FedServing: A Federated Prediction Serving Framework Based on Incentive Mechanism

Jiasi Weng*, Jian Weng*, Hongwei Huang*, Chengjun Cai†, and Cong Wang†
*Jinan University; †City University of Hong Kong

Abstract—Data holders, such as mobile apps, hospitals and banks, are capable of training machine learning (ML) models and enjoy many intelligence services. To benefit more individuals lacking data and models, a convenient approach is needed which enables the trained models from various sources for prediction serving, but it has yet to truly take off considering three issues: (i) incentivizing prediction truthfulness; (ii) boosting prediction accuracy; (iii) protecting model privacy.

We design FedServing, a federated prediction serving framework, achieving the three issues. First, we customize an incentive mechanism based on Bayesian game theory which ensures that joining providers at a Bayesian Nash Equilibrium will provide truthful (not meaningless) predictions. Second, working jointly with the incentive mechanism, we employ truth discovery algorithms to aggregate truthful but possibly inaccurate predictions for boosting prediction accuracy. Third, providers can locally deploy their models and their predictions are securely aggregated inside TEEs. Attractively, our design supports popular prediction formats, including top-1 label, ranked labels and posterior probability. Besides, blockchain is employed as a complementary component to enforce exchange fairness. By conducting extensive experiments, we validate the expected properties of our design. We also empirically demonstrate that FedServing reduces the risk of certain membership inference attack.

Index Terms—Prediction serving, Incentive mechanism, Privacy, Aggregation

I. INTRODUCTION

Machine learning (ML) is revolutionizing our world and the global market for ML driven services is expected to reach $5.330 billion by 2024.[1]. Many data holders, such as mobile apps, hospitals and banks, are able to train models based on the available data they hold, and use the trained models to achieve functionality and business innovation.[2]. From another perspective, most individuals lacking data and power are incapable of training models, so that they hardly benefit from ML. Even if an individual is in possession of a model, it still has the real-world demand to collaborate with others’ models, demonstrated by an existing real-world case, i.e., two banks in North America collaborate to detect money laundering. Obviously, due to privacy concerns, intellectual property issues or business competition, model owners are unwilling to share their trained models. Thus, it is necessary to build a bridge which connects model owners who have no incentives of sharing models with individuals who need models.

Building such a bridge inevitably needs to support three essential requirements as following: (i) providing sufficient incentives to the model owners so that they are willing to contribute their models; (ii) enabling individual users of interest to enjoy as high-performance as possible models; (iii) guaranteeing model privacy, since models imply private information about their training data.[3]. However, there exists no work to realize such a bridge, so that it has yet to truly take off.

While Machine-Learning-as-a-Service (MLaaS) platforms enable monetizing models for prediction serving on a pay-per-query basis, trained models have to reside on the untrusted servers, causing model privacy concerns. Although earlier works present effective approaches[4,6] for protecting models against the untrusted servers, they still are not really satisfactory. Specifically, cryptographic methods are computation-consuming and inefficient when handling large-sized models[4]. but high-performance models usually are large. Differential privacy based defenses would sacrifice prediction accuracy[5]. Trusted hardware-enabled approaches are relatively practical but still have efficiency limitation. It is due to that trusted hardwares are majorly restricted to CPUs, but running large models usually needs GPUs[6].

Motivated by our observations, our goal is to make model owners freely deploy their models without limits, collectively contribute their models to make profits and securely use models without privacy leakage concerns. Towards the goal, we present a federated prediction serving framework, FedServing, towards model owners from various sources in an open setting. Our starting point is allowing model owners to deploy models at local devices and provide aggregated predictions for exchanging with monetary rewards. Standing on top of it, we especially make efforts to design our solution for enforcing prediction accuracy, due to the following two-fold challenges: Challenge (i): Strategic behaviors of model owners. Model owners (hereafter called as providers) are likely to be rational and selfish, so that they may be strategic to report meaningless predictions without effort. In addition, ground truths with respect to given prediction queries are usually unknown, which makes the truthfulness of predictions hard to be verified.

Challenge (ii): Varying quality of models. While aggregating predictions (e.g., via majority voting or averaging) is a classic strategy for improving accuracy, they may be less effective in our case. The issue of majority voting and averaging is that they assume prediction sources (i.e., models) are equally reliable. Yet, the assumption cannot hold in our open setting. It is due to that (1) the qualities of models from various sources are varying and due to local deployment, there is no available authority enforcing the model quality upon answering prediction.
queries; (2) even well-trained models are not always general-
ized well over the whole feature space of all prediction queries,
so producing predictions are probably not always accurate.

In light of the two challenging issues, the state-of-the-
art solutions usually resort to incentive mechanisms in
conjunction with quality-aware aggregation algorithms, as
learnt from the literature \cite{7,12}. Unfortunately, previous
work cannot be used to mitigate our challenging issues.
The main reason is that they fail to simultaneously handle
categorical and continuous data covered by popular prediction
outputs \cite{13}. As a concrete instance, the prediction outputs
for each query of a sentiment analysis task can be top-1 label,
\textit{e.g.}, [upset], ranked labels, \textit{e.g.}, [upset, scared,
distressed, guilty], and posterior probability for each label,
\textit{e.g.}, \langle95.0\%, 2.0\%, 1.0\%, 2.0\%\rangle, which are supported by
Google Photos and Google Cloud Vision API, for example.

Our key design. We customize a complementary mecha-
nism by integrating an incentive design with “truth-finding”
algorithms. Concretely, our mechanism (1) uses \textit{Bayesian
game theory} to model the honest and strategic behaviors
of providers and ensures the existence of a \textit{Bayesian Nash
Equilibrium}, where all providers will offer truthful (rather
than meaningless) predictions for given prediction queries;
(2) employs \textit{truth discovery (TD) algorithms} to learn highly
accurate predictions from the truthful (but possibly inaccurate)
predictions to eliminate the effect of inaccurate predictions;
(3) allocates the providers with fair rewards in proportion to the
truthfulness of their predictions; (4) simultaneously handles
prediction output formats including labels and the respective
posterior probability.

Despite that models are locally deployed, privacy concerns
still exist due to disclosing predictions of a model. Concretely,
a model’s predictions can be exploited to infer if a data record
was used to train a model, \textit{e.g.}, identifying if an individual
was a patient at the hospital, known as membership inference
attacks \cite{14}. To address the privacy concern, we leverage
trusted execution environments (TEEs) to aggregate predic-
tions from multiple providers, and only aggregated predictions
are revealed to users \cite{15}. Owing to the confidentiality and
integrity provided by TEEs, a model’s predictions are not
revealed and aggregated predictions are correctly generated.

It is noteworthy that our proposed incentive mechanism also
can benefit from the TEEs’ integrity, since the procedure of
evaluating the truthfulness of predictions from each model
can be correctly executed, which further enforces fair rewards
guided by the truthfulness. Notably, we do not use privacy-
preserving verifiable cryptography, considering that TEEs are
relatively more performant.

Besides, we need to facilitate an open setting for model
owners from various sources freely joining in FedServing.
But meanwhile, we also need a regulation complementary
to our incentive mechanism for fulfilling the transparent
process of money settlement and deterring providers’ selfish
behaviors, \textit{e.g.}, abortion, thereby achieving the fairness of
money-prediction exchange among users and providers. In
light of the issues, we choose blockchain to facilitate the
open setting and enforce the regulation.

We note that FedServing can be extended to support the
existing prediction serving systems and now we shed light on
the service manner of our FedServing framework. A prediction
serving server can deploy a smart contract as a uniform query
interface for charging users and as an entrance for participating
providers. When receiving the query and fees from a user, the
server resorts to its off-chain TEEs-empowered component to
collect predictions from participating providers who undertake
the prediction task. The TEE strategically aggregates predic-
tions and submits aggregated predictions to the blockchain.
Finally, the user obtains the predictions and meanwhile the
smart contract allocates the user’s fees to the participating
providers according to the truthfulness of their predictions.

In conclusion, this paper makes the main contributions as
following:

- We propose a federated prediction serving framework
  empowered by the blockchain, providing an as accurate
  as possible prediction service with truthful contributions
  from various source models in an open setting.
- We customize an incentive mechanism for eliciting truth-
  ful contributions, by carefully applying a technique of
  peer prediction \cite{16} and fully respecting the formats of
  popular prediction outputs.
- We extend a widely-adopted truthful discovery algorithm
  to support our prediction setting, and make it jointly work
  with our designed incentive mechanism, finally producing
  as accurate as possible predictions.
- We implement our design and conduct extensive experi-
  ments in terms of the performance, validity and ability
  against a privacy attack. For reproducibility, our code
  is publicly available at \url{https://github.com/H-W-Huang/
FedServing}.

II. RELATED WORK

\textbf{Prediction Serving System.} Existing excellent systems \cite{17,18}
centralized manage models as well as deploy models
for low-latency and high-throughput prediction serving, where
models are off-the-shelf. To enhance prediction accuracy, they
generally support ensemble models which aggregate predic-
tions from multiple models.

Different from them, our work focuses on the models from
various sources for prediction serving in an open setting. More
precisely, we consider how to incentivize model owners from
various sources to provide truthful prediction services while
respecting model privacy and ensuring prediction accuracy.
To the end, we present a distributed framework which achieves
the following three-fold components which are less considered
by the existing systems \cite{17,18}.

(i) \textbf{Pricing mechanism.} We customize a pricing mechanism
for compensating participating providers and incentivizing
prediction truthfulness, instead of using an one-price-fits-
all pricing structure, which still respects the pay-per-query
business pattern of the current MLaaS platforms.

(ii) \textbf{Quality-aware aggregation.} Considering that the model
quality and the ground truths of prediction queries are un-
known in our open setting, we use TD algorithms to aggregate predictions rather than simply averaging, thereby eliminating the effect of low-accuracy predictions.

(iii) Model and prediction protection. We make models never leave local devices and multiple predictions are securely aggregated inside TEEs so that users only obtain aggregated predictions. Due to local deployment, providers retain control over when and how their in-house models are used to make predictions, e.g., joining in an ensemble to produce predictions, thereby reducing the risks of privacy attacks [14].

Incentive Mechanism. Prior incentive mechanisms [7–12, 19, 34] are designed for stimulating participation by compensating workers’ costs with monetary rewards, and implementing the economic properties, such as platform profit maximization, individual rationality and budget feasibility, which greatly promote the development of crowdsourcing. The incentive mechanisms generally resort to game-theoretic methods, such as reverse auction [21, 22, 23, 26, 31], double auction [23, 34] and all-pay auction [20], or other game theory [9, 11, 33]. With the game-theoretic analysis, they consider the strategic behaviors of workers and investigate how to encourage workers to behave truthfully.

In this paper, we aim to stimulate the truthfulness of collective predictions considering providers’ strategic behaviors of providing meaningless predictions, thereby achieving quality control. Existing mechanisms do not solve our problem, since the following four requirements cannot be simultaneously satisfied:

(i) Incentivizing truthfulness. Most mechanisms [21, 22, 23, 26] focus on incentivizing workers to reveal their costs truthfully. A few excellent mechanisms like [9, 11] incentivize the truthfulness of crowd data as this paper, but they are unsatisfactory to us due to that (ii) below cannot be supported.

(ii) Simultaneously handling categorical and continuous data. Theseus [9] proposes a truthful mechanism for quality and efforts elicitation while focusing on continuous sensing data. [11] creatively studies the joint elicitation of quality, efforts and data while focusing on discrete data (more precisely, binary data). Their techniques do not solve our problem, since we simultaneously consider labels and confidence values which are categorical and continuous data, respectively.

(iii) No reliance on prior knowledge. Prior arts [8, 23] assume the prior knowledge about workers’ reliability or reputation that helps allocate rewards. We do not use such prior knowledge, since a provider’s predictions for historical tasks are considered irrelevant to the current task.

(iv) Jointly addressing incentive and quality concerns. Most mechanisms for incentivizing truthfulness do not jointly work with TD, except to [7, 10, 12]. These works can be deployed with TD, but they still are unsatisfactory to us: Theseus [9, 8] and [12] majorly consider the continuous data stream; [7] does not focus on workers’ strategic behaviors; [10] cares about binary answers and assumes that most workers are reliable.

1We note that a prediction task usually is requested with batch queries, where the queries belonging to the same task are relevant.

III. SYSTEM OVERVIEW

In this section, we present our FedServing framework. It begins with the system model, and then figures out the threat assumptions and design goals.

![Fig. 1: Overview of the FedServing framework.](image)

A. System Model

At the high level, our FedServing consists of four entities as shown in Fig. 1: prediction providers, user, smart contract and TEEs-based server. Specifically, the prediction providers who own various ML models monetize their prediction query services on the blockchain (e.g., Ethereum). They could publicize non-private model profiles like service APIs for user accessing their models. User is able to browse model profiles on the blockchain, and query prediction services via a smart contract, named as prediction serving contract (PS contract). PS contract aims at receiving the user’s query request, relaying the request, receiving aggregated predictions and achieving the fair payment finalization. As the intermediator between the prediction providers and the PS contract, the TEEs-based server is responsible for strategically aggregating predictions sent by multiple providers, and calculating accuracy-aware scores used to guide allocating rewards. The basic workflow in Fig. 1 is described as following:

1 User sends a transaction which contains a description about her requested task, e.g., a sentiment analysis task, and makes a deposit for payment to the PS contract. Note that the task’s input data, e.g., text files, can be securely stored in an accessible system like IPFS, and then be securely authorized to participating providers.

2 Providers participate in the task by submitting a deposit to the PS contract for potential penalty, e.g., punishing abortion. Here, we omit the phase that they can authentically obtain the task’s input data from IPFS.

3 Participating providers evaluate local models on the input data, and lastly submit predictions to the TEE via an authenticated communication channel.

4 The TEE strategically aggregates predictions from multiple participating providers and compute accuracy-aware scores for each provider. After that, the aggregated predictions are correctly encrypted using the user’s public key and submitted to the blockchain.

5 User retrieves and decrypts the aggregated predictions using her private key, and meanwhile, her deposit is
allocated to the participating providers according to the respective accuracy-aware scores.

B. Threat Model and Assumptions

Prediction Provider. We consider that prediction providers are rational and self-interested. They may act to maximize their profits by submitting arbitrary predictions. The providers answering certain query are named as participating providers and assumed not to collude with each others. In addition, we assume that the input data received by participating providers are benign; perturbed input data known as adversarial examples are out of our consideration.

TEEs. We trust that TEEs, e.g., Intel Software Guard Extensions (SGX), can securely execute specific programs against external observation and manipulation, i.e., ensuring confidentiality and integrity. We note that side-channel attacks and rollback attacks on TEEs are out of the scope of this paper like prior TEEs-empowered work, owing to many off-the-shelf defence mechanisms, e.g., Intel SGX’s Enhanced Privacy ID (EPID) remote attestation protocol.

Blockchain. We trust the blockchain for integrity and availability. Smart contract autonomously and faithfully executes defined functions, e.g., correctly locking deposits and settling rewards, which is assumed not vulnerable to software bugs.

Remarks. We aware that FedServing can suffer from Sybil attacks, where a prediction provider may maliciously use multiple fake accounts to join in certain task. For demoralizing Sybil attacks, a widely adopted solution is to increase the attack cost like solving proof-of-work puzzles and making deposits. In this paper, we require each participating provider to make a deposit before undertaking a task.

C. Design Goals

Truthfulness and accuracy. It means that user can obtain aggregated predictions with truthfulness and accuracy guarantees. Specifically, each participating provider provides truthful (but possibly inaccurate) predictions, and meanwhile, the truth discovery algorithm is correctly conducted on the provided truthful predictions to produce truths, i.e., aggregated predictions, which are regarded accurate enough.

Fairness. It includes the fairness of reward allocation and the fairness of money-prediction exchange. First, each participating provider in a task gets a fair reward guided by a strictly proper score which is computed based on the truthfulness of their predictions. A comparatively truthful prediction leads to a higher score, and the prediction’s provider obtains comparatively more rewards. Second, all participating providers receive rewards if the user obtains the final predictions.

IV. DESIGN OF PREDICTION AGGREGATION

Considering that our FedServing is built in an open setting, participating models might produce inaccurate predictions. The reasons include that (i) varying quality models can freely participate in FedServing, and meanwhile, there is no available authority enforcing the quality of participating models; (ii) trained models are not always generalized well over the whole feature space of every prediction task.

In light of this issue, we study the lessons from the earlier works and leverage TD algorithms to aggregate predictions, so as to learn as accurate predictions as possible from varying quality models in absence of ground truth. We support three common prediction formats in practice. To the best of our knowledge, there is no existing scheme dealing with the issue as this paper. The previous work is similar to our design of prediction aggregation, but it focuses on one single format, i.e., probability vector. We especially consider other popular prediction outputs, e.g., ranked label list, used in Google Photos.

For ease of presentation, we begin with an instance of prediction task. Then, we elaborate three prediction formats and demonstrate how to aggregate them.

Instance Description. Suppose that a social psychologist has a sentiment analysis task for a set of consulting letters from anonymous citizens. She needs to label the set of consulting letters with the emotion states for studying social projection. With the task, she can query the PS contract in our FedServing: what are the emotion states for each consulting letter, distressed, upset, guilty or scared?

Prediction Formats. In the above instance, we introduce three popular prediction output formats: (1) Abstract: a top-1 class label, e.g., ‘upset’, (2) Rank: a ranked list of labels, e.g., [upset, scared, distressed, guilty], and (3) Measurement: a probability vector for possible class labels, e.g., \(\{2.0\%, 95.0\%, 1.0\%, 2.0\%\}\) for \([\text{distressed}, \text{upset}, \text{guilty}, \text{scared}]\) (their sum is 100%).

Apparently, the measurement output contains the most detailed prediction information while the abstract output contains less information. Note that here we mainly discuss classification tasks, but our method can be easily extended to regression tasks which are associated with real-valued predictions.

Prediction Aggregation. We now introduce the algorithm to aggregate predictions adapted to the three formats. Specifically, in order to fluently run the truth discovery algorithm as shown in Algorithm 1, we carefully transform the later two formats into continuous data vectors. For ease of explanation, we suppose that there are three models predicting a given consulting letter with the corresponding label list \([\text{distressed}, \text{upset}, \text{guilty}, \text{scared}]\). Their predictions with respect to the three formats are demonstrated in TABLE II.

We now explain how we uniformly represent the three-format predictions by using continuous data vectors. For the abstract format, the three models separately produce labels ‘upset’, ‘distressed’ and ‘distressed’. We transform them into the corresponding 0/1 value vectors, where the index with value 1 is the most possible label, as shown in the abstract row of TABLE I

For the rank format, the three models provide the ranked lists of possible labels as presented in TABLE I.

For example, a ranked list \([\text{upset}, \text{scared}, \text{distressed}, \text{guilty}]\) is given by the first model. We set ranked integer values to each ranking level. A largest integer represents the highest
They are represented as \( \varepsilon \) outputs the truths as the aggregated predictions \( v \) data vector label, where the length vector containing the confidence values for each class is the number of given possible class labels. Thus, the strategy \( \varepsilon \) is from discrete set \( \Omega \) including two steps. Specifically, we suppose that there are multiple predictions from \( m \) \((m \geq 3)\) providers for \( n \) consulting letters. Each prediction is a \( c \)-length vector containing the confidence values for each class label, where \( c \) is the number of given possible class labels. They are represented as \( \{I^j\}_{i=1}^{m,n} \), where \( I^j \) is a continuous data vector \( v^j = (v^j_{1}, ..., v^j_{c}) \). Now, with Algorithm 1 we iteratively estimate the truths on \( \{I^j\}_{i=1}^{m,n} \) and update \( m \) providers’ weights until convergence. The algorithm finally outputs the truths as the aggregated predictions \( \{O^{(c)}\}_{j=1}^{n} \) with respect to each consulting letter.

### Algorithm 1 Truth discovery

**Input:** provider predictions \( \{I^j\}_{i=1}^{m,n} \)

**Output:** truth predictions \( \{O^{(c)}\}_{j=1}^{n} \)

1. Initialize \( r = 1 \) and weights \( \{w^i(r) = 1\}_{i=1}^{m} \).
2. repeat
   3. for each \( j \in [1,n] \) do
      4. \( O^{(r+1)} \leftarrow \frac{\sum_{i=1}^{m} w^i(r) I^j_i}{\sum_{i=1}^{m} w^i} \) \((1)\)
   5. end for
   6. for each \( i \in [1,m] \) do
      7. \( w^i(r+1) \leftarrow -\log \left( \frac{\sum_{j=1}^{n} h_{loss}(O^{(r+1)}, I^j_i)}{\sum_{k=1}^{n} h_{loss}(O^{(r+1)}, I^k_i)} \right) \) \((2)\)
   8. end for
9. \( r = r + 1 \)
10. until \( r \leq \varepsilon \)
11. return \( \{O^{(c)}\}_{j=1}^{n} \)

Initially, we set each provider’s weight with 1 and denote an iteration threshold \( \varepsilon \). Then, with fixed weights, \( m \) providers’ predictions are aggregated via the weighted mean method (Step (1)). During the iterative computation, the aggregated predictions are closer to that of the providers having higher weights. With the aggregated predictions, each provider’s weight is updated based on the distances between his predictions and the aggregated predictions with respect to \( n \) consulting letters (Step (2)). The provider whose predictions are closer to the aggregated predictions will be assigned with a higher weight. Here, the loss function \( f_{loss}(\cdot) \) is used to characterize the distance and specifically, we use the normalized squared loss function. Step (1) and (2) are iteratively computed until \( r \) reaches pre-defined threshold \( \varepsilon \).

### V. DESIGN OF PRICING MECHANISM

The previous section introduces the process of aggregating predictions with the aim to filter out less accurate predictions. Yet, the accuracy of aggregated predictions still cannot be guaranteed if a majority of self-interested providers offer meaningless predictions. In order to motivate the self-interested providers to provide truthful predictions, we jointly design our pricing mechanism by employing the Bayesian game theory. Notably, predictions contain categorical and continuous data which will be simultaneously handled.

This section begins with the setting definitions and design objectives, and then presents the pricing mechanism formulation and an approximate solution. To the end, an analysis for the proposed pricing mechanism is elaborated.

#### A. Mechanism Setting

We use the game theory method to model the strategic behaviors of participating providers inspired by the works [9][11]. Concretely, we model participating providers \( P = \{i, ..., m\} \) playing a non-cooperative game, where each of them independently gives a private prediction for each query requested by a certain user. Note that a requested task can include multiple queries, e.g., labeling multiple consulting letters.

In the game, participating providers behave as utility maximizers. They behave strategically by evaluating their expected utility. Specifically, they will not participate if the expected utility is negative, and otherwise, they offer predictions via a specific strategy that maximizes the expected utility. In general, the evaluation needs some technical assumptions [16]. We assume that participating providers undertaking the same task have a common prior belief, and meanwhile, they use the same belief updating procedure, i.e., Bayes’ rule.

A provider’s behavior is described by strategy. A strategy is denoted by \( s = (l, v) \) meaning giving a prediction for a query \( l \), or \( \perp \) meaning abort. Herein, \( l \) is a list of claimed possible class labels and each label in \( l \) is from discrete set \( \Omega \); \( v \) is the corresponding posterior probability values which are drawn from probability density distributions \( \Psi \). Thus, the strategy space is \( (\Omega, \Psi) \cup \{\perp\} \). Then, the participating providers’ strategy profile is \( S = (s_1, ..., s_m) \), if we suppose that there are \( m \) participating providers.

Next, we continue to formulate the provider model, the user model and a Bayesian Nash Equilibrium for providers.

**Provider Model.** Within the defined game, a provider’s payoff depends on his own strategy with regard to other providers’ strategies. Specifically, given a payment function \( p(\cdot) \), a cost
function \( c(\cdot) \) and deposit \( d_0 \), we define any provider’s utility \( u_i(S), i \in P \) in a game with a strategy profile \( S \) as following:

\[
u_i(S) = p_i(S) - c(s_i) - d_0.
\]

Next, any provider can evaluate the expected utility:

\[
\mathbb{E}_{S_{-s_i}}[u_i(s_i, S_{-s_i})] = \mathbb{E}_{S_{-s_i}}[p_i(s_i, S_{-s_i})] - c(s_i) - d_0,
\]

where \( S_{-s_i} \) is the strategy profile excluding \( s_i \). Note that a participating provider’s deposit for \( n \) queries is \( d = n \times d_0 \).

**User Model.** A user’s objective is to obtain the aggregated predictions whose accuracy is as close as possible to the truth accuracy. To exchange the aggregated predictions \( \{v^j\}_{j=1, \ldots, n} \) of \( n \) queries from \( m \) participating providers, she makes amount of deposits, namely budget \( B \), on the blockchain. Assume that the market publicizes budget curves relative to the number of employed providers via market survey. With the budget curves, the user deposits a budget level that enables soliciting certain number of prediction providers.

**Bayesian Nash Equilibrium.** A strategy profile \( S^* \) is denoted as a Bayesian Nash Equilibrium (BNE) in the defined game, if no provider \( i \in P \) can increase her expected utility by changing the current strategy \( s_i^* \) with regard to other providers’ strategies \( S_{-s_i}^* \):

\[
\mathbb{E}_{S_{-s_i}}[u_i(s_i^*, S_{-s_i}^*)] \geq \mathbb{E}_{S_{-s_i}}[u_i(s_i^*, S_{-s_i}^*)].
\]

At the BNE, our mechanism aims to achieve several design objectives in Section V-B.

**B. Design Objectives**

With the strategy \( S^* \) at the BNE, we state three design objectives below.

**Definition 1. (Truthfulness)** An aggregated prediction for a query is truthful if and only if (i) the aggregation computation is correctly executed, and meanwhile, (ii) every participating provider \( i \in P \) at BNE \( S^* \) provides a prediction \( s_i^* = (1, v_i) \) satisfying the following condition:

\[
I_i = \mathbb{P} \land D_{KL}(v^T || v_i) \leq \theta.
\]

Here, vector \( \mathbb{I} \) contains the public possible class labels, e.g., [distressed, upset, guilty, scared] in Section IV. \( v^T \) is the true posterior probability for \( \mathbb{I} \). \( D_{KL}(||) \) is the Kullback-Leibler (KL) divergence function. \( D_{KL}(v^T || v_i) \) measures the information lost using \( v_i \) to approximate \( v^T \). Clearly, condition (i) can be guaranteed by leveraging TEEs. Next, we design a pricing mechanism to meet condition (ii), that is, every provider has no motivation to provide a prediction which deviates from the truthful labels and the corresponding truthful posterior probability. However, \( v^T \) is unknown in our setting. Our designed pricing mechanism will take it into consideration.

**Definition 2. (Individual Rationality)** A pricing mechanism satisfies individual rationality (IR) if every participating provider \( i \in P \) at the BNE has non-negative expected utility:

\[
\mathbb{E}_{S_{-s_i}}[u_i(s_i^*, S_{-s_i}^*)] \geq 0.
\]

**Definition 3. (Budget Feasibility)** A pricing mechanism satisfies budget feasibility (BF) if the total payment allocated to the participating providers \( i \in P \) at the BNE is not more than a user’s given budget for every query:

\[
\mathbb{E}_S[\sum_{i=1}^m p_i(S^*\})] \leq \frac{B}{n},
\]

where \( m \) is the number of providers while \( n \) is the number of queries.

**C. Pricing Mechanism Formulation**

We are now ready to formulate the optimization problem of designing our pricing mechanism for participants’ predictions (called as PPP), i.e.,

\[
\max_{p(\cdot)} \sum_{i=1}^m \Pr(D_{KL}(v^T || v_i) \leq \theta)
\]

s.t. \( \mathbb{E}_{S_{-s_i}}[u_i(s_i^*, S_{-s_i}^*)] \geq 0 \)

\[
\mathbb{E}_S[\sum_{i=1}^m p_i(S^*)] \leq \frac{B}{n}.
\]

As elaborated, given a set of participating providers \( P = \{1, \ldots, m\}, n \) queries and budget \( B \), we aim to customize a payment function \( p(\cdot) \) which satisfies both constraints of IR and BF, as well as maximizes the objective function, that is, the overall probability of the KL divergence between every provider’s prediction at BNE \( S^* \) and the true prediction which is less than given threshold \( \theta \).

Solving PPP optimization problem will effectively minimize the loss between the accuracy of the aggregated predictions via truth discovery and the truth accuracy, which is the user’s objective. First, given \( n \) queries, \( \sum_{j=1}^n \sum_{i=1}^m D_{KL}(v^T_j || v_i^j) \) is apparently minimized, if PPP optimization problem is solved for every query. Next, we can achieve that the result accuracy via truth discovery is as close as possible to the truth accuracy due to \( \sum_{j=1}^n \sum_{i=1}^m \Pr(D_{KL}(v^T_j || v_i^j) \leq \theta) \geq \sum_{j=1}^n \Pr(D_{KL}(v^T_j || v_i^j) \leq \theta) \). The conclusion is according to the following derivation:

\[
\sum_{j=1}^n \sum_{i=1}^m D_{KL}(v^T_j || v_i^j) \geq \sum_{j=1}^n \sum_{i=1}^m w_i \frac{\Pr(D_{KL}(v^T_j || v_i^j))}{\sum_{j=1}^n w_i}
\]

\[
= \sum_{j=1}^n \sum_{i=1}^m w_i \frac{\sum_{j=1}^m D_{KL}(v^T_j || v_i^j)}{\sum_{j=1}^m w_i}
\]

\[
= \sum_{j=1}^n D_{KL}(v^T_j || v_i^j) \|
\]

However, solving PPP optimization problem is hard and the ground truth is unavailable, namely \( v^T \). Hence, we approximately solve it by applying the idea of divergence-based Bayesian Truth Serum (BTS) method [16]. The main idea of the divergence-based BTS method is rewarding a player based on the divergence between her reports and a randomly selected counterpart’ reports, when there is no ground truth.
for verification. It is an effective approach to incentivize report truthfulness and control report quality [16]. We inherit such desirable properties from the divergence-based BTS method, and in the meantime, we handle both discrete data and continuous data, i.e., label and posterior probability, which is different from prior works [9] [11] considering either continuous data or discrete data.

Derived from the divergence-based BTS method, we denote our payment function. It rewards a participating provider \( i \) based on its strategy \( s_i \) and a randomly selected provider \( r \)'s \( s_r \) by calculating two scores. The payment function is \( p_i(s_i) = \alpha_i \times (\text{score}_{i1} + \text{score}_{ip1} + 1)^2 \), where \( \alpha_i > 0 \). The two scores are denoted accordingly as following:

1) \( \text{score}_{i1} = \text{score}(s_i, s_r) \) measures a penalty value if \( s_i \) reports the same labels with \( s_r \), but the corresponding posterior probability disagrees with each others.

\[
\text{score}_{i1}(s_i, s_r) = -\mathbb{I}_{l_i = l_r} \Delta D_{KL}(v_i || v_r) > 0
\]

Herein, \( \mathbb{I} \) is an indicator. Its value is 1, if condition \( \alpha \) is valid; otherwise, its value is 0.

2) \( \text{score}_{ip1} = \text{score}(s_i, s_r) \) measures a reward value if \( s_i \)'s posterior probability fits close to the distribution of the class labels provided by \( s_r \).

\[
\text{score}_{ip1}(s_i, s_r) = \frac{1}{c} \sum_{k=1}^{c} [2 - (1 - \text{Pr}(l_{rk} | l_{ik}))^2 - \sum_{l_r \in \Omega(l_{rk})} \text{Pr}(l_r | l_{ik})]
\]

Herein, \( c \) is the number of possible class labels; \( \text{Pr}(l_r) \) means the posterior probability for label \( l \) and \( \{\text{Pr}(l_{r1}), ..., \text{Pr}(l_{rk})\} \) constitute \( v_i \) with constraint \( \sum_{l_{rk} \in \Omega} \text{Pr}(l_{rk}) = 1 \). Concretely, \( \text{Pr}(l_{rk} | l_{ik}) \) represents \( i \)'s the posterior probability for label \( l_{rk} \in \Omega \). If \( \text{Pr}(l_{rk}) = \text{Pr}(l_{rk} | l_{ik}) = l_{rk} \), the score value is maximized, being equal to 0. If \( \text{Pr}(l_{rk} | l_{ik}) \neq l_{rk} \), the score value is minimized, being equal to 0.

With the definitions above, the value of \( (\text{score}_{i1} + \text{score}_{ip1} + 1) \) falls in the range \([0, 3]\). Also, it is worth noting that the scoring rule consisting of the two scores has been proved strictly Bayes-Nash incentive-compatible relying on stochastic relevance in [16]. It means that a truthful prediction is always configured with a higher score compared to a untruthful prediction so as to achieve the goal of fairness (refer it to Section III-C).

Considering the potentiality of a participating provider aborting, we revise our payment function. If provider \( i \) does not abort, her deposit \( d_0 \) should be refunded, that is, \( p_i(s_i) = p_i(s_i) = p_i(s_i) + d_0 \). Otherwise, her deposit \( d_0 \) will be forfeited.

**D. Analysis**

In this section, we proceed to analyze how to achieve the design objectives in Section IV-B by using the presented pricing function as an approximately solution.

To begin with, we quantify the cost function with respect to different participating providers, which is useful to estimate the providers’ expected utility. For simplicity, we assume that participating providers’ costs are known, which refers to the complete information scenario. Their costs derive from the identical two cost parameters \( c_1 > 0 \) and \( c_2 > 0 \) which are far smaller than a user’s budget \( B \). We assume that the cost of generating a product linearly increases with the product’s quality. Recall that we measure the truthfulness of a prediction via the divergence-based BTS method due to the lack of ground truths. Specifically, using two scores measures a prediction truthfulness. Thus, we next naturally regard the two scores as the quality metric to calculate the corresponding cost of every strategy \( s_i \). That is, \( c(s_i) = c_1 (\text{score}_{i1} + \text{score}_{ip1} + 1) + c_2 \). It is noteworthy that the cost monotonically increases with score \( (\text{score}_{i1} + \text{score}_{ip1} + 1) \) increasing.

We are now ready to analyze that with our pricing mechanism, there exists a BNE achieving our design objectives via parameter constraints. Specifically, we set constraint conditions on parameter \( \alpha \), considering the design objectives of individual rationality and budget feasibility, based on which we find a BNE, where all participants adopt the strategy of offering truthful predictions. Below, we demonstrate and prove this finding by Theorem 1.

**Theorem 1.** In the non-cooperative game, there exists a BNE \( S^* = (s^*_1, ..., s^*_m) \), where every participating provider \( i \in \{1, ..., m\} \) provides \( s^*_i \) containing \( l_i = l_r \) and \( D_{KL}(v_i || v_r) \leq \theta \) compared with \( s^*_r (r \neq i) \) when parameter \( \alpha_i \) satisfies (1) \( \alpha_i \geq \frac{c_1}{2 \cdot \text{score}_{i1}^*} \), (2) \( \alpha_i \geq \frac{c_1 \cdot \text{score}_{ip1}^* + c_2}{\text{score}_{ip1}^*} \) and (3) \( \alpha_i \leq \frac{B}{nm} \cdot \frac{1}{\text{score}_{i1}^*} \), where \( \text{score}_{i1}^* = (\text{score}_{i1} + \text{score}_{ip1} + 1) \).

**Proof.** Given other participating providers’ strategies \( S^* - s_i \) and a randomly selected provider’s strategy \( s^*_r \), every provider \( i \) can estimate her expected utility by

\[
\mathbb{E}_{S_{-s_i}} [u_i(s_i, S_{-s_i}) | s^*_r] = \mathbb{E}_{S_{-s_i}} [p_i(s_i, S_{-s_i}) | s^*_r] - c(s_i) - d_0 = \alpha_i \times (\text{score}_{i1} + \text{score}_{ip1} + 1)^2 + d_0 - c_1 \cdot (\text{score}_{i1} + \text{score}_{ip1} + 1) - c_2 - d_0.
\]

Here, we suppose that provider \( i \) does not abort. If she aborts, apparently her expected utility is equal to \(-d_0\) which is negative. For every provider \( i \) not aborting, she can maximize her expected utility when her strategy \( s^*_i = (l_i, v_i) \) leads to \( \text{score}_{i1}^* \) reaching the maximum among \([\frac{c_1}{2 \cdot \alpha_i}, 3]\). Therefore, every rational provider \( i \) is doomed to choose the strategy which enables \( \text{score}_{i1}^* \) being equal to 3. To be more clear, we summarize the possible cases for every provider \( i \)'s strategy and her expected utility as following:

(a). If \( s^*_i = (l_i, v_i) \), where \( l_i = l_r \), her expected utility is negative due to \( (\text{score}_{i1} + \text{score}_{ip1} + 1) = 0 \) leading to \( \mathbb{E}_{S_{-s_i}} [u_i(s_i, S_{-s_i}) | s^*_r] = -c_2 \).

(b). If \( s^*_i = (l_i, v_i) \), where \( l_i = l_r \), and \( D_{KL}(v_i || v_r) \leq \theta \), her expected utility is equal to \( 9c_1 - 3c_1 - c_2 \) which is positive due to parameter constraint (2), and maximized due to \( (\text{score}_{i1} + \text{score}_{ip1} + 1) = 3 \).

(c). If abort, her expected utility is negative due to \( \mathbb{E}_{S_{-s_i}} [u_i(s_i, S_{-s_i}) | s^*_r] = -d_0 \).

Hence, strategy profile \( S^* = (s^*_1, ..., s^*_m) \) in Theorem 1 where \( s^*_i \) satisfies \( l_i = l_r \) and \( D_{KL}(v_i || v_r) \leq \theta \) is a BNE. \( \Box \)
VI. EXPERIMENT

A. Implementation and Setup

Prediction Aggregation with TEEs. We initialize TEEs by utilizing SGX SDK of version 2.5. In the SGX environment, we implement the prediction aggregation program (i.e., Algorithm 1) by using C/C++ programming language.

Smart Contract. We also implement the PS contract with the Solidity programming language of Ethereum and deploy it on the Ropsten Test Network via MetaMask².

Dataset. We totally use three datasets to simulate three prediction tasks. Specifically, we use two well-studied image datasets, including MNIST² and ImageNet³, and a public text dataset, namely 20 Newsgroups⁴, for text prediction. With respect to three datasets, we will correspondingly sample a number of test data for evaluation. Note that MNIST, ImageNet and 20 Newsgroups contain 10K, 100K and near 8K test data, respectively. More concrete information of the three datasets are shown in TABLE III.

| Dataset       | Type   | Size       | Features | Labels |
|---------------|--------|------------|----------|--------|
| MNIST         | Image  | 70K        | 20x20    | 10     |
| ImageNet      | Image  | 1.26M      | 224x224x3| 1000   |
| 20 Newsgroups | Text   | 18846      | –        | 20     |

TABLE III: Real-world datasets used in the experiment.

Provider Simulation. We collect three groups of various trained models which are used to simulate providers for serving prediction. We separately collect 6, 10 and 15 models under various frameworks which are evaluated on MNIST, 20 Newsgroups and ImageNet. Specifically, we implement and train the models for the MNIST and 20 Newsgroups by ourselves, and download off-the-shelf models for ImageNet from two public model sources⁵. Due to the space limitation, we only present the models trained on 20 Newsgroups dataset in TABLE IV.

| Model       | Framework | Acc. | Model      | Framework | Acc. |
|-------------|-----------|------|------------|-----------|------|
| Boost       | SKLearn   | 0.740| KNN        | SKLearn   | 0.660|
| Bagging     | SKLearn   | 0.660| CNN        | Keras     | 0.730|
| Dec. tree   | SKLearn   | 0.550| DNN        | Keras     | 0.810|
| Ran. forest | SKLearn   | 0.760| RNN        | Keras     | 0.760|
| SVM         | SKLearn   | 0.820| RCNN       | Keras     | 0.720|

TABLE IV: Models evaluated on 20 Newsgroups.

We simulate distrustful predictions by perturbing normal predictions, where perturbations are sampled from the uniform distribution on interval (0, 1). With the perturbation methods, we simulate a distrustful provider by perturbing a model’s all predictions.

We will consider three cases in perturbing predictions of models, including (a) no perturbation, (b) perturbing no more than $\frac{M}{3}$ models’ predictions, and (c) perturbing more than $\frac{M}{3}$ models’ predictions, where $M$ is the total number of models. Note that case (a) is used to simulate the BNE setting induced by Theorem 1 where each provider is incentivized to provide truthful predictions; case (c) creates the setting, where providers lack sufficient motivation for prediction truthfulness; case (b) refers to the setting between case (a) and (c).

In addition, our experiments are conducted in a Ubuntu 16.04 server equipped with a CPU of 3.40GHz, 32 GB RAM and a GPU of Nvidia GTX-1080.

B. Evaluation

Our evaluation is four-fold: (i) To highlight the advantage of Algorithm 1, we compare the accuracy of predictions generated by Algorithm 1 and that by averaging (a traditional ensemble strategy); (ii) To demonstrate the effectiveness of the incentive mechanism, we plot and compare simulation results of prediction aggregation regarding case (a), (b) and (c) in terms of accuracy; (iii) To show service cost, we estimate the computation complexity of prediction aggregation with a TEE and evaluate gas costs caused by the interaction between the PS contract and the TEE; (iv) To answer whether or not prediction aggregation via Algorithm 1 is effective to resist membership inference attacks, we conduct state-of-the-art attacks ⁶ and present empirical evidences.

| Dataset  | Avg. Label Rank Probability | Rank | Probability |
|----------|----------------------------|------|-------------|
| MNIST    | 0.907 0.978 0.973 0.981    | 0    | 0.981       |
| ImageNet | 0.724 0.790 0.764 0.789    | 2    | 0.789       |
| 20 Newsgroups | 0.721 0.862 0.836 0.862 | 2    | 0.862       |

First of all, as shown in TABLE V, for each dataset, the accuracy of the predictions generated by Algorithm 1 regarding three output formats (i.e., $3_{th}$ to $5_{th}$ column) is always better than the averaging accuracy (i.e., $2_{th}$ column) of all participating models. We can see that on ImageNet dataset, the accuracy improvement is relatively small, but as pointed out by [44], spending a lot of time and energy to achieve minor accuracy improvement on difficult object recognition task is deserved.

Second, Fig. 2 and Fig. 3 (from left to right) show the accuracy of aggregated predictions regarding three perturbation cases on MNIST, 20 Newsgroups and ImageNet, respectively. For each dataset, it can be clearly seen that the accuracy in case (a) is always higher than that in case (b) and (c), which is because that participating providers offer truthful predictions with sufficient incentives. We also can see that in case (c), where a vast majority of participating providers report meaningless predictions, the accuracy is never better than 0.5. The reason is that Algorithm 1 fails to learn the truth when a majority of predictions are not enough accurate, and thus our incentive mechanism is necessary to handle case (c). In addition, from Fig. 2, the evaluated accuracy slightly grows up with the increasing queries. According to Fig. 3, we also notice that the accuracy of the rank-level predictions on ImageNet drops more obviously than the other two datasets.

²https://metamask.io/
³http://yann.lecun.com/exdb/mnist/
⁴http://www.image-net.org/challenges/LSVRC/2012/
⁵http://www.cs.unc.edu/~jason/20Newsgroups/
⁶https://keras.io/api/applications/
⁷https://pytorch.org/docs/stable/torchvision/models.html
in more serious perturbation cases. It might be caused by the large number of labels, i.e., 1000, on ImageNet dataset.

Third, Fig. 4 presents the estimated time costs of prediction aggregation inside the TEE over three datasets. Note that we omit the one-time cost of setting up a TEE. Clearly, more queries spend more times. By comparing the three sub-figures, we also can know that the time complexity becomes higher as the number of labels of the query task increases. Recall that the number of class labels of MNIST, 20 Newsgroups and ImageNet is 10, 20 and 1000, respectively. Besides, gas costs are mainly derived from two parts: (1) execution costs of the PS contract when its three entry points, Deposit, Request and Response, are correspondingly invoked, and (2) execution costs of the TEE’s transaction (on entry point Response) which contains outputs outp, outp_attr and signatures σ, σ_attr (entirely $2 \times 70$ bytes). Also, the gas costs grow up with increasing participating providers. Note that encrypted input data and predictions are transmitted off-chain, and thus the magnitude of query makes negligible effect on the gas costs. We only test the gas costs by simulating 6 providers (on MNIST). Specifically, part (1) totally spends 510, 815 units gas, including 389, 373 units for Deposit, 102, 200 units for Request and 19, 242 units for Response. The gas costs for sending the response transaction in part (2) are about 74, 370 units.

TABLE VI: Comparison of adversaries’ attack performance.

| Type        | Target model | Precision | Recall |
|-------------|--------------|-----------|--------|
| Adversary 1 | Single       | 0.996     | 0.503  |
|             | Ensemble     | 0.056     | 0.054  |
| Adversary 2 | Single       | 0.997     | 0.504  |
|             | Ensemble     | 0.987     | 0.499  |

Last, we launch membership inference attacks using two types of adversaries with increasingly strong attack capabilities in prior work (i.e., adversary 1 and 2, detailed in [43]’s TABLE I) and show the attack results. Similar to the work [43], we adopt three models as an ensemble, but the difference is that our ensemble strategy is Algorithm 1 rather than stacking. Besides, the used three models are CNN, RNN and MLP trained on the MNIST dataset. For comparison, we also conduct the same attacks on the single CNN model. As shown in TABLE VI, the attack results demonstrate that ensemble model under Algorithm 1 is able to reduce the attack performance of adversary 1, but not adversary 2. Concretely, for adversary 1, the precision drops from 0.996 to 0.056 and the recall drops from 0.503 to 0.054. But for adversary 2, there has no effect. It is difficult to suggest certain confident explanation for the attack results like the previous work [43].

VII. LIMITATION AND FUTURE WORK

Other prediction formats. Our work focuses on the prediction formats, including top-1 label, ranked labels and posterior probability, but fails to support other formats, such as text data, in Natural language processing (NLP) tasks. Taking language translation as an example, Sequence-to-Sequence models are usually used, which take as input a sequence of words in certain language and output another sequence of words in a target language, where output format belongs to text data.

Adversarial examples. We assume benign input data and do not consider adversarial examples (AEs), i.e., input data injected with imperceptible perturbations [35]. AEs can mislead a deep neural network to incorrectly classify an originally correctly classified input. Recently, a promising approach against AEs is to create a robust ensemble model by carefully considering the diversity of individual models [45, 46]. In our future work, we will follow this direction and take into account the factors regarding model diversity to refine our incentive mechanism for FedServing.

VIII. CONCLUSION

In this paper, we present a prediction serving framework, named as FedServing, towards trained models from various sources. FedServing enables locally deploying models and provides collective prediction services for charging users. For motivating truthful predictions, we customize an incentive mechanism based on Bayesian game theory. For boosting prediction accuracy, we use truth discovery algorithms working jointly with the incentive mechanism to eliminate the effect of low-accuracy predictions. Our proposed design supports popular prediction formats, including top-1 label, ranked labels and posterior probability. Besides, we build FedServing on the blockchain to ensure exchange fairness and leverage TEEs to securely aggregate predictions as well. With extensive experiments, we effectively validate the expected properties of our mechanism and empirically demonstrate its capability of reducing the risk of certain membership inference attack.
