willing and actively participating individuals with the benefit of knowing that personal information should be accurate and current.

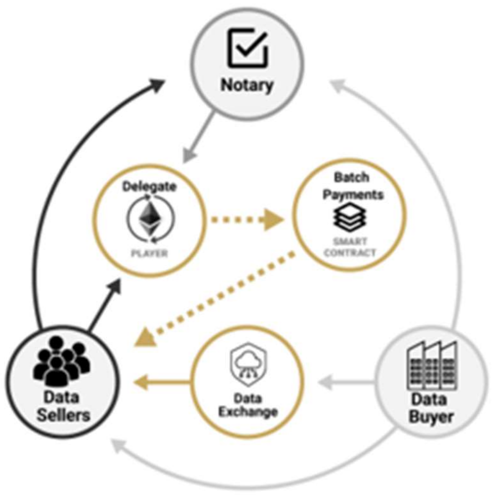


Figure 11 Wibson Protocol

5.2.2 Brave browser and ad protocol

Brave[8] is a fast, open-source, privacy-focused browser that blocks third-party ads and trackers and builds in a ledger system that measures user attention to reward publishers accordingly. Brave used BAT (Basic Attention Token) [20], a token for a decentralized ad exchange. The Brave browser knows where users spend their time, allowing them to calculate and reward publishers with BATs. This service creates a transparent Blockchain-based digital advertising market. Publishers receive more revenue because middlemen and fraud are reduced. Users are opt-in to an inclusive and rewarding private ad experience. And advertisers get better data on their spending.

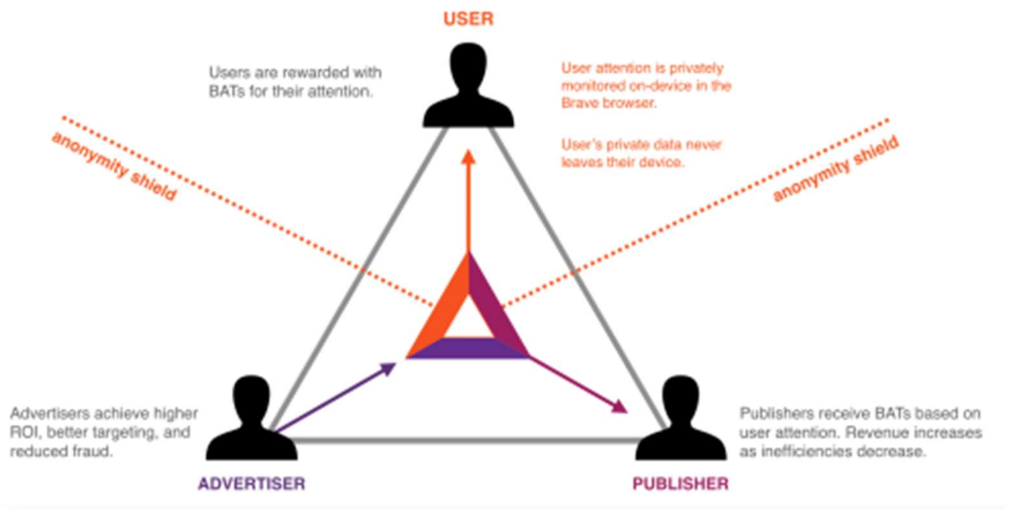

USER
Users are rewarded with BATs for their attention.
User attention is privately monitored on-device in the Brave browser.
User's private data never leaves their device.
anonymity shield
anonymity shield
Advertisers achieve higher ROI, better targeting, and reduced fraud.
ADVERTISER
Publishers receive BATs based on user attention. Revenue increases as inefficiencies decrease.
PUBLISHER

Figure 12 Brave implementation

[8] https://brave.com

5.2.3 Gener8 Ads

Gener8[9] enables people to take control of their advertising experience and earn money from the ads they see. Gener8 browser extension enables users to control and be rewarded for their data. It tailors all the ads that users see online to be based on their preferences. It works by stopping 3rd party trackers and ads and by showing Gener8 partner ads based on users' preferences and election. At the same time, users gain points to exchange this on their marketplace.

5.3 Technical Design

5.3.1 Basic Functionality

The Trading Engine might interact with a total of six other PIMCity components. Although it implies an important integration effort, it is needed to execute a secure, transparent, and fair-for-all transaction of data in exchange for value.

For example, when a Data Buyer wants to place a Data Offer, the Trading Engine does the following:

1. Gets the price for the audience or data being bought from the Data Valuation Tools.
2. Calculates how many people fit in the budget.
3. Gets the certified list of users with active consents.
4. Fetches their data. If the data size is too big to handle at once, streams are used.
5. Cleans the data through the Data Provenance.
6. Handles back the data and updates credits in the accounts.

The following **figure** shows such a flow:

[9] https://gener8ads.com/products/ads

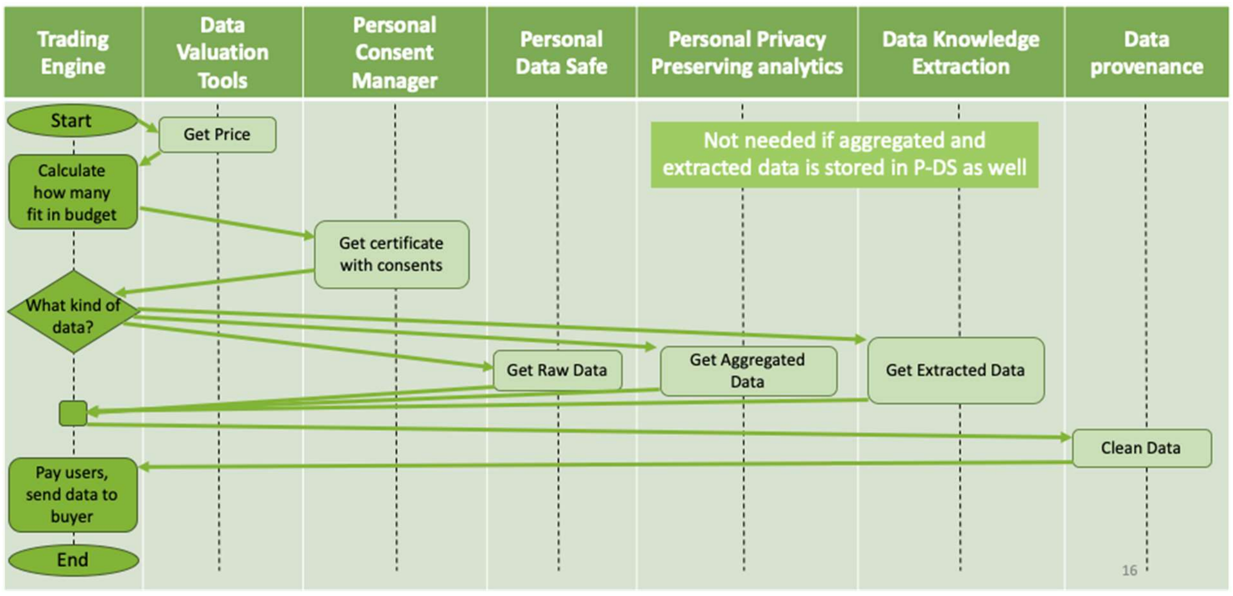

Trading Engine
Data Valuation Tools
Personal Consent Manager
Personal Data Safe
Personal Privacy Preserving analytics
Data Knowledge Extraction
Data provenance
Start
Get Price
Calculate how many fit in budget
Not needed if aggregated and extracted data is stored in P-DS as well
Get certificate with consents
What kind of data?
Get Raw Data
Get Aggregated Data
Get Extracted Data
Clean Data
Pay users, send data to buyer
End
16

Figure 13 Trading Engine algorithm

To carry out these transactions, Data Buyers have to buy credits upfront on a platform different and independent from PIMCity, i.e., before creating any Data Offer. Those credits will then be used to pay End Users for their data. On the other hand, end-users earn credits every time they monetize their data through PIMCity. Those credits can be exchanged later on for rewards and/or real money.

5.4 Internal Operation

The Trading Engine is exposed as an API REST that works with JSON documents to receive requests and send responses. Then, before the actual request is processed, the API checks if the user has the proper role and access to that resource. Finally, the Trading Engine applies the business logic needed to solve the request and serve back a response. The Trading Engine is used by Data Buyers, End Users, and the general public. With that in mind, the following Roles and Accesses are defined.

5.4.1 Roles and Access

The API is open with three different access levels:

- To the public,
- To End Users,
- To Data Buyers,

Therefore, the API defines two different roles:

- End-User,
- Data Buyer.

All endpoints related to placing Data Offers in the platform or the ability to get the current price for a certain audience and data type are restricted to the Data Buyer role. Apart from that, an Authorization header is required to allow only the account owner to create Data Offers on its behalf. On the other hand, the endpoint that shows the number of credits available in the account is restricted to the Data Buyer and End-User roles. Finally, endpoints related to the service's general status, like exposing if it is online or which version is deployed, are open to the public.

5.4.2 Architecture

In order to carry out the business logic, the Trading Engine is separated into three different sub-components:

- API REST: The service is exposed to the different roles that receive the requests and applies the business logic until a response can be served back.
- Message Broker: A queue of jobs to be performed asynchronously to carry out the heavy-weight tasks that the API cannot perform within the request because they would time out or block the whole service.
- Database: Key-Value store to persist the state of the Trading Engine. It can be accessed only by the API or Message Broker.

5.4.3 Data Model

The Trading Engine uses two Key-Value stores to persist in the business state in the following fashion:

- Users Store: It stores the name, email, account creation date, and credits for every user.
- Data Offers Store: It stores all the information about each offer created with the Trading Engine. The audience, the data type, the buyer information, the purpose, the terms and conditions, the budget, and the offer general status.

5.5 Interfaces

The Trading Engine is exposed via a REST API to all its users, i.e., Data Buyers, End Users, and the public. Those users will communicate with the Trading Engine by sending requests to the endpoints detailed below. Such requests might be able to include JSON-like documents for the users to provide all the needed information to be processed during the corresponding operation. At the end of each transaction, the API will answer back with another JSON-like document as a response. The REST API for the Trading Engine offers the following capabilities:

- Account Management: Read profile details such as email, name, and avatar, add and read available credits to the account.
- Market Data: Shows the price for a piece of data for a specific audience and enables Data Buyers to place new offers within the market.
- Service Status: Anyone can check if the service is up or not.

Data Buyers and end-users have access to all three capabilities, although they can only manage and read their account details. Moreover, Data Buyers are the only ones capable of creating new Data Offers. The public, as well as Data Buyers and end-users, can check the service status to see if the Trading Engine is online or not. This service is used by:

- Data Buyers dashboard (Easy PIM),
- End-User.

This service uses:

- Data Valuation Tools,
- Personal Consent Manager (P-CM),
- Personal Data Safe (P-DS),
- Personal Privacy-Preserving analytics (P-PPA),
- Data Knowledge Extraction Tools (DKE),
- Data Provenance.

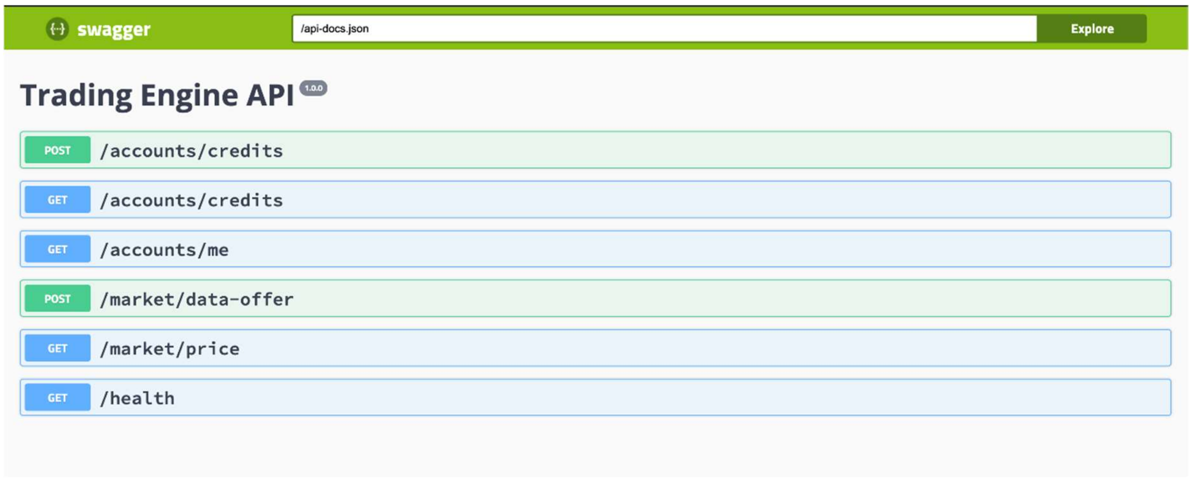
All these components offer an API REST that the Trading Engine uses to connect and exchange messages. Payloads are in JSON format, requests, and responses alike. This becomes efficient in reducing the technologies used and potential bugs. In general, final responses are obtained when the request is resolved, but the Trading Engine is also prepared to submit a callback endpoint when the task to be performed by any other component is heavy-weight and needs to be executed offline. Then, the result can be informed using the callback sent previously.

5.5.1 API REST Online Documentation Website

Since the API can be used directly by Data Buyers or End Users, it ends up being powerful and handy to have an online interactive documentation website for the service. By visiting it, the reader can quickly understand and try out the API functionalities.

Link to the documentation:

https://gitlab.com/PIMCity/wp5/open-api/-/blob/master/WP3/data-trading-engine.yml



Trading Engine API

```
POST /accounts/credits
GET  /accounts/credits
GET  /accounts/me
POST /market/data-offer
GET  /market/price
GET  /health
```

Figure 14 Online documentation service

5.6 Configuration and Use

The preliminary software implementation of the Trading Engine can be found in the following directory of the PIMCity's Gitlab:

https://gitlab.com/pimcity/wp3/datatradingengine/-/tree/master/DataTradingEngine

5.7 Dependencies

- Docker Engine: v20.10.x
- Docker Compose: v1.27.x

5.8 Configuration

The Trading Engine can be configured in two similar ways:

1. Using environment variables.
2. Defining those environment variables in a file called ".env" at the root directory of the folder with the PDK deployed, declaring them in the same way a variable is defined in the UNIX shell. For example:

```
NODE_ENV=production
PORT=9000
HOST=localhost
CACHE=enabled
ERROR_LOG=/path/to/error.log
COMBINED_LOG=/path/to/combined.log
```

The variables (in no particular order) are the following:

- NODE_ENV: development | staging | production
- PORT: the port to be used to run the service
- HOST: the host to be used to run the service (for example: “localhost”)
- ERROR_LOG: the path to the file for logging errors.
- COMBINED_LOG: the path to the file for logging all messages.
- LEVEL_DIRECTORY: the directory where LevelDB instance is stored.

5.9 Execution

To run the Trading Engine service, the following command should be executed at the root of the repository:

```
$ docker-compose up
```

6 OpenAPI documentation for WP3 modules

Please find below the link to API documentation of the modules:

6.1 OpenAPI documentation for DVTMP

https://gitlab.com/PIMCity/wp5/open-api/-/blob/master/WP3/DVTMP.yaml

6.2 OpenAPI documentation for DVTUP

https://gitlab.com/PIMCity/wp5/open-api/-/blob/master/WP3/DVTUP.yaml

6.3 OpenAPI documentation for TE

https://gitlab.com/PIMCity/wp5/open-api/-/blob/master/WP3/data-trading-engine.yml

7 Conclusions

This document described the final detailed design of the PIMCity modules responsible for estimating the value of the users' data and the module responsible for trading the users' data with interested data buyers. In particular, this deliverable describes the internal functionality of each module as well as its communication interfaces. The modules whose first complete design has been described in this document are:

- Data Valuation Tools from Market Perspective (DVTMP), from Task 3.2, aims to estimate audiences' value from different advertising platforms.

- Data Valuation Tools from User Perspective (DVTUP), from Task 3.2., aims at estimating the value of end-users' data using the value-based fairness principle.
- Trading Engine (TE), from Task 3.2, a key piece in the PIMCity architecture responsible for trading end-user's data with interested third party buyers.

Moreover, following the design of each of the modules, we deliver a preliminary version of the software implementing: the DVTMP and the API implementing its communication interface; the DVTUP and the API implementing its communication interface; the TE and the API implementing its communication interface.

Both the design and the implementation of these modules and their APIs have been done collaboratively between the WP3 members, guaranteeing an easy integration. The different WP3 partners have carefully reviewed the document. Following the final design of the WP3 modules, their preliminary software version (delivered with this deliverable) will be further extended and improved in the next months. The final version of the software implementing each of the WP3 modules will be delivered in Deliverable 3.4.

8 References

[1] González Cabañas, J., Cuevas, A., & Cuevas, R. (2017, May). FDVT: Data valuation tool for Facebook users. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (pp. 3799-3809).

[2] Green, P. E., & Srinivasan, V. (1978). Conjoint analysis in consumer research: issues and outlook. Journal of consumer research, 5(2), 103-123.

[3] Green, P. E., & Srinivasan, V. (1990). Conjoint analysis in marketing: new developments with implications for research and practice. Journal of marketing, 54(4), 3-19.

[4] Gustafsson, A., Herrmann, A., & Huber, F. (2013). 1 Conjoint Analysis as an. Conjoint Measurement: Methods and Applications, 5.

[5] Feijóo, C., Gómez-Barroso, J. L., & Voigt, P. (2014). Exploring the economic value of personal information from 'firms' financial statements. International journal of information management, 34(2), 248-256.

[6] Acquisti, A., John, L. K., & Loewenstein, G. (2013). What is privacy worth?. The Journal of Legal Studies, 42(2), 249-274.

[7] Agarwal, A., Dahleh, M., & Sarkar, T. (2019, June). A marketplace for data: An algorithmic solution. In Proceedings of the 2019 ACM Conference on Economics and Computation (pp. 701-726).

[8] Brovman, Y. M., Jacob, M., Srinivasan, N., Neola, S., Galron, D., Snyder, R., & Wang, P. (2016, September). Optimizing similar item recommendations in a semi-structured marketplace to maximize conversion. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 199-202).

[9] Castro, J., Gómez, D., & Tejada, J. (2009). Polynomial calculation of the Shapley value based on sampling. Computers & Operations Research, 36(5), 1726-1730.

[10] Ghorbani, A., & Zou, J. (2019). Data shapley: Equitable valuation of data for machine learning. arXiv preprint arXiv:1904.02868.

[11] Carrascal, J. P., Riederer, C., Erramilli, V., Cherubini, M., & de Oliveira, R. (2013, May). Your browsing behavior for a big mac: Economics of personal information online. In Proceedings of the 22nd international conference on World Wide Web (pp. 189-200).

[12] Chen, L., Koutris, P., & Kumar, A. (2019, June). Towards model-based pricing for machine learning in a data marketplace. In Proceedings of the 2019 International Conference on Management of Data (pp. 1535-1552).

[13] Reimsbach-Kounatze, C., Reynolds, T., & Stryszowski, P. (2013). Exploring the economics of personal data-a survey of methodologies for measuring monetary value.

[14] Fernandez, R. C., Subramaniam, P., & Franklin, M. J. (2020). Data market platforms: Trading data assets to solve data problems. Proceedings of the VLDB Endowment, 13(12), 1933-1947.

[15] Azcoitia, S. A., Paraschiv, M., & Laoutaris, N. (2020). Computing the Value of Spatio-Temporal Data in Wholesale and Retail Data Marketplaces. arXiv preprint arXiv:2002.11193.

[16] Shapley, L. S. (1953). A value for n-person games. Contributions to the Theory of Games, 2(28), 307-317.

[17] Yan, T., & Procaccia, A. D. (2020). If You Like Shapley Then 'You'll Love the Core.

[18] Ghorbani, A., & Zou, J. (2019). Data shapley: Equitable valuation of data for machine learning. arXiv preprint arXiv:1904.02868.

[19] Fernandez, Daniel, Ariel Futoransky, Gustavo Ajzenman, Matias Travizano, and Carlos Sarraute. "Wibson protocol for secure data exchange and batch payments." arXiv preprint arXiv:2001.08832 (2020).

[20] Token, B. A., & PRIMER, L. (2018). Basic Attention Token. BasicAttentionToken.[Online]. Available: https://www. basicattentiontoken. org.[Accessed: 09-May-2019].

[21] Acquisti, A., Friedman, A., & Telang, R. (2006). Is there a cost to privacy breaches? An event study. ICIS 2006 Proceedings, 94.

9 Appendix

In the appendix, we put the status of the software repositories as available at the time of finalizing this document

Update README.md
Dani Fernandez authored 2 days ago

3f6d45dd

README.md 3.85 KB

Data Trading Engine

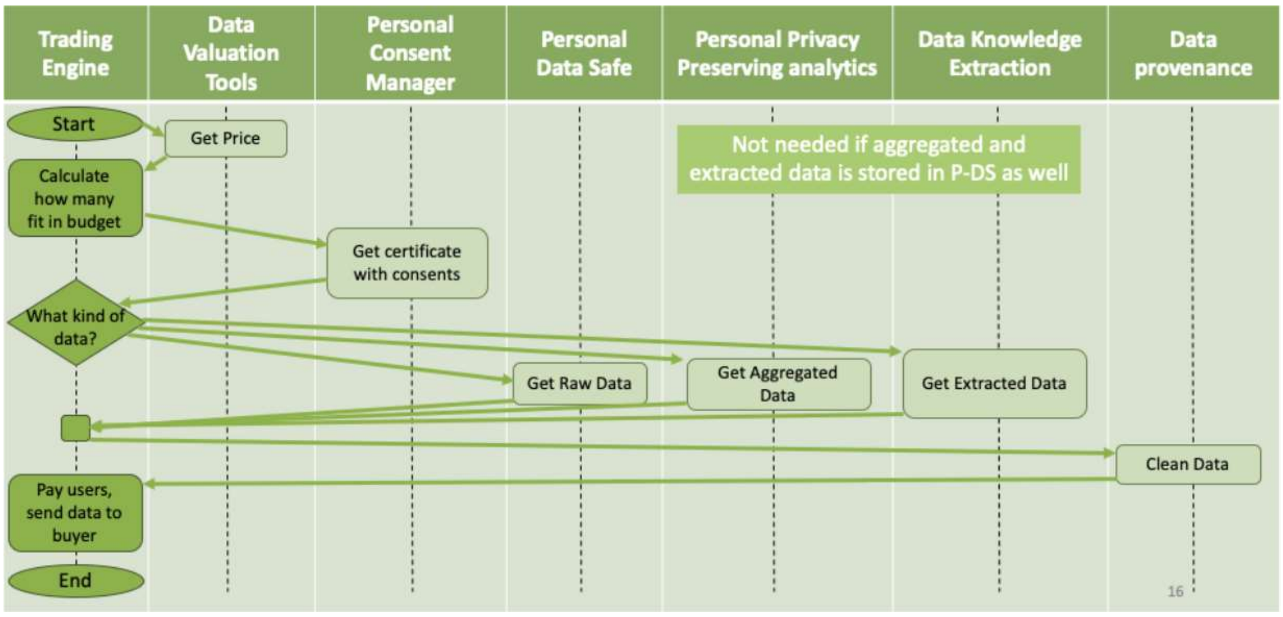
Introduction

The Data Trading Engine (TE) is responsible for trading the data of users registered/handled by the PIM with interested buyers. Hence, the TE serves as a communication interface between the PIM backend and the data buyers. There is a myriad of data types that can be sold. The TE focuses in bulk data and audience data.

Bulk data is typically bought in a non-real-time manner to receive information from a (typically large) group of users. Some examples include: a health insurance company may be interested in people's medical records; a car insurance company may be interested in the people's mobility data to understand who drives through tough roads or at higher speeds; a mortgage-issuing company may be interested in people's financial records, etc. None of these data need to be traded in real-time, and typically data buyers try to buy it in bulk.

An audience is a term used in marketing to refer to a specific group of the population defined by three parameters: location (where the user is located); demographic information (age, gender, etc.); interests (a list of interests, e.g., outdoor activities, sports, science, automobile, etc.). Most of these parameters are typically extracted from well-known taxonomies such as the one offered by IAB (which is a de-facto standard in digital marketing). Online advertising (a.k.a. digital marketing) is arguably one of the most important businesses exploiting data utilization, specifically audience data, to deliver targeted ads to users in real-time.

The TE is exposed as a REST API which can be accessed by other components in PIMCity or external players like Data Buyers and individuals. The creation of a transaction is depicted in the figure below.



Trading Engine
Data Valuation Tools
Personal Consent Manager
Personal Data Safe
Personal Privacy Preserving analytics
Data Knowledge Extraction
Data provenance
Start
Get Price
Not needed if aggregated and extracted data is stored in P-DS as well
Calculate how many fit in budget
Get certificate with consents
What kind of data?
Get Raw Data
Get Aggregated Data
Get Extracted Data
Clean Data
Pay users, send data to buyer
End
16

Installation

The documentation and the following instructions refer to a Linux environment, running with **Docker Engine v20.10.x** and **Docker Compose: v1.27.x**. The TE project has been cloned from this GitLab repository.

Follow accurately the next steps to quickly set-up the TE backbone on your server. All relevant steps are designed for a Linux machine, perform the equivalent procedure with other environments.

1. Prepare the environment:

```
> sudo apt-get update
> sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

```
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring
> echo \
  "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
> sudo apt-get update
> sudo apt-get install docker-ce docker-ce-cli containerd.io
```

2. Import the project from the GIT repository:

```
> git clone https://gitlab.com/pimcity/wp3/datatradingengine.git
```

Execution

To run the Trading Engine service, the following command should be executed at the root of the repository:

```
> docker-compose up
```

Configuration

The Trading Engine can be configured in three similar ways:

1. Using environment variables.
2. Defining those environment variables in a file called ".env" at the root directory of the folder with the PDK deployed, declaring them in the same way a variable is defined in the UNIX shell. You can check this example.
3. Modifying docker-compose.yml file.

Usage Examples

You can check here the Swagger OpenAPI definition to see how the API is defined and used. Examples are provided as well.

License

The TE is distributed under AGPL-3.0-only, see the LICENSE file.

Copyright (C) 2021 Wibson - Daniel Fernandez, Rodrigo Irarrazaval

Update README.md
amir.mh authored 3 days ago

7525368d

README.md 2.83 KB

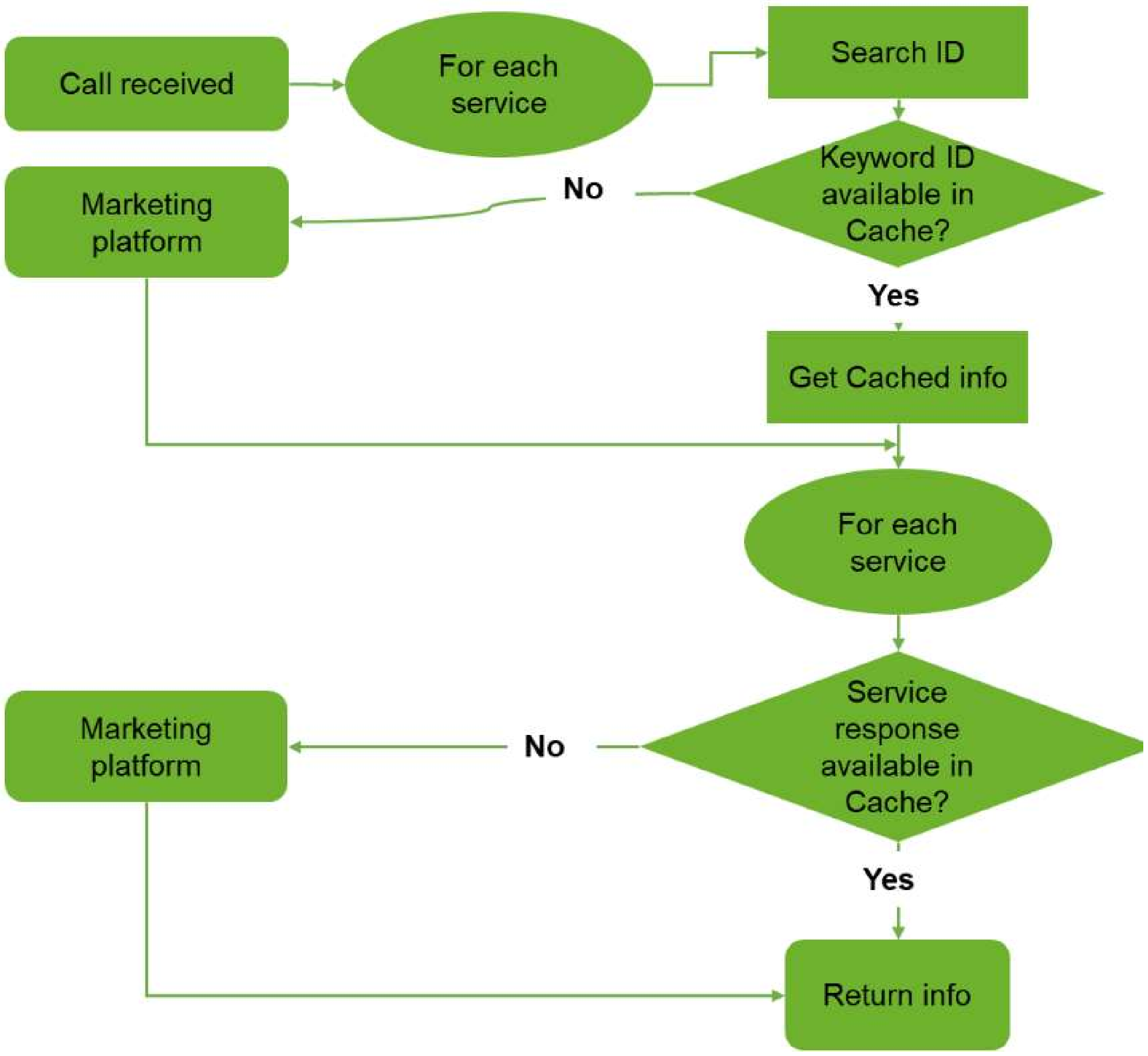
Data Valuation Tool from Market Perspective (DVTMP)

Data valuation tool provides an estimation for the audiences in online social media advertising platforms (Facebook, instagram and linkedin)

Introduction

The Data Valuation Tools from the market perspective (DVTMP) module developed in PIMCity will leverage some of the most popular existing online advertising platforms to estimate the value of hundreds to thousands of audiences. The DVTMP module aims to provide the monetary value of audiences traded on the main online advertising platforms. This will serve any PIM deciding to implement the DVTMP module to have a realistic estimation of audiences' value to be traded. PIMs which implement a more accurate value estimation module will have a competitive advantage in the market. Since the information about values collected from these advertising platforms is based on aggregated historical pricing data, we can assert that DVTMP provides full-privacy guarantees. Moreover, a given audience's value can be obtained in real-time from the referred online advertising platform. In particular, the design of the DVTMP pursues the following objectives:

1. Crawling data value of audiences from Facebook, Instagram, and LinkedIn
2. Process, clean, and curate the collected data
3. Store processed data
4. Provide access to the data through an API



Installation

The requirements of this project are:

Requirements

Docker Engine: v20.10.x Docker Compose: v1.27.x

Installation Command

```
cd /Build/request_handler
docker-compose up
```

Usage

The server is located on port 3700 by default and this address can be changed in "dvtmp/Build/request_handler/docker-compose.yml"

The server recieves a POST request with an audience description and responses with a value estimation using Facebook, Instagram and LinkedIn platforms. e.g.

```
import json
url = '127.0.0.1:3700/auValuator'

body = {
    "min_age" : 18,
    "max_age": 50,
    "interest": ["advertising"],
    "location": "IT",
    "platform":"Facebook",
    "priceType":"CPM",
    "gender":"male"
}

r=requests.post(url, data=json.dumps(body))
```

The response (r) will be in the following form:

```
r= {
"status": "success",
"data": {
"audience_validated": {
"max_age": 50,
"min_age": 18,
"interest": [
"advertising"
],
"location": "IT"
},
"Platform": "Facebook",
"info": {
"title": "Data Evaluation tool Market Perspective",
"version": "1.1.0"
},
"servers": {
"url": "http://localhost:37000",
"description": "local server"
},
"tags": {
"name": "auValuator",
"description": "operation in the local database"
},
"value": 3.892516819904835,
"currnecy": "USD"
}
}
```

License

moved
Martino Trevisan authored 10 minutes ago

2f9a4542

readme.md 4.19 KB

Data Valuation Tool from the User's Perspective

Intro

Welcome to PIMCITY's Data Valuation Tool from the User's Perspective source code. We are still in developing stage of the tool, please find more information in the corresponding readme.md files of each version.

The objective of the Data Valuation Tools from an End-User Perspective (DVTUP) module is to provide estimated valuations of end-users' data for the dataset they are selling through the marketplace as bulk data. In particular, DVTUP will provide tools for the TE to:

1. Provide buyers with a hint of how valuable a piece of data is for a certain type of model or even for a specific task.
2. Calculate a fair breakdown of data transaction charges by seller, looking forward to rewarding each user proportionally to the value that each piece of data from different sellers brings to the buyer for a specific task.

In the first case, the output will be the expected accuracy the buyer will get from a dataset if purchased from the marketplace. In the second case, the output will estimate the percentage of a transaction value that corresponds to each seller, and a log of data and results obtained to justify rewards paid to different sellers. For that purpose, different methodologies and algorithms will be designed and implemented to allow data marketplaces to calculate such breakdown in different ways, namely: i) using traditional heuristics such as data volume or the number of sources, ii) using a value-based data evaluation framework to test data on the specific task the buyer is intended to use it in, and breaking the payment according to a value function (e.g., accuracy yielded by a given model or algorithm) provided by the buyer.

This is a preliminary software implementation of the Data Valuation Tools from End User Perspective.

Installation

DVTUP module is implemented in Python3, and requires that the following Python libraries are installed:

- Numpy version 1.19.1 or above
- Pandas version 1.1.1 or above
- Dask version 2.25.0 or above
- Flask version 1.1.2 or above
- Statsmodels version 0.12.1 or above
- Markupsafe version 1.1.1 or above

All the aforementioned libraries are available to be installed by using pip (“pip install <library_name>”).

To run the server, the following command must be executed from the folder where the source code is stored:

python DVTUPServer.py -p <PORT_NUMBER>

Where <PORT_NUMBER> specifies the port number the server will listen to. By default, DVTUP server will use port 5000. Consequently:

```
python DVTUPserver.py - starts a DVTUP server in localhost:5000
python DVTUPserver.py -p 5005 - starts a DVTUP server in localhost:5005
```

The server will check any dependencies from external PIMCITY modules during the initialization process, and will prompt you in case further action is needed. In this version, the module will work with preloaded models and data to show the feasibility of executing these valuations. Refer to the online documentation to test API calls to the server and use its functionality.

Usage examples

Please refer to the following Jupyter notebooks which include examples to show how to deal with the different APIs and components of the DVTUP.

- vbde Sample Test.ipynb - shows how to deal with the value-based data valuation framework and their abstract classes, and how to calculate the relative value of data sources in different sample cases.
- DVTUP Client.ipynb shows how to start the server and use a REST HTTP client to send valuation requests and get their results.

License