Data-Sharing Markets: Model, Protocol, and Algorithms to Incentivize the Formation of Data-Sharing Consortia

Raul Castro Fernandez
The University of Chicago
raulcf@uchicago.edu

ABSTRACT

Organizations that would mutually benefit from pooling their data are otherwise wary of sharing. This is because sharing data is costly—in time and effort—and, at the same time, the benefits of sharing are not clear. Without a clear cost-benefit analysis, participants default in not sharing. As a consequence, many opportunities to create valuable data-sharing consortia never materialize and the value of data remains locked.

We introduce a new sharing model, market protocol, and algorithms to incentivize the creation of data-sharing markets. The combined contributions of this paper, which we call DSC, incentivize the creation of data-sharing markets that unleash the value of data for its participants. The sharing model introduces two incentives; one that guarantees that participating is better than not doing so, and another that compensates participants according to how valuable is their data. Because operating the consortia is costly, we are also concerned with ensuring its operation is sustainable: we design a protocol that ensures that valuable data-sharing consortia form when it is sustainable.

We introduce algorithms to elicit the value of data from the participants, which is used to: first, cover the costs of operating the consortia, and second compensate data contributions. For the latter, we challenge the use of the Shapley value to allocate revenue. We offer analytical and empirical evidence for this and introduce an alternative method that compensates participants better and leads to the formation of more data-sharing consortia.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Raul Castro Fernandez. 2018. Data-Sharing Markets: Model, Protocol, and Algorithms to Incentivize the Formation of Data-Sharing Consortia. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (SIGMOD'23).* ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

SIGMOD'23, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Whenever data is shared, transformation ensues. Combining data often increases its value for machine learning, causal inference, and other data-driven tasks used for decision-making. For example, combining research data accelerates discovery [15], organizations pooling data may train better machine learning models to help with problems such as threat discovery [17], finding new drugs [52], and more, and social scientists with access to data from more agencies can assess the causal impacts of policies more precisely [19]. Despite the upsides of sharing data, many organizations will not share because establishing a sharing consortium is a long, tedious, and costly process, while the benefits it brings are often uncertain. For example, consider the following examples:

Sharing to improve patient care. After addressing the legal and regulatory challenges, hospitals still face a technical and incentive challenge: will the benefits to patient care outweigh the price of overcoming these challenges?

Sharing to improve fraud detection. Financial institutions could pool their data to build better fraud detection mechanisms. Before sharing their data, they will ask whether the future benefits will be at least as good as the value their data will produce to others.

The uncertainty of the examples is common in many scenarios, often resulting in a "default" decision of not sharing. The consequence is that many data-sharing opportunities remain unexploited. Because there is an increasing number of scenarios that would benefit from data access but do not materialize because of the aforementioned uncertainty, incentivizing the formation of data-sharing consortia is an important data management challenge.

In this paper, we introduce a new sharing model, market protocol, and algorithms that, together, incentivize the creation of data sharing markets; we call this DSC (Data-Sharing-Consortia). The protocol uses two incentives to stimulate the formation of data-sharing consortia. First, participants are guaranteed that the payoff of sharing data is higher than not sharing, e.g., they will obtain a more accurate fraud detector; we call this incentive **sharing dominance**. Second, participants are compensated according to how much the data they share contributes to consortium's payoff; we call this incentive **entitlement stake**. These two incentives create a positive feedback loop: participants want to contribute better data to capture a larger compensation, and better data, in turn, translates into higher task's payoff for every participant. Designing such a protocol requires tackling important technical challenges:

- **Implementing sharing dominance and entitlement stake incentives.** To implement sharing dominance, the protocol must *signal* participants the value they will perceive. To implement entitlement stake, the protocol must elicit the value from participants so it can distribute it back to them.

- **Robustness Against Bad Inputs.** When a participant's data deteriorates the quality of the consortia, such a participant is detected to avoid harming the consortium. These *bad inputs* include dirty data, strategically crafted datasets, and more.
- **Sustainability.** Operating the protocol has a cost that must be covered to sustain the market over time; otherwise the consortium ceases to exist. The protocol covers costs from the revenue it raises from participants, before using the remaining revenue to implement the entitlement stake incentive.

Designing a protocol that always raises revenue higher than the operation costs is impossible in these kinds of data markets according to the Myerson-Satterthwaite theorem [55]. Instead, we design a protocol with the following desirable guarantee: **if the consortium is worth forming, the protocol makes it sustainable, if the consortium is not sustainable, it is not worth forming**. Thus, we ensure only beneficial consortia is formed.

The main contribution of the paper to data management is the protocol and its implementation, which makes use of algorithms to address challenges of data markets, including allocation, elicitation, and revenue allocation that are often only discussed using mathematical models. Concretely, the algorithms solve core technical questions such as: i) how to extract value from participants to cover operation costs—using optimal auction techniques from the field of mechanism design [54]; ii) how to confer robustness against bad inputs—no matter whether dirty data, strategic inputs, or chance—by including a detection mechanism in the protocol; and iii) how to distribute the remaining value back to participants to compensate them for their data contributions—arguing against the use of the now commonly employed Shapley value [63] and introducing a more efficient and sustainable alternative.

We demonstrate, using real data and tasks, that the the cost of operating the consortia with the new DSC market is much lower than when using alternative functions thus leading to more consortia forming. We demonstrate the effectiveness of the new algorithms to tackle the challenge of bad inputs, including bad quality and strategic data. Finally, we demonstrate the qualities, both in accuracy and runtime, of the new data compensation algorithm when compared to existing state of the art Shapley value-based solutions [34].

Sharing landscape. To scope our work, it is useful to differentiate between three sharing scenarios. In the first, there exists an external incentive to share, e.g., governments sharing open data for increased transparency [41] or requiring making scientific results public [58]. In these cases data is shared without further intervention. In the second, there exists a disincentive to share, e.g., sensitive or secret data, individuals (private) data. In these cases, no intervention will cause sharing. In the third group, there is no external incentive or disincentive and sharing becomes a matter of weighing benefits versus effort: this is the case we target by designing data markets that incentivize participants to participate when it is beneficial.

Data sharing and data integration. If participants decide to share, they need to combine their data, which requires preparation. In many sharing scenarios, this combination task is much simpler than solving the general data integration problem. In general data integration, N databases must be integrated together, e.g., after a merge and acquisition (M&A). In a sharing scenario, active participants curate what data to share, often only a small portion of all data they possess, and they are incentivized to put in the effort to prepare their data so as to facilitate its combination. Our paper concentrates on the methods to incentivize participation, and not on the technology to combine datasets. As in other work, we assume data can be combined [12, 68].

Related work. Federated databases address the problem of physically connecting a set of nodes and enabling querying over the joint data. Data federations combine federated databases with the security aspect of federated learning: that no data must leave its owner's premise. Federated databases and Data Federations deal with the *"how"* of data sharing. Our work is orthogonal and complementary to these efforts: we deal with the problem of incentivizing participants to share in the first place (the *"why"*). This is a problem that data markets can address and that the data management community is uniquely positioned to tackle [1, 33, 49, 51].

The rest of the paper is organized as follows. Section 2 introduces the setup and notation. Section 3 introduces the sharing model. Section 4 presents the data-sharing protocol. Section 5 presents a new revenue allocation function. Section 6 discusses how the protocol tackles strategic behavior. We present evaluation results in Section 7, related work in Section 8, and conclusions in Section 9.

2 SETUP AND NOTATION

A data-sharing consortium consists of a set of $i \in N$ participants who decide to collaborate in solving a data-driven task of common interest. They collaborate by contributing each their own data, D_i. A data-driven task, $\mathcal{T}$, uses data to answer queries. Some examples of data-driven tasks are machine learning, causal inference, information retrieval, and analytical queries. The quality of the query answers is measured with respect to a benchmark, B_i.

A task's lifecycle consists of three stages: *setup, query*, and *evaluate*. During the *setup* stage, a task takes an input dataset, D_i, and returns a task instance, $T = \mathcal{T}(D_i)$. During the *query* stage, a task instance takes a query x and returns an answer $T(x)$. During the *evaluate* stage, a task instance takes a set of queries, $\vec{x}$, and a benchmark, $B = \{x, y\}$, which consists of a set of queries and the answer, y, and returns a payoff, $P(T, \vec{x}, B)$. The existence of a benchmark is common in many tasks, e.g., the test set in supervised machine learning is used to evaluate a model's quality.

Each participant's goal is to answer all their queries and obtain the maximum *value*, v_i, from the answers. The value is given by a *value function* that is increasing with the payoff, $v_i = V_i(P(T, \vec{x}_i, B_i))$. v_i is private information to the participant. Participants may indicate their value to the market via a bid, b_i. When $b_i = v_i$, we say the bid is truthful. A participant can setup its own task and obtain value v_i, or they can participate in the consortium to obtain a higher payoff and hence higher value, v_i^P. For the purposes of this paper it may help to think of "money" as the unit of value.

Data's combinatorial power. In theory, a combination of datasets is more valuable than the sum of the parts; this is data's combinatorial power [3, 22]. This "excess" value is what participants gain from participating. In practice, combining datasets not always increases their value due to dirty data, strategic inputs, and more. The algorithms we design detect these situations to avoid harming the pool of data, ensuring $v_i^P - v_i \geq 0$.

Incentives and Uncertainty. Despite the upsides of sharing data, in many scenarios participants do not know how much value they will get from participating, i.e., $v_i^P - v_i$, and how to compare that value to the cost of sharing, which may include data preparation costs and potential risk of leakage. Without solving this uncertainty they default in not sharing and the consortium does not form. We propose a model and protocol to tell participants $v_i^P - v_i$ so they can decide whether to participate.

3 SHARING MODEL

Incentives. We design a sharing model that incentivizes participants to pool their data by offering two incentives:

- **Sharing Dominance** Participants obtain more value when they participate (i.e., share data) than when they do not, $v_i^P \geq v_i$. The higher value of participating stems from access to the consortium's data that leads to better payoff in general and to the same payoff in the worst case.
- **Entitlement Stake** Participants are compensated in proportion to their data's contribution to the consortium. The compensation is computed via a *revenue allocation* function *RA*.

The second incentive promotes *data sustainability*: Because better data is compensated higher, participants are incentivized to identify and share high-valuable data. This positive feedback loop benefits every participant.

Costs. Establishing and maintaining a data-sharing consortium requires covering the costs of operation, including:

- Cost of task setup, $C_T(\mathcal{T}(D_i))$. E.g., preparing a dataset and training a machine learning model. This cost is manageable when participants share a selected dataset and they are motivated to prepare it so it is easy to combine.
- Cost of storing data, $C_S(D_i)$. E.g., cloud blob storage costs.
- Cost of task querying, $C_Q(\vec{x})$.
- Cost of task evaluation, $C_E(\vec{x}, T, B)$. E.g., the costs of obtaining the task instance's payoff on the given benchmark.
- Cost of running the revenue allocation function, C_{RA}.
- Cost of combining datasets and setting up the task with the newly combined data, $C_T(\mathcal{T}(D^I))$.
- We do not model the cost of running the protocol (negligible).
- Cost of uploading data to an analysis platform applies both when sharing and not sharing. We do not represent these costs because they complicate expressions without offering any new insight.

The utility function for a participant who does not share is:

$$u_i(\vec{x}_i, D_i, B_i) = V(P((D_i), \vec{x}_i), B_i) - C_T(\mathcal{T}(D_i)) - C_S(D_i) - C_Q(\vec{x}) - C_E(\vec{x}, T, B) \quad (1)$$

and we can also show the utility function corresponding to a participant who shares:

$$u_i^P(\vec{x}_i, D_i, B_i) = V(P((D^I), \vec{x}_i), B_i) + RA(R(\vec{b}, \vec{x})) - C_S(D_i) - C_Q(\vec{x}) - C_E(\vec{x}, T, B) - C_P \quad (2)$$

where C_P represents costs specific to the sharing protocol used. For example, it includes the cost of running the revenue allocation function C_{RA}, of combining data, $C_T(\mathcal{T}(D^I))$, and others. The expressions do not include agent-specific costs, e.g., the cost of privacy loss or data leakage. In considering whether to participate, participants weigh costs and benefits: our protocol's goal is to communicate the benefits so agents can decide.

3.1 Requirements of Sharing Infrastructure

There are three requirements for the data-sharing infrastructure. First, the infrastructure *must guarantee participants that their data will be protected* and will not leak throughout the process. Second, participants need to decide *who is responsible for performing the consortium's tasks* and how to cover the associated costs described above. Third, because the asset they exchange is data, they *face Arrow information paradox* [8]. As a *consumer*, accessing new data boosts v_i but participant i does not know by how much. To learn that, they need to access the data. But as a *provider*, j does not want to reveal their data to other consumers before obtaining a compensation. This is because data is non-rival, so once revealed, it cannot be revoked. This results in a deadlock that prevents the potentially valuable consortium from forming. Different sharing infrastructure address these problems differently as we discuss next:

Federated Learning [48]. In federated learning, a central server maintains a global model and each participant downloads and updates the model with gradients derived from their local data. Because data never leaves the participants' infrastructure, this infrastructure's security guarantees are as high as the security of any of the involved participants' infrastructure.

Data escrow [72]. The data escrow is a system to control data flows. It supports delegated, auditable, and trustworthy computation. Participants share data with the escrow as if it was their own infrastructure; they have full control on who accesses data and for what purpose e.g., to run task setup, querying, and evaluation. The escrow derives its source of trust from keeping data encrypted end-to-end, even during execution. This is achieved by leveraging cryptographic protocols and secure hardware enclaves.

What architecture solves the three infrastructure challenges? In principle, all three challenges may be solved in either, and the protocols we introduce in this paper could be adapted to both. In practice, however, it is easier to reach agreement when there is a central trusted entity, and it is easier to work around Arrow Information Paradox using the escrow, who can signal to participants the value of the pool data, v_i^P, because it can compute it centrally while guaranteeing that no data is released. Furthermore, there are other important practical differences between the architectures:

- The escrow is a generic architecture that accepts any data-driven task unmodified, unlike federated learning that concentrates on machine learning tasks only and requires modification of the learning algorithms.
- The escrow keeps participant's data fully protected, while in federated learning, the sharing of gradients leaks information that sophisticated attackers may exploit [65].
- The escrow is more amenable to perform *data integration* [40] and *data discovery* [32] than federated learning because of its logically

centralized nature. And solving these tasks will be necessary in many sharing scenarios.

3.2 Intermediary Platform as a Data Escrow

In this paper, we leverage the data escrow to represent the intermediary *platform P* to tackle the three sharing infrastructure challenges explained in the previous section: i) protecting participants' data; ii) performing the consortium's tasks; iii) dealing with Arrow information paradox. The platform performs the consortium's tasks while implementing a protocol to circumvent Arrow's Information paradox. Concretely, the platform creates task instances given input data, and it pools datasets from different participants together, $D^I = D_i, D_{i+1}, ..., D_N$, as well as benchmarks, $B^I = B_i, B_{i+1}, ..., B_N$. Participants trust the platform because it is implemented on a data escrow and each participant can verify the protocols and algorithms run on the escrow are correct.

Public and Private Information. The participants and the task $\mathcal{T}$ are public information known by everyone. The task instance, T is only known by the platform to avoid participants reverse engineering the task to craft data [39]. A participant's queries, answers, and benchmark $\vec{x}_i, T(\vec{x}_i), B_i$ are never revealed to other participants. The combined benchmark, B^I is only known by the platform. The participant's value, v_i is private, and the bid b_i is only known by the participant and the platform, but not other participants.

3.3 Challenges

C1. Sharing Protocol for data escrow. Given the utility functions above, participants know there is a value gain $v_i^G = v_i^P - v_i$ (i.e., the sharing dominance incentive) in participating in the consortium and that $RA \geq 0$, the entitlement stake incentive. These two incentives improve their utility. However, participants do not know whether the improved utility outweighs the costs. This uncertainty becomes a barrier to participation. To address this challenge, in Section 4 we design a data-sharing protocol that is implemented on top of a data escrow and is geared towards removing this uncertainty.

C2. Data Sustainability. The protocol is designed to permit the allocation of the value of data back to the participants. A component of such allocation is the design of a *revenue allocation* algorithm. Revenue allocation is expensive. The more expensive it is, the less revenue is left to allocate back to participants. In Section 5 we present a new algorithm that implements the *entitlement stake* incentive to address this challenge.

C3. Dealing with Bad Inputs. Some participants may provide *bad* data, dirty or strategically crafted, and harm the consortium. No matter the reason, such data must be detected and removed to prevent harming others. We introduce an algorithm to deal with this problem in Section 6.

4 FORMING DATA-SHARING CONSORTIA

We make two key design decisions to design the DSC protocol:

KD1. Pushing storage, querying, and evaluation costs to participants. The cost of storage, querying, and evaluation is paid by participants and not by the platform to avoid tragedy-of-the-commons scenarios [59], where participants overuse resources because they do not pay for them. For example, this disincentivizes participants from uploading all available data with the hope some will be useful to some task and will return a profit. Instead, it demands participants to consider carefully what data is worth sharing because they pay for it. The cost of storage is the same whether they share or not; we assume participants upload the data to some infrastructure for analysis, e.g., the cloud.

KD2. Platform covers all other costs. This decision is key to implementing the *sharing dominance* incentive, as we see next. The utility of participating is higher or the same than not participating when: $u_i^P \geq u_i \implies v_i^G + RA + C_T \geq C_P$. Then, if the platform covers C_P, the participants' are strictly better participating because of the two incentives we create: i) the improvement of their payoff, v_i^G; and, ii) the data compensation, RA.

The first design decision (**KD1**) aligns incentives among participants and platform, avoiding pitfalls that stem from oversharing. The second design decision (**KD2**) is central to the goals of the sharing protocol, but it introduces a challenge on the platform, that must cover participation costs to remain sustainable. With this background the protocol's goals are:

- To signal participants the benefit of participating, in terms of payoff, so they learn v_i^G.
- To extract the value of data from the consortium to: i) cover operation costs and remain sustainable; and ii) redistribute the remaining value back to the participants.

The second point is critical. Without extracting sufficient revenue to cover costs, the platform is not sustainable and the consortium cannot exist, and the challenging part is that the revenue extracted by the platform depends on the value of data for the participants, which is not known a priori. The Myerson-Satterthwaite theorem [55] is an important negative result in economics that tells us that it is impossible to guarantee that the platform will not need to operate at a loss. That is, we cannot guarantee simultaneously that every participant's utility increases by participating *and* that the platform is sustainable and can cover the costs of operation. Instead, we engineer the protocol to achieve the following. **If the consortium is worth forming, the protocol makes it sustainable, if it is not sustainable, it is not worth forming.** We present the DSC protocol in 4 stages.

4.1 The 4 Sharing Protocol Stages

The protocol consists of 4 logical stages geared towards signaling participants the value of participating (Stage 1) and extracting the value of data. Because the value of data is private to each participant, the protocol must elicit that valuation from participants (Stage 2) while promising to redistribute it back to them based on their data contributions (Stage 3). Crucially, the revenue allocation takes place only after the platform has captured part of that value to cover costs. We use Fig. 1 to present the protocol stages.

4.1.1 Stages 1, 2, and 3. We present the core stages first.

Stage 1: Signalling. Participants share their data, D_i, and benchmark B_i with the platform. The platform performs the task setup and evaluation stages on each participants' data, obtaining P_i, the

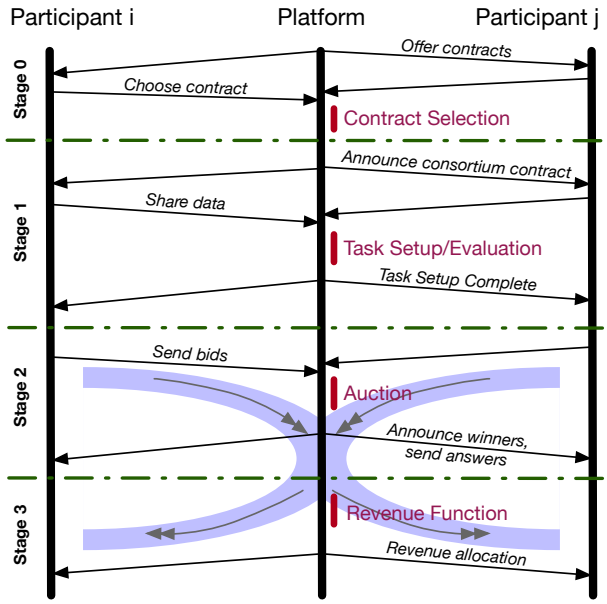

Figure 1: Data-Sharing Protocol

payoff. Then it combines the participants' data and performs the task setup and evaluation stage on D^I. Recall the platform is implemented on a data escrow and is thus trusted to perform this task (see Section 3.1). After this step, the platform gains access to P_i^P, and hence, $P_i^P - P_i$, which corresponds to the improvement on the payoff due to pooling data. When the platform *signals* this information to the participants, these apply their value function to learn the value of participating, v_i^P, and the benefit with respect to not participating, $v_i^G = v_i^P - v_i$. The platform incurs costs to perform this signalling step:

- $C_T(\mathcal{T}(D^I))$ and $N * C_T(\mathcal{T}(D_i))$. This is to setup the task using the consortium data, D^I, and each participants' data, D_i.
- $2N * C_E$, the cost of evaluating the quality of the consortium and individual model on each individual's benchmark, B_i

Pooled data is more valuable (i.e., yields higher payoff) than individual datasets. The goal of Stage 1 is to inform participants how much more valuable. Note that at this stage, the participants know collectively *the value of data*, $\sum_{i \in N} v_i^P$, even though no single participant or the platform knows it yet.

Stage 2: Value Extraction. The platform's goal is to extract the value of data from participants, $\sum_{i \in N} v_i^G$. To elicit that information, the platform runs an auction mechanism [54]. In an auction, participants bid to get access to an asset. Winners are chosen based on an allocation function, and they get access to the asset in exchange of a payment, which is determined by a payment function. The payment transfers value from the participants to the platform and is the mechanism by which the platform captures the value of data. State 2 must overcome three challenges. First, losers cannot be deprived of access to data, or otherwise participants will not participate because doing so may harm their utility. Second, to avoid long bidding times, the bidding process must be automated, so the process must be simple. Finally, the auction mechanism must be robust to strategic players seeking their own benefit at the expense of others. We explain this stage in detail in Section 4.2.

Stage 3: Value Allocation. After Stage 2, the platform has captured the value of data in the form of revenue. In Stage 3, the platform covers costs from the revenue raised. Then, it runs a revenue allocation function to distribute the remaining revenue back to the participants. The revenue allocation function itself is computationally expensive, so it is crucial to design algorithms that perform this step efficiently (Section 5) to avoid consuming more of the revenue that would otherwise go to the participants. In summary, Stage 2 extracts the value of data only to distribute it back to the participants, who are ultimately the beneficiaries of a value that stems from their own data and participation. Yet, Stage 2 is necessary so the platform can cover the costs of operating the consortium, without which all value would remain locked. After participants receive their compensation the protocol finishes.

4.1.2 Stage 0: Contract Agreement. The data-sharing consortium is governed by rules chosen by participants during Stage 0. The rules apply to: i) value extraction; ii) value allocation; iii) task-specific rules. Concerning value extraction, participants must choose among *homogeneous* pricing, where every winner pays the same query amount, and *discriminatory* pricing, where payments may differ among winners. We analyze both in detail in the next Section. Regarding value allocation, participants may choose between *Even Allocation*, that splits the raised revenue evenly, or *Entitlement Allocation* that allocates revenue based on data contributions. Finally, participants must agree on the implementation of the task setup stage. For example, in ML tasks, they may choose how much resources to invest in model search and tuning. All these decisions are made during Stage 0. We do not provide a mechanism to reach agreement as this is orthogonal to this paper.

4.1.3 Data-Sharing Costs. We flesh out the utility functions given the details of the protocol. The utility function for participating is:

$$\begin{aligned} u_i^P(\vec{x}_i, D_i, B_i) = V(P(\mathcal{T}(D^I), \vec{x}_i, B_i)) + RA(R(\vec{b}, \vec{x}), \vec{D}_i) \\ -C_T(\mathcal{T}(D_i)) - C_S(D_i) - 2 * C_Q(\vec{x}) - 2 * C_E(\vec{x}, T, B) \\ -1/N * C_{RA} - 1/N * \mathcal{T}(D^I) - 1/N * C_S(D^I) \end{aligned} \quad (3)$$

some costs are split among the N participants because they correspond to functionality the platform performs on their behalf; these costs include a $1/N$ factor. The corresponding utility function for not participating is:

$$\begin{aligned} u_i(\vec{x}_i, D_i, B_i) = V(P(\mathcal{T}(D_i), \vec{x}_i, B_i)) - C_T(\mathcal{T}(D_i)) \\ -C_S(D_i) - C_Q(\vec{x}) - C_E(\vec{x}, T, B) \end{aligned} \quad (4)$$

Hence, $u_i^P \geq u_i$ when:

$$\begin{aligned} v_i^G + RA \geq C_E(\vec{x}, T, B) + C_Q(\vec{x}) + 1/N * \mathcal{T}(D^I) + \\ 1/N * C_S(D^I) + 1/N * C_{RA} \end{aligned} \quad (5)$$

To prevent participants from facing uncertainty, the platform covers all costs. The consortium is sustainable when $\sum_{i \in N} v_i^G \geq C_E(\vec{x}, T, B) + C_Q(\vec{x}) + 1/N * \mathcal{T}(D^I) + 1/N * C_S(D^I) + 1/N * C_{RA}$. The revenue to allocate in Stage 3 is given by the difference between

the value raised and the costs. This puts pressure on Stage 2, that must raise enough revenue to cover these costs and to reallocate the remaining value back to participants during Stage 3, hence incentivizing (with *RA*) participants to keep sharing data.

4.2 Stage 2: Value Extraction with Auctions

We explain how Stage 2 deals with losing participants, and how it elicits private valuations.

4.2.1 Challenge 1: Dealing with losing participants. Because the asset is data, a non-rival and free to replicate asset [44], the platform could make every participant a winner and serve them the data. However, if participants knew they will be serviced no matter what, they would have no incentive to bid their true valuation because any bid wins. For the auction mechanism to extract value from participants, it needs to add competition. With competition there are winners and losers. Every winner obtains access to the asset and losers do not. This is unfortunately against the promise of the protocol, which is that no participant will receive lower utility when they participate than when they do not.

The idea to address this challenge is as follows. The protocol serves everyone, but with different quality of service: i.e., the protocol creates differentiation in the data offered instead of creating artificial scarcity. Concretely, the protocol makes participants compete to access the consortium data, D^I. Participants want to access D^I because of the promised gain in utility, v_i^G, which they learn after the signalling of Stage 1 of the protocol. After competing, winners gain access to D^I and losers do not. Hence, only winners obtain the gain in utility v_i^G. But losers still gain access to D_i, which, crucially, is equivalent to what they would have obtained had they not participated, i.e., they do not lose utility by participating. Crucially, this does not introduce additional costs on the platform side because the platform must already set up a task with D_i during Stage 1 to signal the participants what is their value in the first place. Then, setting a task on D_i achieves two goals: i) it lets participants learn v_i^G, which they need to decide whether to participate; ii) it lets (even losing) participants answer their queries, $\vec{x}$.

4.2.2 Challenge 2: Eliciting private value. In an ideal scenario, the participants pay the platform the marginal gain of participating so the platform can reallocate that value back to the participants according to the revenue allocation chosen for Stage 3, e.g., based on the value of their data contributions. The total value of data is $\sum_{i \in N} v_i^G$, so Stage 2 would capture this value and then reallocate it during Stage 3. In practice, *strategic* participants may misreport their value, submitting a bid that is lower than their truthful valuation, $b_i < v_i^P$, aiming to win the auction, and hence access to D^I, for a lower price, resulting in the platform capturing much less revenue than the total value of data. We want auction mechanisms that capture as much of the value of data as possible while being subject to the constraint of eliciting *homogeneous* or *discriminatory* prices, as agreed by the consortium during Stage 0.

We borrow incentive-compatible mechanisms from the mechanism design literature to elicit truthful reporting from participants. Because it elicits the true valuation, the result is that the platform *learns* the value of the data-sharing consortium, even though it may not be able to *extract* it all, as we will see soon. Still, because the goal of this extraction process is to cover costs and then reallocate the remaining revenue back to the participants, the mechanisms we are interested in are those that maximize revenue.

Setup. Each participant wants to use the consortium's data to answer their queries, thus obtaining as payoff P_i^P. To access the consortium's data, participants compete with each other on an auction which works as follows. First, participants send a bid b_i, which indicates how much they are willing to pay for each query x and they share the number of queries they have, $|\vec{x}|$. Then, the platform considers the total price the player would pay, $|\vec{x}| * b_i$ and discounts the costs of allocating compute, $C_Q(\vec{x})$ necessary to run the participants' queries. At this point, the platform sorts the participants according to the profit they generate, where participant in position $k = 1$ generates the highest profit and $k = N$ the least and runs the *homogeneous* or *discriminatory* price auction to determine the winners. No matter which auction runs, we ensure *allocation fairness*, which means that a winner in one auction would be a winner in the other one as well, and conversely, a loser would be so in both. This simplifies choosing a mechanism in Stage 0.

Homogeneous price auction mechanism. This mechanism guarantees that all winners pay the same price for each query, irrespective of what they bid. After the platform sorts the bids based on the profit they generate (as explained in Setup), it chooses the price, p_k according to: $p_k \mid k = \arg_{k \in 1...N} \max k * b_k$, where k is the position of the sorted bid in the list of received bids. Every participant i with $b_i > p_k$ wins and pays p_k per query. The rest lose. Because participants do not pay their bid, they are incentivized to bid truthfully, as that maximizes their chances of winning [35].

Discriminatory price auction mechanism. We implement a sequential second price auction [66]. The mechanism sorts bids and chooses winners. A winner does not pay what they bid, but the next highest bid, i.e., winner k pays b_{k-1}, winner $k-1$ pays b_{k-2} and so on. If we let everyone win, then the auction is not truthful because bids do not need to reflect the true valuation. Hence, we choose a price below which nobody wins, a *reserve price*. The *reserve price*, p_k, is chosen using the same procedure as in the homogeneous price auction. Notice that the only reason why we can run this kind of auction is because data can be sold infinite times[1] and the platform can choose at runtime how much computation to allocate to serve winners. Finally, choosing p_k as the reserve price ensures *allocation fairness*: winners are winners in either mechanism. Then, anyone who bids above the reserve price wins and the payment is done as explained above.

Analysis. Note we limit participants' actions to submit a single bid that indicates what they are willing to pay for each query. Allowing bidding different quantities for different queries may seem like a natural relaxation, but unfortunately it opens up opportunities for price manipulation in the form of *demand reduction* [9]. Similarly, it is tempting to think of alternative auction mechanisms, such as *all-pay* auctions with a reserve price, that intuitively seem to capture higher revenue than the discriminatory price one. All-pay auctions are harder to analyze, harder to play for participants, and are not guaranteed in general to lead to higher revenue. Furthermore, we wish to let participants program agents to bid on their behalf, so as

[1] or rather, in this case, access to data via queries

to avoid human in the loop during Stage 2 and reduce costs. Then, a harder-to-play auction translates into arbitrariness in the way agents automate bidding. Incentive-compatibility makes playing simpler because the best strategy is clear. This makes programming bidding agents easier.

Although ideally Stage 2 would capture the full value of data, the existence of strategic behavior means that some value (revenue) cannot be extracted. Neither the homogeneous or discriminatory mechanisms fully capture the value of data, and the discriminatory mechanism captures more of it than the homogeneous one.

4.3 What does the Protocol Achieve?

The protocol ensures every participant is better off sharing data than not, i.e., the sharing dominance incentive. It achieves so by capturing the value of data first from the participants, then covering the costs of signalling and circumventing Arrow information paradox, and finally distributing the remaining revenue back to the participants, thus implementing the entitlement stake incentive.

The protocol is designed to reduce the many problems of forming consortia to eliciting value and allocating it back to participants. It leverages auction theory to extract revenue from participants. The most important feature of the protocol is that it makes sustainable only those consortia that are worth forming: the revenue truthfully reflects the value of data so only when it is higher than the costs is the consortium worth. When revenue is not sufficiently high to cover costs the socially efficient outcome is for the consortium to not be formed in the first place.

5 REVENUE ALLOCATION

In this section, we review the requirements for revenue allocation (Stage 3 of the protocol). Then we present a popular method to implement this functionality (Section 5.1), our argument against its use (inefficiency) (Section 5.2), and an alternative efficient algorithm for forming data-sharing consortia in Section 5.3, concluding with an analytical discussion of the solution in Section 5.4.

Requirements of Revenue Allocation. During Stage 3 of the protocol, the value extracted during Stage 2 is distributed back to the participants after covering the platform's operation costs. At this stage, the platform knows the magnitude of all costs except for C_{RA}. The first requirement for the revenue allocation function is that its cost, C_{RA}, should be lower than the remaining revenue, so that the platform covers operation costs and so there is some revenue left to compensate participants and implement the Entitlement Stake incentive. We consider two variants:

- **Even Allocation.** This revenue allocation function splits the revenue evenly across participants. It is simple and minimizes the cost, therefore maximizing the revenue to allocate to participants. However, it does not incentivize data sustainability because every participant is compensated equally regardless of the quality of their data contribution.
- **Entitlement-based Allocation.** This revenue allocation function distributes revenue according to an entitlement specific to each participant. When the entitlements are computed according to the value the participant's data contributed to the consortium's data, then it implements the Entitlement Stake incentive.

The revenue allocation function is agreed upon by participants during Stage 0. We do not discuss the *even allocation* further as it is simple and less interesting than the *entitlement-based allocation*, to which we devote the rest of this section.

5.1 Entitlements and the Shapley Value

In the context of revenue allocation, the entitlement of a participant indicates the percentage of revenue it receives. The entitlement must reflect the participant's data contribution to the consortium. Once entitlements are computed, their allocation is straightforward because revenue (money) is a divisible good [16]. Hence, the main challenge is in computing entitlements for each participant.

Recent literature has proposed using the Shapley value [4, 42, 43, 71] to calculate the entitlements of each participant based on the value of their data for a task, and it has primarily focused on machine learning tasks. The motivation for using the Shapley value is that it achieves a number of desirable properties such as i) *group rationality*, which means that the summation of entitlements is 100%, i.e., all value is distributed across participants; ii) *fairness* which means that data that contributes the same to the task should have the same value; iii) *linearity* which indicates that the value of the combination of two datasets is the sum of each dataset's value; and iv) *null player* that says that a dataset with no value should receive a value of 0. Furthermore, the Shapley value is proven to be the only way of achieving these properties. The expression for the Shapley value is as follows:

$$s_i = \sum_{S \in I \setminus \{i\}} \frac{1}{N\left(\frac{N-1}{|S|}\right)} [P(S \cup i) - P(S)] \tag{6}$$

where I indicates the set of N participants and S is a subset (i.e., *coalition*) of those. The payoff P is computed over the data from the coalition S, i.e., when S is one participant, this corresponds to that individual's D_i. The Shapley value computes the marginal contribution of a dataset to every possible *coalition* of the remaining datasets and then averages these contributions to produce the final value. Based on the Shapley value the entitlement for a participant i is computed as $s_i / \sum_{i \in I} s_i$.

Despite the apparent benefits of the Shapley value we argue it is not a good solution for computing entitlement in a data-sharing consortium. Next, we explain why and propose an alternative.

5.2 The Case Against Shapley-based RA

In choosing the Shapley value to allocate revenue, the literature pays attention to the 4 aforementioned desirable properties but it ignores the cost (in compute time and resources) of computing Shapley and its implications to revenue allocation. The source of Shapley's complexity is that it enumerates *all* possible coalitions the participants *could* form. But in a real setting only *one* coalition forms. If we wanted to enumerate all those *virtual* coalitions the platform would need to pay for them.

In principle there is nothing wrong in trying hard to unearth all data's interactions (by enumerating all virtual coalitions), except that this costs money, and that translates into lower revenue allocated to participants. This is at odds with the sole reason to run a revenue allocation function, which is to compensate participants.

As a result, there is a tension between unique entitlements, as computed by the Shapley value, and entitlements that do not follow those 4 properties but that leave more revenue to participants; we discuss this tradeoff in detail in Section 5.4. All in all, it amounts to asking participants to choose between Shapley entitlements or a higher compensation.

In summary, using Shapley as the revenue allocation function increases costs leading to two undesirable implications. For consortia that forms, a larger proportion of the value of data is consumed in redistributing it back to participants, leaving less for compensation. Even worse, because only consortia with $\sum_{i \in N} v_i^G \geq C_{RA}$ forms, many fewer would form when C_{RA} grows. To conclude, many efforts offer approximations for the Shapley value that work in both specific [42] and general [34] cases. Despite the more efficient runtime, these methods still incur a much higher cost than our proposal, as we demonstrate in the evaluation section.

5.3 The Entitlement Stake Algorithm

The intuition of the algorithm follows closely that of leave-one-out [31], so we emphasize the differences. The algorithm first performs N task setup stages, each using as input data D^I_{-i}, that indicates all the consortium's data but i's dataset. It then computes the payoff of each of the N task instances and the task instance built on the consortium data, D^I on each of the evaluation queries in B^I. This permits the algorithm to identify what is the payoff if a participant's data was not used and thus obtain the difference.

Robustness. Here, we assume that every participant's data contributes to the overall payoff, i.e, that bad inputs have been removed using the algorithm we present in Section 6.

Entitlement calculation. The algorithm creates a matrix where rows correspond to the evaluation queries and columns to the task instances. There are $N + 1$ task instances (all D_{-i} and the consortium's). Elements in the matrix indicate whether a task instance succeeded or failed on a particular evaluation query (e.g., 1 and 0 to indicate hit and miss). Then, the algorithm distributes entitlements based on what queries are answered correctly. First, it filters all but queries for which the consortium's model got the right answer. Then, it assumes each remaining evaluation query has a unit of value. The unit is split evenly among participants whose data contributes to obtaining the right answer; a participant's contributes to the answer if removing their data leads to a wrong answer, $P(D_{-i}, x_i, B^I) = 0$. Thus, the algorithm accounts for the difficulty of answering a query; the more participants' data contribute to get the answer right, the less value each participant receives. Finally, entitlements correspond to the percentage of value captured by each participant.

5.4 Analysis

In justifying the new algorithm, we explained that due to the costly computation of Shapley (SHA), DSC would leave more revenue available to participants. Here, we want to argue analytically when this additional revenue means every participant gets a higher compensation in DSC than in SHA. Later, in the evaluation section, we will show the differences empirically.

Building Intuition. Participants compensation depends on their entitlement, and their entitlement depends on both their contribution and others' contribution. If revenue in SHA and DSC was the same, then we cannot guarantee DSC would leave more revenue to participants because the contribution computed by DSC may differ from that of SHA. We can use the cost of computation as a proxy for the revenue available, e.g., if SHA is X times more computationally expensive than DSC, it consumes that much more revenue (wlog: assuming linearity of computational resources costs). We express cost of computation as the algorithm complexity. DSC grows with $n + 1$ (n task setups for participants plus an additional one for the consortium) and SHA grows as $2n(2^{n-1})$. Therefore, DSC will be cheaper than SHA for any $n > 2$ and will be the same otherwise. Then, if entitlements were the same, DSC would certainly leave more revenue for participants. However, DSC may compute different contributions and hence entitlements than SHA. Hence, we want to identify a bound that indicates how much lower an entitlement can be in DSC than in SHA, and then use the bound to determine when is DSC cheaper than SHA.

Bound. First, notice that if we had only 2 participants, then DSC and SHA would do the same amount of computation and would lead to the exact same result. With $n > 2$, DSC will only consider 1 coalition per participant, while SHA will consider all coalitions—which explains its higher cost. The contributions in SHA, each arising from checking the contribution of the participant's data to each coalition, form a distribution. We empirically test that this distribution is normal by collecting such contributions and testing for normality with a D'Agostino and Pearson's test [24]. We cannot reject the null hypothesis ($p >> 0.3$) so the contributions' distribution is normal. Note that DSC is a sample from that distribution, so we can calculate a confidence interval. We cannot use the traditional method because our sample is of size 1 and the standard deviation is undefined. We leverage the results from Wall, Boen, Tweedie [70], that build upon earlier results by Machol and Rosenblatt [53] to derive confidence intervals from samples of size 1. A 95% confidence interval is computed for a sample x as $9.68 * |x| \pm x$.

When does DSC yield higher revenue than SHA?. The previous result tells us that the contribution of a participant can be off by $\sim 10x$. When contributions are represented as entitlements, in the worst case, one participant in the consortium obtains a contribution that is 10x smaller than SHA and all other participants get a contribution that is 10x larger than SHA. This would result in a 100x lower entitlement. Then, the cost of SHA would need to be $> 100x$ higher than DSC for such participant to still obtain a higher revenue in DSC than in SHA. When the ratio $\frac{2n(2^{n-1})}{n+1}$ is larger than 100, the claim is true. This is the case when $n \geq 7$. This is a theoretical worst case. In practice, we find that entitlements calculated by DSC are accurate and that even when $n = 4$ it is much cheaper than SHA and than state-of-the-art approximations of SHA. We show this in detail in the evaluation section.

Shapley axioms and the Entitlement Stake algorithm. The Shapley value is the unique way of obtaining the 4 properties of *group rationality*, *fairness*, *linearity*, and *null player* outlined at the beginning of the section. These properties are desirable when studying the leverage different coalitions of participants have on a game's

outcome: this is the origin of the Shapley value. We now contrast and discuss tradeoffs compared to what our algorithm achieves. After reducing the cost of revenue allocation, both Shapley and the new algorithm achieve *group rationality* and *null player*. In contrast to these two properties, the degree to which the new algorithm achieves *fairness* and *linearity* depends on the size of the consortia; for small consortia, the new algorithm achieves similar results to Shapley (see evaluation section). Finally, the *fairness* definition of the Shapley value may not coincide with what a consortia understands as a *just* allocation: Shapley allocations lead to lower compensation for each participant due to its algorithmic cost.

6 ROBUSTNESS AGAINST BAD INPUTS

In this section, we discuss how to make the data-sharing protocol robust to *bad inputs* that may break the promise that $v_i^P \geq v_i$.

6.1 Breaking $P(D^I) - P(D_i) > 0$

An incentive to participate is that the consortium's data, D^I will yield higher value than accessing only their own data, D_i. In theory, more data helps (see the argument in Data's Combinatorial Power of Section 2) because information cannot hurt [22]. Unfortunately, in practice, adding an additional D_i to a combination of datasets may harm the payoff. The reasons include bad quality data, bad quality benchmark, e.g., with missing entries, bad labels, strategic actors who craft datasets seeking to increase their utility at the expense of others, and, in some cases randomness associated with the task, e.g., ML algorithms that get stuck on a local optima.

When a participant's D_i harms payoff, it reduces the value of the consortium, $P(D^I) - P(D_i)$ and thus the chances of others participating. The solution is to design a method to detect bad D_i and ban those participants until they improve their own data. Consider the following scenarios:

- (Strategic) *i creates B_i so it performs well with their D_i with the expectation they will capture more revenue.* If the consortium model does worse, this will be noticed.
- (Strategic) *i finds data D_i that does well with B_i*. The participant aims to perform better on B_i by identifying a better input data, D_i. This is welcomed because it may help the consortium as well.
- (Quality) *i uploads D_i and B_i*. Both D_i and B_i could be of *bad* quality: the data could be old, irrelevant to the problem, could be riddled with errors, etc.
- (Random) *i uploads D_i* and B_i that affects the consortium's data quality due to the randomness of the task, e.g., a ML algorithm getting stuck on a bad local optima when using D_i.

Some strategic behavior is harmful (first point) and some is welcomed (second point). And sometimes there is no strategic behavior but bad quality data, or task randomness harms the consortium's data performance. Our goal is to detect any D_i that reduces the consortium's data quality, irrespective of the underlying reason. When D_i does not harm the consortium, then it is welcomed, even if D_i has been created strategically. In this case, the participant's incentives are aligned with the data-sharing consortium's: we want participants to share D_i that increases consortium's value.

The robustness algorithm. We implement an algorithm to detect bad inputs. The algorithm executes at the end of Stage 1. The algorithm runs simultaneously with the revenue allocation algorithm, thus saving computational resources. It searches for inputs that, when not present, lead to payoff improvements.

It proceeds iteratively. First, it removes all inputs provided by a participant i, which include D_i and B_i and evaluates each D^I_{-i} on the consortium's benchmark without the participant's B_i, i.e., B^I_{-i}. The algorithm performs this step for each of the participants and compares the payoff with and without their data. If removing a participant's inputs increases the consortium's payoffs, this participant is banned from participating. The process repeats until no new bad inputs are detected. It follows that consortia where most participants provide bad data are costlier to run but this is aligned with the promise of the protocol: that only worth forming consortia form. After all, if most participants bring bad data, the consortia will not be worth forming.

6.2 Strategic Bids Across Stage 2 and 3

In addition to data, participants may strategize with the *bids* they submit across stages 2 and 3, which cycle. To prevent this problem, the platform establishes a barrier at the end of Stage 3, thus preventing strategic participants from creating bid sequences to obtain a higher utility at the expense of the consortium. While a round of this stage is robust to strategic bids because the auctions we implement are incentive-compatible, composing mechanisms, including auctions, is difficult [66]: many properties that apply to a mechanism break when it is composed with others. In particular, if the protocol memorized past bids, and use those to elicit a price in subsequent auctions, strategic participants could exploit this knowledge and submit a sequence of bids that would modify the price in a profitable direction [2, 18].

In scenarios where bids arrive online and must be answered immediately—such as in ad auction mechanisms—this is an issue. In our setting, we eliminate this behavior by creating a barrier between cycles of the loop and not carrying any information across iterations. In other words, each auction is completely independent from the previous one and hence, participants cannot strategize over time. To work, participants willing to bid must ensure their queries arrive in the platform within a specified window. This seemingly drawback that requires participants attending to when is it necessary to send queries is no different than how consortia of participating organizations must identify ways of coordinating today. The advantage is that it simplifies the protocol, reduces gaming opportunities, and the central platform can act as the coordinator.

7 EVALUATION

We structure the section around three key research questions that we answer empirically:

- **RQ1: When is consortia viable?** The protocol ensures a consortium is formed only if it generates value from data for participants. Here, we compare costs of different baselines to study what markets are viable.
- **RQ2: Market Robustness. Does D^I perform better than D_i in practice?** The protocol relies on the consortium's pooled data yielding higher quality than any of the individual participants'

alone. In practice, bad inputs may harm the task's payoff, causing some participants to not become part of the consortium. Here we explore whether the algorithms we present detect and remove bad inputs, thus maintaining the robustness of the protocol.

- **RQ3: What is the practical difference between Shapley-based revenue allocation methods and the algorithm presented here?** From a complexity perspective, the answer is clear. Here, we explore empirically whether such difference matters. We explore and compare these methods in runtime (cost), and in the difference of the entitlements they compute.

Setup. Although the data-sharing protocol and algorithms support many tasks, we focus here on machine learning tasks, that command a lot of attention in current sharing efforts. We implement the algorithms in Python and test them on a MacBook Pro with 16GB of memory and a Chameleon [46] node with 256 TB of memory to verify the runtime we measure is due to the algorithm complexity and not to lack of computational resources.

Non-Deterministic Payoffs. Machine learning tasks introduce a new challenge in practice because the payoff function is in general non-deterministic, i.e., two models trained on the same data will yield different payoff, we must use a stability mechanism as part of the task setup stage. The one we employ consists of training 10 models and testing each on 10 boostrap samples of the test set [38], then averaging the resulting 100 scores and delivering a more stable estimate of the test error. Non-deterministic payoffs play an important role in data-sharing consortia because they affect revenue allocation, among others. We will study the effects of achieving this stability throughout this section.

Datasets. None of the experiments we conduct depend on the specific dataset or task. This means one dataset suffices to answer the research questions; still, we use a second dataset on each experiment to demonstrate the trends remain. We use the Census Income Dataset from UCI [29] which consists of about 33K training samples and 15 attributes and a prediction task (predicting income $\geq$ 50K), and the hotel booking dataset from [6], which consists of 120K samples and 32 attributes and a prediction task (predicting hotel cancellations). For the UCI dataset, we study the data distribution effect by using two variants:

- **Biased split.** This resembles real scenarios, where certain participants may have data that follows different distributions, e.g., data about only a subset of countries, segment of the population, etc. We use the What-If Tool [36] on a trained model to understand what attributes impact the prediction task the most and then split the dataset so as to ensure some participants have high-impacting attributes and others do not.
- **Even split.** The dataset is split evenly across the participants and as a result each participant contributes the same amount of data with the same distribution. We use this variant for experiments where the performance of the model does not matter and with the goal of reducing other effects.

During the *task setup* stage, we train and tune the models, following standard practice: i) preparing the data by applying normalizations and standardizations of values when useful; ii) performing hyperparameter search using grid search; iii) evaluating results using cross validation with 10 folds. To evaluate performance, we hold a test set with 10% of the training data. We use a random forest for the UCI dataset, and the CatBoost model [28] for the hotel dataset. In the hotel dataset, we split the dataset according to the top 5 countries with the most hotel reservations.

Baselines. We call DSC to the implementation of the new protocol. We then implement alternative revenue allocation strategies. We implement the Shapley value, SHA. We implement the truncated monte-carlo approximation of the Shapley value [34], TMC. TMC takes a number of iterations as a parameter. Instead of configuring this parameter manually, we follow the approach described in the original paper and execute the algorithm until the average contribution has empirically converged [34].

7.1 RQ1: When is consortia viable?

A consortium is viable when it raises sufficient revenue to cover costs and compensate participants. Then, the more expensive to run a consortium, the less likely it will form because each participant would need to pay more to access the consortium's data. In this first experiment, we compare the baselines for the binary classification ML task to understand their effect on consortium formation. In addition, we implement a variation of DSC that we call NOS and that does not attempt to obtain stable predictions. NOS helps us understand the cost of stabilizing the task's payoff.

In this experiment we consider the **Even split** dataset and consortia sizes of 4, 8, and 12 participants. Because splits are equally sized, larger consortia pool larger data together and task setup times (involving training, and tuning the model) will accordingly increase. For each size we only measure runtime which involves setting up the task and computing revenue allocation, which itself includes running the robustness algorithm.

Fig. 2a (left) shows that DSC is 5x faster (thus cheaper) than TMC, the Truncated Monte-Carlo simulation [34], which is the state of the art approximation to the Shapley value. TMC leads to more expensive consortia, thus fewer form, as only those consortia that raise revenue sufficient to cover the costs form. At the same time, TMC computes less accurate entitlements than DSC, as we will show in Section 7.3. The original Shapley value, SHA, takes 6 hours for a consortia of size 8; we do not run it for consortia of size 12 as the runtime estimation is of 10 days. Compared to the 6 hours of SHA, TMC takes 3.5 hours, and our protocol DSC only 43 minutes. Furthermore, TMC grows faster than DSC. Finally, the no stable variation of DSC, NOS, is cheaper than DSC because it trains a fixed constant fewer models than DSC. The cost of stability is well spent, though, as we will investigate in more depth in Section 7.3.

Interpreting the results. Because lower runtime translates into lower operation cost, more consortia will form with DSC than the alternatives because the amount of revenue needed to cover costs is lower. Notice that the model we use in the experiments is a relatively simple Random Forest where a single model training and tuning process takes less than a minute. If the model was more complex and expensive to train, such as a neural network, the cost of operation would increase rapidly, and the gap between DSC and TMC would become even more practically significant. For example, in AWS [10], a representative cloud vendor, a 3.5 $/hour instance suffices to run the Random Forest models, but high-performance instances with GPUs can cost up to 32$/hour. In conclusion, when using DSC more

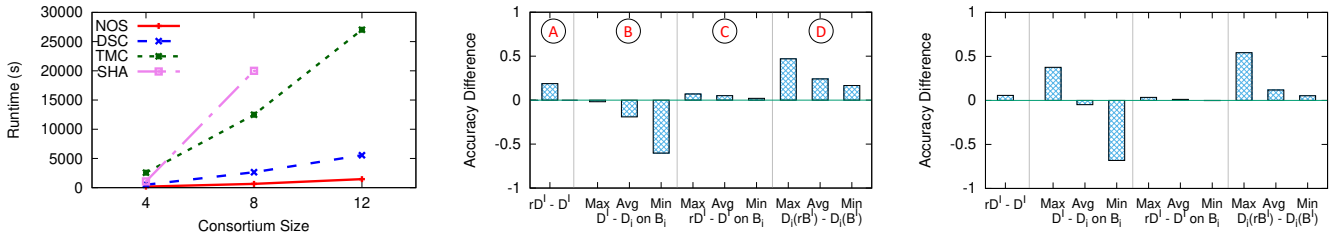

Figure 2: Consortium cost (runtime) on the left. Effects of bad inputs with 8 participants (middle) and with 12 on the right.

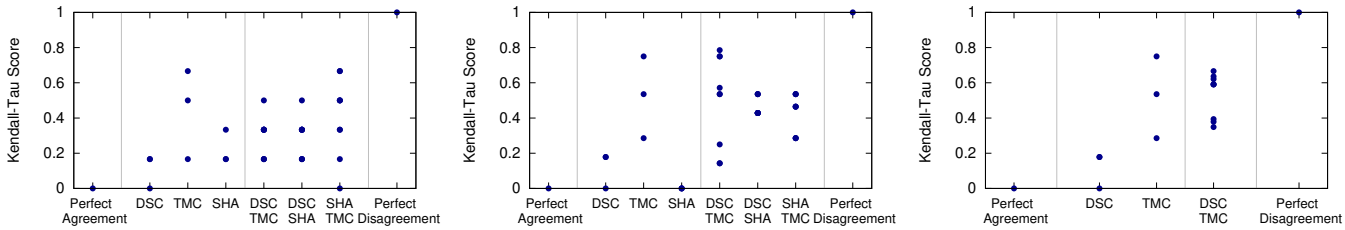

Figure 3: Agreement between rankings for consortia of 4, 8, and 12 participants

consortia will form, and participants of that consortia will receive a better data incentive because more revenue will be available.

7.2 RQ2: Is the protocol robust to bad inputs?

One of the incentives the protocol uses to form consortia is that participants get a chance to access D^I and that will yield higher payoff than using their own data, D_i. However, in practice there are cases where this may not be true due to bad inputs (see Section 6.1). In this experiment, we measure the effectiveness of the robustness algorithm on the even and biased versions of the dataset:

Even Split case. We use a consortia size of 8 and we perturb the labels of one of the consortium's D_i to simulate a low quality dataset. Then, we run the protocol with and without the robustness control and take different measures that help us understand what are the outcomes on each case. We use the letter r to refer to data that is subject to the robustness control, e.g., rD^I, rB^I and so on. The results are shown in Fig. 2b, which shows 4 vertical bars to separate the different measures.

Starting from the left hand size of the plot (A), the measure $rD^I - D^I$ indicates the difference in accuracy of the consortium model when running with robustness control and without. There is a big difference of 19 points, explained by the harmful effect the bad data has on the consortium's model. Although the difference in accuracy is important, there is an even more harmful effect, which is shown in the second segment of the plot (B), with the measure $D^I - D_I$ on B_i. This measure corresponds to the value gain, v_i^G, that each participant perceives when sharing, and we measure it without running the robustness control. Because there are many participants, we show the maximum, average, and minimum measure. The results show that for the average participant, participating in the consortium reduces their utility (by up to 19 points): without using robustness, the protocol cannot ensure that participating is strictly positive and hence, the consortium will not be formed.

In contrast, when running the robustness control, every participant sees an average benefit of 5 points when participating (C), which is a noticeable difference for many machine learning applications. Finally, the right-hand segment of the plot (D) shows the difference in performance of individual models on the consortium benchmark, D^I, when using robustness control and not, which we show as $D_i(rB^I) - D_i(B^I)$, slightly abusing notation. These values are used by the revenue allocation function to compute entitlements. The results confirm that the robustness method leads to large improvements and hence correct entitlements. Or conversely, bad inputs that perturb the consortium benchmark affect the data compensation that participants receive. We reproduce the results for consortia of size 12. In this case it is harder to identify bad inputs because only 1/12 of the data is perturbed, so its effect on the overall consortia is smaller. Still, the algorithm detects the adversary.

Biased Split case. When using the biased split, the robustness protocol remains effective. Fig. 2c shows the results. First, robustness increases the accuracy of the model. Second, in contrast to the *Even Split* case, the maximum difference of accuracy without robustness is positive in the *Biased* case: this is because the baseline quality of one of the players is low. However, the average participant would not participate without robustness, and some participants' utility deteriorates even more than in the *Even Split*, again, due to the data's original bias. Finally, when robustness is used, every participant sees an improvement when participating.

Different dataset. The results for the Hotel dataset are comparable to the previous ones (see Figure 5a): without robustness, some participants that would benefit from participating would not do it. In this case, we use a different ML model, and different amount and type of data (see specs at the beginning of the section). This confirms the applicability of the robustness protocol.

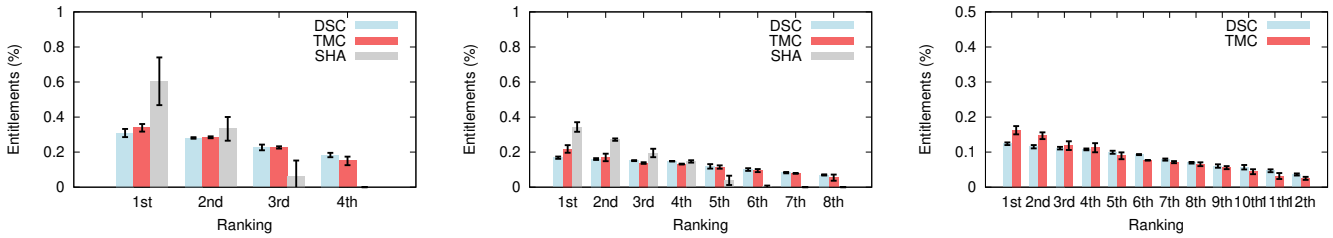

Figure 4: Absolute difference in entitlements as computed by DSC, SHA, and TMC, and in consortia of sizes 4, 8, and 12

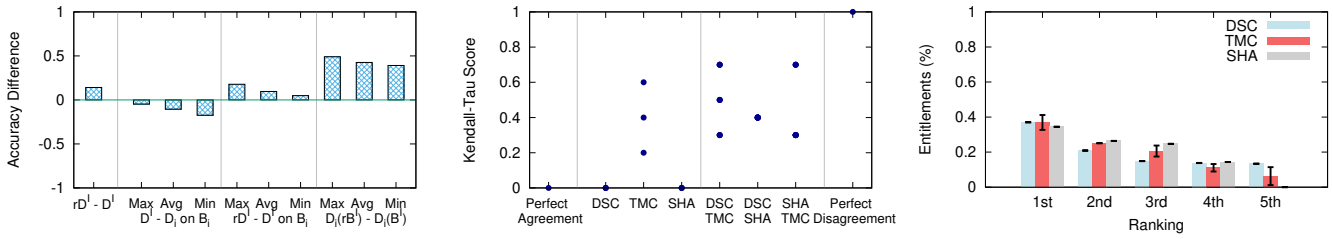

Figure 5: Effects on bad inputs, ranking agreement (center), and absolute entitlement difference (right) on the hotel dataset.

7.3 RQ3: Practical differences between Shapley-based and DSC RA functions

Here we compare the outcomes of revenue allocation. We perform two experiments. First, we focus on the stability of the entitlement: does the revenue allocation function lead to the same results when run on the same inputs even if the task is stochastic? We expect the answer to be yes, otherwise participants will be wary to participate in a lottery. Second, we measure the absolute differences between entitlements when using each revenue allocation function.

7.3.1 Stability of Outcomes. In this experiment, we compare the outcomes of DSC, TMC, and SHA. Here, the outcome is the ranking of participants each method produces. Ideally, when we run the same method on the same input, we expect the same outcome as well. If we run such a method N times, ideally all rankings would be the same. However, in practice, tasks with a stochastic component, such as machine learning tasks, may produce different results even when trained on the same data. If these results affect the ranking order, then the data compensation is not reflecting the value of data alone, but also the noise the task setup introduces. To measure the agreement between rankings, we use the Kendall-Tau Score.

We show the results for consortium sizes of 4, 8, and 12 in Fig. 3. We show scores for perfect agreement (0) and disagreement (1) for reference. We run each revenue allocation method 3 times and record the output rankings. We then enumerate all pairs of rankings, measure the Kendall-Tau score for each pair, and plot the score. The results show how the agreement between DSC is almost perfect across consortia sizes, which is what we want. In contrast, the agreement among TMC results is a lot less stable, as the higher scores demonstrate. This trend is true for all consortia sizes. When measuring the agreement of SHA, we observe relatively large disagreement with consortium of size 4 and perfect agreement with consortium of size 8 (we did not run SHA for 12 because of the huge running time it would take). SHA enjoys some stability by default because it is the result of averaging over all marginal contributions of every participant's data to every single coalition.

We then measure the agreement between the rankings produced by different methods. Here, when comparing two methods, we measure the score between every pair of rankings across methods. If we consider SHA a good reference of the value of data, then the disagreement of a method with respect to SHA would indicate the quality of the method. The second segment of the plot shows the results for this experiment. It demonstrates that DSC agrees with SHA more than TMC does. This difference is a consequence of the natural instability of TMC. Ranking stability is crucially important if a protocol is to attract participants based on their data contribution. DSC is more stable and more efficient than TMC.

7.3.2 Absolute Entitlement Differences. We measure the absolute difference in entitlements across baselines. We show the average and standard deviation of entitlements for each position in the ranking and for each method, all in Fig. 4.

There are two key aspects to highlight from this experiment. First, the standard error in the case of TMC is much higher than the rest, as it is expected after observing the disagreement explained above. For example, the 1st and 2nd positions in the consortium of size 8 overlap. And so is the case for positions 4th, 5th, and 6th. The trend emphasizes as the consortium size increases, with a large overlap in the first 8 positions of the consortium of size 12. In contrast, the standard error of DSC is much smaller, again, demonstrating the larger agreement achieved.

Second, TMC is an unbiased estimator of the Shapley value, and DSC's absolute entitlements are very similar throughout the experiment to those of TMC. This demonstrates that, practically, DSC achieves similar entitlements at a fraction of the cost. In other words, participants get a much higher utility (5x as per RQ1) when in a consortium that uses DSC than alternatives. And this justifies empirically the protocol and algorithm design.

Different dataset. We run the experiment on the hotel dataset and confirm the trends remain (Figure 5b-5c): DSC's ranking is more stable than TMC and similar to Shapley's. The absolute entitlements also show a similar trend.

8 RELATED WORK

This paper presents the first data-sharing market, introducing incentives to participation and a protocol and algorithms to implement them. We discuss several related lines of work.

Incentive Mechanisms for Federated Learning. There is a plethora of related work in the area of incentive mechanisms for federated learning [67, 74]. Incentives are frequently designed for specific scenarios, such as reinsurance [62], medical data mining [21], and others. In contrast, our work is the first to elucidate a protocol: i) for general tasks; ii) implementing the sharing dominance and entitlement stake incentives; iii) and that consider the market sustainability as a design requirements. Many mechanisms seek to incentivize participants to participate by compensating for the costs they incur in using their own infrastructure to update the global model [74]; in contrast, the protocol we introduce seeks to achieve data sustainability. Whenever their data contribution is used to derive such compensation, existing work uses the Shapley value [34, 43]. We have argued against its use (and demonstrated our claims) when the consortia size is small, such as in scenarios involving a few big companies sharing data.

Federated databases and Data Federations. Federated databases such as the garlic project [45] permit querying over multiple conceptually separate databases. More recently, data federations propose querying over multiple databases, like federated databases, but including the security requirement of federated learning: that no data must leave the premise. Data federation systems such as Hu-Fu [68], SMCQL [12], SAQE [13] and [64] concentrate on the problem of securely sharing data when it is distributed. In contrast, our work focuses on the earlier problem of incentivizing participants to share data in the first place. Then, a data-sharing consortia is a group of participants that have agreed to share data with each other; a data federation is one concrete way in which such data-sharing consortia can implement sharing, see Section 3.1.

Data Management Research. Many efforts in data management focus on easing data sharing. The ORCHESTRA system [40] facilitates the curation and integration of datasets. Many primitives and techniques have been proposed to facilitate sharing data, such as materialized views [20], data versioning (e.g., OrpheusDB [73], datahub [14]) and more. In contrast, this paper focuses on incentivizing the creation of data-sharing consortia.

Today's data markets. There are many kinds of data markets. In individual-platform data markets, individuals barter data for services such as search, entertainment, and more. In these markets, platforms capture and exploit the value of data. Some initiatives, such as data dividends [26], data trusts [27], data cooperatives [25], and data-as-labor [61], propose to capture some of the value and distribute it back to the individuals. But these remain largely theoretical or focused on legal aspects. Online data marketplaces act as a storefront where owners show a list of their datasets and a price. Buyers willing to pay the price access the data. These markets are different than the data-sharing markets discussed in this work. Finally, there are more radical approaches to trading data and implementing the marketplace in a trust-less blockchain [11], or exploiting the properties of differential privacy [50]. None of these markets is geared towards creating data-sharing consortia.

Data lakes, commons, cooperatives. In many settings, participants are naturally incentivized to share data [47]. For example, in enterprise scenarios, data lakes accumulate datasets that different teams contribute with the expectation that other teams will benefit [7, 57, 60]. Similarly, data commons are a kind of data lake that is often used in the life sciences with the same objective of sharing data among interested researchers [37]. Both lakes and commons may host sensitive data but they pre-specify a policy to share that data so the sharer can sign and upload. In contrast, we focus on scenarios where there is not an incentive for participants to contribute their data.

Federated learning, homomorphic encryption, multi-party computation, blockchain. There are many technologies to facilitate data-sharing at the infrastructure level. In many scenarios, participants will not trust their data to intermediaries unless their data is *secure*. Different technologies focus on providing different degrees of *security*. In federated learning [48], participants do not need to share their data, but only some derived data products with a central server. Homomorphic encryption [56] and multi-party computation techniques [23] ensure that no plain data is ever visible to the other participants and yet they permit coordinate the execution of certain tasks. Finally, blockchain-based solutions [30] aim to share data without requiring a central trusted entity. All these technologies focus on the infrastructure level. None is concerned with the problem of incentivizing participants to want to share data in the first place.

Legal frameworks. Most existing data-sharing consortia are the result of a long tedious process involving data-sharing agreements and teams of lawyers who must anticipate the uses, misuses, and risks of sharing data among organizations, as well as the liabilities if there is a problem [5, 69]. These one-off negotiations mean that creating a new data-sharing consortium is often an entirely new endeavor and little effort from existing sharing agreements can be reused. Although in many scenarios data-sharing agreements remain necessary, the focus of our work is to incentivize the creation of data-sharing consortia in the first place.

9 CONCLUSIONS

In this paper, we introduced a new data sharing model, protocol, an algorithms (DSC) to incentivize participants to share data and solve a task of mutual interest. We designed a protocol that implements sharing dominance and entitlement stake incentives, incentivizing participation. The protocol compensates participants for the data they share. We argued that Shapley value-based solutions are not adequate for this kind of data-sharing market and then we proposed an alternative. The evaluation demonstrates our claims. DSC leads to the formation of more data-sharing consortia, successfully detect bad inputs, protecting participants, and it better compensates data contributions by using a cheaper-to-execute and similarly accurate revenue allocation function than the Shapley value.

While we demonstrated the new data-sharing protocol for scenarios where a consortium wishes to train a machine learning model on their joint data, there are interesting extensions for future work, such as cost-sharing the training of large, expensive (millions of dollars) machine learning models to increase their reach to smaller organizations, and the sharing of models, thus letting the platform create ensembles, or the results of analytical queries, with the intention of identify disagreements. All in all, the work is a step towards designing the abstractions necessary to enable data sharing markets.

REFERENCES

[1] Daniel Abadi, Owen Arden, Faisal Nawab, and Moshe Shadmon. 2020. Anylog: a grand unification of the internet of things. In *Conference on Innovative Data Systems Research (CIDR '20)*.

[2] Jacob D Abernethy, Rachel Cummings, Bhuvesh Kumar, Sam Taggart, and Jamie H Morgenstern. 2019. Learning auctions with robust incentive guarantees. *Advances in Neural Information Processing Systems* 32 (2019).

[3] Daron Acemoglu, Ali Makhdoumi, Azarakhsh Malekian, and Asuman Ozdaglar. 2019. *Too much data: Prices and inefficiencies in data markets.* Technical Report. National Bureau of Economic Research.

[4] Anish Agarwal, Munther Dahleh, and Tuhin Sarkar. 2019. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*. 701–726.

[5] Claudia Allen, Terrisca R Des Jardins, Arvela Heider, Kristin A Lyman, Lee McWilliams, Alison L Rein, Abigail A Schachter, Ranjit Singh, Barbara Sorondo, Joan Topper, et al. 2014. Data governance and data sharing agreements for community-wide health information exchange: lessons from the beacon communities. *EGEMS* 2, 1 (2014).

[6] Nuno Antonio, Ana de Almeida, and Luis Nunes. 2019. Hotel booking demand datasets. *Data in brief* 22 (2019), 41–49.

[7] Michael Armbrust, Ali Ghodsi, Reynold Xin, and Matei Zaharia. 2021. Lakehouse: a new generation of open platforms that unify data warehousing and advanced analytics. In *Proceedings of CIDR*.

[8] Kenneth Arrow. 1962. Economic welfare and the allocation of resources for invention. In *The rate and direction of inventive activity: Economic and social factors*. Princeton University Press, 609–626.

[9] Lawrence M Ausubel and Peter Cramton. 2002. Demand reduction and inefficiency in multi-unit auctions. (2002).

[10] Amazon AWS. 2022. Amazon AWS Instance Types. https://aws.amazon.com/ec2/instance-types/

[11] Shaimaa Bajoudah, Dong Changyu, and Paolo Missier. 2019. Toward a Decentralized, Trust-less Marketplace for Brokered IoT Data Trading using Blockchain. In *Procs. 2nd IEEE International Conference on Blockchain (Blockchain 2019)*. IEEE, Atlanta, USA.

[12] Johes Bater, Gregory Elliott, Craig Eggen, Satyender Goel, Abel Kho, and Jennie Rogers. 2016. SMCQL: Secure querying for federated databases. *arXiv preprint arXiv:1606.06808* (2016).

[13] Johes Bater, Yongjoo Park, Xi He, Xiao Wang, and Jennie Rogers. 2020. Saqe: practical privacy-preserving approximate query processing for data federations. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2691–2705.

[14] Anant Bhardwaj, Souvik Bhattacherjee, Amit Chavan, Amol Deshpande, Aaron J Elmore, Samuel Madden, and Aditya G Parameswaran. [n.d.]. Datahub: Collaborative data science & dataset version management at scale. ([n. d.]).

[15] Christine L Borgman. 2012. The conundrum of sharing research data. *Journal of the American Society for Information Science and Technology* 63, 6 (2012), 1059–1078.

[16] Steven J Brams, Steven John Brams, and Alan D Taylor. 1996. *Fair Division: From cake-cutting to dispute resolution.* Cambridge University Press.

[17] Anna L Buczak and Erhan Guven. 2015. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials* 18, 2 (2015), 1153–1176.

[18] Raul Castro Fernandez. 2022. Protecting Data Markets from Strategic Participants. (2022).

[19] Victor Chernozhukov, Hiroyuki Kasahara, and Paul Schrimpf. 2021. Causal impact of masks, policies, behavior on early covid-19 pandemic in the US. *Journal of econometrics* 220, 1 (2021), 23–62.

[20] Rada Chirkova, Jun Yang, et al. 2012. Materialized views. *Foundations and Trends® in Databases* 4, 4 (2012), 295–405.

[21] Feature Cloud. 2022. Transforming medical research with federated learning. https://featurecloud.eu/about/our-vision/

[22] Thomas M Cover. 1999. *Elements of information theory.* John Wiley & Sons.

[23] Ronald Cramer, Ivan Bjerre Damgård, et al. 2015. *Secure multiparty computation.* Cambridge University Press.

[24] RALPH D'AGOSTINO and Egon S Pearson. 1973. Tests for departure from normality. Empirical results for the distributions of b 2 and sqrt(b). *Biometrika* 60, 3 (1973), 613–622.

[25] datacoop 2021. Mozilla Research. Shifting power through data governance. https://foundation.mozilla.org/en/data-futures-lab/data-for-empowerment/shifting-power-through-data-governance/.

[26] datadividend 2021. Data Dividend, My data, my money. https://www.datadividendproject.com/.

[27] Sylvie Delacroix and Neil D Lawrence. 2019. Bottom-up data Trusts: disturbing the 'one size fits all'approach to data governance. *International data privacy law* 9, 4 (2019), 236–252.

[28] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. 2018. CatBoost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363* (2018).

[29] Dheeru Dua and Casey Graff. 2017. UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

[30] Muhammad El-Hindi, Carsten Binnig, Arvind Arasu, Donald Kossmann, and Ravi Ramamurthy. 2019. BlockchainDB: A shared database on blockchains. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1597–1609.

[31] André Elisseeff, Massimiliano Pontil, et al. 2003. Leave-one-out error and stability of learning algorithms with applications. *NATO science series sub series iii computer and systems sciences* 190 (2003), 111–130.

[32] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1001–1012.

[33] Raul Castro Fernandez, Pranav Subramaniam, and Michael J Franklin. 2020. Data market platforms: Trading data assets to solve data problems. *arXiv preprint arXiv:2002.01047* (2020).

[34] Amirata Ghorbani and James Zou. 2019. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*. PMLR, 2242–2251.

[35] Andrew V Goldberg and Jason D Hartline. 2001. Competitive auctions for multiple digital goods. In *European Symposium on Algorithms*. Springer, 416–427.

[36] Google. 2022. What-If Tool - People + AI Research (PAIR). https://pair-code.github.io/what-if-tool/

[37] Robert L Grossman, Allison Heath, Mark Murphy, Maria Patterson, and Walt Wells. 2016. A case for data commons: toward data science as a service. *Computing in science & engineering* 18, 5 (2016), 10–20.

[38] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

[39] Ling Huang, Anthony D Joseph, Blaine Nelson, Benjamin IP Rubinstein, and J Doug Tygar. 2011. Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*. 43–58.

[40] Zachary G Ives, Todd J Green, Grigoris Karvounarakis, Nicholas E Taylor, Val Tannen, Partha Pratim Talukdar, Marie Jacob, and Fernando Pereira. 2008. The orchestra collaborative data sharing system. *ACM Sigmod Record* 37, 3 (2008), 26–32.

[41] Marijn Janssen, Yannis Charalabidis, and Anneke Zuiderwijk. 2012. Benefits, adoption barriers and myths of open data and open government. *Information systems management* 29, 4 (2012), 258–268.

[42] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nezihe Merve Gurel, Bo Li, Ce Zhang, Costas Spanos, and Dawn Song. 2019. Efficient task-specific data valuation for nearest neighbor algorithms. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1610–1623.

[43] Ruoxi Jia, David Dao, Boxin Wang, Frances Ann Hubis, Nick Hynes, Nezihe Merve Gürel, Bo Li, Ce Zhang, Dawn Song, and Costas J Spanos. 2019. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 1167–1176.

[44] Charles I Jones and Christopher Tonetti. 2020. Nonrivalry and the Economics of Data. *American Economic Review* 110, 9 (2020), 2819–58.

[45] Vanja Josifovski, Peter Schwarz, Laura Haas, and Eileen Lin. 2002. Garlic: a new flavor of federated query processing for DB2. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. 524–532.

[46] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. 2020. Lessons Learned from the Chameleon Testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association.

[47] Rob Kitchin. 2014. *The data revolution: Big data, open data, data infrastructures and their consequences.* Sage.

[48] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine* 37, 3 (2020), 50–60.

[49] Yifan Li, Xiaohui Yu, and Nick Koudas. 2021. Data Acquisition for Improving Machine Learning Models. *VLDB* 14, 10 (jun 2021), 1832–1844.

[50] Qiongqiong Lin, Jiayao Zhang, Jinfei Liu, Kui Ren, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. 2021. Demonstration of dealer: an end-to-end model

marketplace with differential privacy. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2747–2750.
[51] Jinfei Liu, Jian Lou, Junxu Liu, Li Xiong, Jian Pei, and Jimeng Sun. 2021. Dealer: an end-to-end model marketplace with differential privacy. *VLDB* (2021).
[52] Yu-Chen Lo, Stefano E Rensi, Wen Torng, and Russ B Altman. 2018. Machine learning in chemoinformatics and drug discovery. *Drug discovery today* 23, 8 (2018), 1538–1546.
[53] RE Machol and J Rosenblatt. 1966. Confidence interval based on single observation. *Proc. IEEE* 54, 8 (1966), 1087–1088.
[54] Roger B Myerson. 1981. Optimal auction design. *Mathematics of operations research* 6, 1 (1981), 58–73.
[55] Roger B Myerson and Mark A Satterthwaite. 1983. Efficient mechanisms for bilateral trading. *Journal of economic theory* 29, 2 (1983), 265–281.
[56] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. 2011. Can homomorphic encryption be practical?. In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*. 113–124.
[57] Fatemeh Nargesian, Erkang Zhu, Renée J Miller, Ken Q Pu, and Patricia C Arocena. 2019. Data lake management: challenges and opportunities. *Proceedings of the VLDB Endowment* 12, 12 (2019), 1986–1989.
[58] NIH. 2023. Final NIH Policy for Data Management and Sharing. https://grants.nih.gov/grants/guide/notice-files/NOT-OD-21-013.html
[59] Elinor Ostrom. 2008. Tragedy of the commons. *The new palgrave dictionary of economics* 2 (2008).
[60] Ippokratis Pandis. 2021. The evolution of Amazon redshift. *Proceedings of the VLDB Endowment* 14, 12 (2021), 3162–3174.
[61] Eric A Posner and E Glen Weyl. 2019. *Radical Markets*. Princeton University Press.
[62] Swiss Re. 2022. Swiss Re to explore AI in reinsurance. https://www.lifeinsuranceinternational.com/news/swiss-re-webank/
[63] Alvin E Roth. 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.
[64] Yexuan Shi, Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Bolin Ding, and Lei Chen. 2021. Efficient Approximate Range Aggregation over Large-scale Spatial Data Federation. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[65] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. 2017. Machine learning models that remember too much. In *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*. 587–601.
[66] Vasilis Syrgkanis and Eva Tardos. 2013. Composable and efficient mechanisms. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. 211–220.
[67] Ming Tang and Vincent WS Wong. 2021. An incentive mechanism for cross-silo federated learning: A public goods perspective. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.
[68] Yongxin Tong, Xuchen Pan, Yuxiang Zeng, Yexuan Shi, Chunbo Xue, Zimu Zhou, Xiaofei Zhang, Lei Chen, Yi Xu, Ke Xu, et al. 2022. Hu-Fu: efficient and secure spatial queries over data federation. *VLDB* (2022).
[69] USGS. 2022. USGS Data-Sharing Agreement. https://www.usgs.gov/data-management/data-sharing-agreements
[70] Melanie M Wall, James Boen, and Richard Tweedie. 2001. An effective confidence interval for the mean with samples of size one and two. *The American Statistician* 55, 2 (2001), 102–105.
[71] Tianhao Wang, Johannes Rausch, Ce Zhang, Ruoxi Jia, and Dawn Song. 2020. A principled approach to data valuation for federated learning. In *Federated Learning*. Springer, 153–167.
[72] Siyuan Xia, Zhiru Zhu, Chris Zhu, Jinjin Zhao, Kyle Chard, Aaron J Elmore, Ian Foster, Michael Franklin, Sanjay Krishnan, and Raul Castro Fernandez. 2022. Data station: delegated, trustworthy, and auditable computation to enable data-sharing consortia with a data escrow. *Proceedings of the VLDB Endowment* 15, 11 (2022), 3172–3185.
[73] Liqi Xu, Silu Huang, SiLi Hui, Aaron J Elmore, and Aditya Parameswaran. 2017. Orpheusdb: a lightweight approach to relational dataset versioning. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1655–1658.
[74] Rongfei Zeng, Chao Zeng, Xingwei Wang, Bo Li, and Xiaowen Chu. 2021. A comprehensive survey of incentive mechanism for federated learning. *arXiv preprint arXiv:2106.15406* (2021).