A Protocol for a Market of Machine Learning Models

Claudiu Marius Popescu (✉ Claudiu.Popescu@bel.utcluj.ro)

Technical University of Cluj-Napoca: Universitatea Tehnica din Cluj-Napoca <https://orcid.org/0000-0001-9131-2752>

Research Article

Keywords: machine learning, asymmetric

Posted Date: November 11th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1030456/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

A protocol for a market of machine learning models

Claudiu Marius Popescu*

October 2021

Abstract

This paper describes a protocol for a market of machine learning models. The economic interaction involves two types of agents: data providers- agents that have some data and want to use it to get a predictive model, and model providers- agents able to use the data to generate predictive models. First, we will show that the process is informationally asymmetric, therefore a standard direct market can not function. Then, we design a protocol with the aim of creating a viable and efficient market mechanism for these particular services, under the specific challenges of information asymmetries. The protocol is theoretically analysed, to establish it's correctness and computational complexity. We also propose a simple reference implementation based on a HTTP API. The implementation is then used in a few case studies, and analysed empirically.

1 Introduction

In the last three decades, machine learning has become, from a specialised research area, a common technology with impact on almost all aspects of modern society (Prashant Johri, 2020; Jordan & Mitchell, 2015; Shalev-Shwartz & Ben-David, 2014). This paradigm shift has driven an increased interest at the border of technological and economical aspects of this field. In this paper we deal with a problem arising exactly at this frontier. More concurrently, we develop a protocol which can facilitate the interaction between those that need machine learning-based solutions, and those that can provide them.

As the datasets become larger, more complex (e.g. multimedia or genomics data) or "noisy" (missing or corrupted features, many irrelevant

*Technical University of Cluj-Napoca, George Barițiu 26-28, Cluj-Napoca 400027, Romania, Claudiu.Popescu@bel.utcluj.ro

attributes, etc.), building predictive models requires a large amount of computational resources, and big development teams, with a mix of skills ranging from statistics to software engineering, and from databases creation to infrastructure administration (Lavalley, Lesser, Shockley, Hopkins, & Kruschwitz, 2011) (Amershi, Begel, Bird, DeLine, Gall, Kamar, Nagappan, Nushi, & Zimmermann, 2019). This evolution starts to put a lot of pressure for specialisation and division of labor in this field. This is the main reason why an entity (organization or person) that has data and wants to use it to solve a specific problem using some type of predictive modeling (classification, regression, ranking, etc.) may not be able, or may not find it efficiently to engage in the process of developing such a model.

The solution to the problem described in the previous paragraph can be solved by a market composed of two types of agents:

- First, the agents that have the data and are interested in leveraging it to solve a significant problem. This process usually require developing a predictive model, as discussed above. We will call this agents data providers;
- Second, the agents that have the skills and technology to construct predictive models. In the rest of the paper we will call them model providers. In a typical scenario, they will use the data to train a (large scale) machine learning model (e.g a Deep Neural Network with a specific architecture (Goodfellow, Bengio, & Courville, 2016)).

The interaction between the two types of agents is, at a first look, straight forward: the data provider formulates a learning problem related to some dataset and, in exchange for some price, the model provider will generate a predictive model according to the specifications. As it is typical in machine learning, the model provider will receive a training dataset, and the performance of the model can be evaluated by computing the empirical risk (error) on a test dataset (Shalev-Shwartz & Ben-David, 2014).

However, we will show that the market, in it's basic form, display a type of information asymmetry (Hillier, 1997). At a high level, the information asymmetry comes from the fact that the model provider knows more about how the model was generated than the data provider, while the data provider knows more than it's counterpart about how the two datasets were produced. This problem makes a direct market unsuitable.

In order to alleviate the information asymmetry problem, we propose a lightweight protocol that will, under some basic assumptions, determine the two types of agents to behave such that both of them will have a positive

outcome. The protocol consist of 13 steps, each one requiring a simple computation and/or sending some data to the other party. Our analysis shows that rational and self interested agents will have no possibility or no incentives to cheat during the protocol execution.

We provide a simple implementation of the protocol as a HTTP API (Fielding & Taylor, 2000). Using this implementation, we study the computational and communication overhead induced by the protocol. The experiments are done with a few public datasets and learning algorithms that we believe illustrates the behaviour of the protocol on typical and extreme cases. The empirical results agree with the theoretical observation that the overall complexity of the protocol is very low.

2 Related work

The shortcomings of the current approaches to machine learning models development are well known and some interesting solutions have been proposed for some of them. In this section we will try to review those that are addressing related problems, sub-problems or issues similar to those that we tackle in the current paper.

In (Blum & Hardt, 2015), the authors describe a simple procedure for ranking machine learning models during a competition. The naive approach, based on just evaluating the test error and rank according to it, is prone to bias, because the competitors can select the models based on the feedback they receive from the ranking algorithm, improving artificially their position. The idea of the proposed algorithm is very simple. For each given model, it compares the empirical error estimate of the model to the previously smallest error. If the estimate is below the previous best by some margin, it updates the best estimate and publish it. If the estimate is not smaller by a margin, the algorithm releases the previous best error.

In addition, the paper also shows that if the scores are in the interval $[0, 1]$, the maximum error of the algorithm is never worse than $O((\log(k)/n)^{1/3})$, where k is the number of submissions and n is the size of the dataset used to compute the error. This is an important improvement over the naive solution (which has an upper bound of $O(\sqrt{k})$), and it is almost optimal.

In the last couple of years, the crowdsourcing contests become very popular and an active area of research (Segev, 2020). In such a contest, a requester posts a task (e.g. programming task, design task, etc.) on a platform, together with a monetary reward that he or she is willing to pay for the winning solution. Contestants send solutions on the platform and the

requester chooses the best solution (or solutions) and awards the prize. The usual tasks discussed in the literature don't have the information asymmetry problem that arises in case of machine learning tasks. For example, if the task is to solve a hard combinatorial optimization problem, the solution will be a program in some high level programming language, that can be reviewed. The program can be also unambiguously tested on any instance of the problem that it claims to solve.

Much of the research on crowdsourcing contests is orthogonal to the current paper, focusing on equilibrium, efficiency and users behaviour. In (Chawla, Hartline, & Sivan, 2019), a theory for optimal crowdsourcing contests was introduced. The study focuses on the optimal crowdsourcing contests over all single-stage all-pay formats, when the goal is to optimize the quality of the best submission. The main conclusion is that the wasted effort can be relatively small for optimal crowdsourcing contests if the award can be divided between agents dynamically depending on the qualities of submissions.

In (Cavallo & Jain, 2012), an algorithm and incentive mechanism that achieves equilibrium implementation of the socially optimal policy for an important class of crowdsourcing contests was proposed. In the analysed contests, the principal seeks production of a good within a limited time interval (after the deadline, any good procured has no value). Other works deal with similar issues, for different types of contests (e.g. in (Archak & Sundararajan, 2009), the focus is on the asymptotic behavior of the contest outcome, and providing simple rules for the best division of the contest budget among the participants).

A complementary task to the one that we focus on, which can be restated as custom predictive models trading, is data trading. In a recent paper, (Zhao, Yu, Li, Han, & Du, 2019), a protocol for this activity is described. More concretely, the authors designed and evaluated a new blockchain-based fair data trading protocol for big data market. The protocol ensure data availability for the data consumer, privacy of the data provider and payment fairness. A related area is that of optima data procurement for learning. For example, in (Abernethy, Chen, Ho, & Waggoner, 2015), past data is used to actively price future data in order to obtain learning guarantees, when the cost can depend arbitrarily on the data itself. The authors propose a method to convert a large class of no-regret algorithms into online posted-price and learning mechanisms. Some related topics are discussed in a large number of papers (see, for example, (Chen, Immorlica, Lucier, Syrgkanis, & Ziani, 2018; Chen & Zheng, 2019; Gast, Ioannidis, Loiseau, & Roussillon, 2020; Zhang, Arafa, Wei, & Berry, 2021)).

The protocol proposed in this paper is in part close in spirit to the (interactive) zero knowledge proofs (Goldreich, Micali, & Wigderson, 1991) (Goldwasser, Micali, & Rackoff, 1985). In that setting, important in cryptography, one party (the prover) must convince the other party (the verifier) that he/she has a valid proof for a statement without revealing the proof. The notions of "proof" and "statement" are very general, not merely the mathematical ones. As in our case, a zero knowledge proof involves an exchange of data between two parties. One important difference is that in our case both both parties must show to the other side that he/she really has some information that he/she claims to have (a good predictor, and *i.i.d.* samples, respectively). Another fundamental particularity of our setting is the uncertainty inherent to predictive modeling problems, and all the other particularities of machine learning setting discussed in this paper (in particular in Section 5.2).

As observed in the Introduction, the problem that we try to solve in this paper is generated by a form of information asymmetry. Starting with the work of George Akerlof (Akerlof, 1970), the subject received a lot of attention (Hillier, 1997; Akerlof, 1970; Spence, 1973; Rothschild & Stiglitz, 1976; Lambert, Leuz, & Verrecchia, 2011; Hoppe & Schmitz, 2013).

A distinct, but related topic is that of testing machine learning based software products. The subject received a lot of attention over the last couple of years. A recent and comprehensive survey is (Zhang, Harman, Ma, & Liu, 2020).

We end this section by mentioning the only line of work, to the best of our knowledge, that deals with the topic of information asymmetry in the context of artificial intelligence. In (Marwala & Hurwitz, 2015) and (Marwala & Hurwitz, 2017) the impact of artificial intelligence on the theory of asymmetric information is studied. The most important conclusion of the authors is that with the arrival of artificial intelligence methods and agents, signaling and screening in the markets are easier to achieve. The results of the performed simulations demonstrate that artificial intelligence agents reduce the degree of information asymmetry and, therefore, the markets where these agents are used are more efficient.

3 Double information asymmetry and Arrow information paradox

In it's basic form, the market for predictive models consists of two types of agents, data providers and model providers. The data providers are the

entities that formulate the learning task and offer the data. Model providers are the entities capable of creating solutions for the learning task. On the market, each data provider will present a learning task and ask for a solution. The model providers can analyse the existing learning tasks, and offer a solution in exchange for some price. In principle, the data providers will try to get a good solution for a low price, while the model providers will try to get clients, and a price as high as possible. A transaction will consist of an exchange of a predictive model for a certain amount of money.

Now we will outline the main reason why such a market can not be implemented in a straightforward way. For this, we rely on the theory of information asymmetry. The information asymmetry is one of the most common and well researched market failure causes (Hillier, 1997; Akerlof, 1970). Information asymmetry is, simply put, the difference in knowledge between a seller and a buyer regarding a product. In the most common scenario, the seller knows more about the product that is being sold than the buyer and he/she can use this information in his advantage, causing the market to be inefficient or, in some extreme cases, make it impossible to function. The phenomenon was observed in many types of markets, two important classic examples being the job market (Spence, 1973) and the insurance market (Rothschild & Stiglitz, 1976).

One key observation on which the paper rest is what can be called inherent double information asymmetry of the particular economic interaction in which we are interested. What this means is that both the data provider and the model provider have some private information. The first agent knows about the origin and the structure of the dataset (are the instances independent and identically distributed? etc.), and these aspects influence greatly the process of producing the machine learning model and the end result. The second agent in its turn has valuable private information about the predictive model (was it produced by overfitting? What is the estimated generalisation error? etc.).

This information can, in some circumstances, be accessed by the other party, but in general it is private information. These aspects make the interaction between the two parties less efficient or impossible when we make the standard assumptions that the agents are rational and self-interested.

The proposed protocol offers also a solution for a specific instance of Arrow information paradox (Arrow, 1962). The paradox consists of the dilemma that the buyer and the supplier faces when they need to exchange a product based on intellectual property. The buyer require to know the technology behind the product so he/she can trust it, but the supplier on the other hand does not want to reveal this information since it has intrinsic

economic value. The usual approach to mitigate this problem and make the market work is to enforce a contract between the two parties by which the buyer obliged to not use the additional information or the seller guarantee that the product has certain qualities. Another common solution is the use of patents.

In our case, the data provider wants to know that the model has an accuracy above a certain threshold. On the other hand, for the model provider it might be undesirable to reveal all the details of how the model was generated. The protocol that we describe is an alternative to the contract and patent solutions for this conundrum. While the classical contract solution may be an alternative, the uncertainty inherent to machine learning-based products makes the problem of designing such a contract non-trivial.

The concrete manifestation of the information asymmetry phenomenon, together with a detailed and quantitative analysis, will be shown in Section 5.2.

4 Protocol design

Now we will present the protocol in detail. It's aim is quite general, but for simplicity the reader can keep in mind the simplest prediction problem: binary classification. The protocol can, however, directly or with minimal adjustments, be applied to a wide range of machine learning problems: multi-class classification, regression, ranking, structured prediction, etc.

The protocol prescribe the interaction between the two types of agents, data providers and model providers, as a sequence of steps. Synthetic presentations of the protocol are provided in Table 1 and Figure 1.

The starting point is when the data provider chooses an ordering of the data instances. In Section 5.3.1, we will show that the order does not have in fact any impact on the final outcome.

In the next step, the hash of the dataset is computed. This can be achieved using any cryptographically secure one way function. For concreteness, we choose the popular option SHA-2 (Lilly, 2004) for all hashes used in the protocol.

The step 3 consists of selecting k (pseudo-)random numbers (Johnston, 2018) from the set $\{1, 2, \dots, n\}$, and sending them to the data provider. The operation can be achieved using a public random numbers generator¹. The critical aspect here is that the process is not under the control of the data provider.

¹For example, <https://www.random.org/> (Accessed: 2021-03-12)

1.	The data provider chooses an ordering of the dataset.
2.	The data provider computes a hash of the ordered dataset.
3.	Using a (public) random numbers generator, k random distinct numbers from the set $\{1, 2, 3, \dots, n\}$ are extracted, and sent to data provider.
4.	The data provider publishes the k instances selected on step 3 (this is the training set), the computed hash, the minimum expected accuracy, the base price and the price factor.
5.	The data provider starts an auction for a predictor. The winner is the agent which required the lowest price (we will call it the model provider).
6.	The model provider uses the dataset to train a model, and outputs a predictor P .
7.	The model provider computes the hash of P , and sends it to the data provider.
8.	The data provider sends to the model provider the test sample without the labels.
9.	The model provider computes the labels of the received datasets, and sends them to the data provider which will evaluate the error. If the error is below the threshold, the process continues.
10.	The data provider sends to the model provider the labels of the test dataset in the correct order, and the computed error.
11.	The model provider reconstructs the ordered dataset, computes its hash, and compare it with the one received from the data provider. If they agree, the process continues.
12.	The model provider computes the test error, and compare it with the value received in step 10. Then, it sends the model to the data provider.
13.	The data provider computes the test error and the hash of the predictor P , and compare them with the ones obtained in the previous steps. If everything agrees, the transaction finishes with the data provider paying to the model provider the price p .

Table 1: **The description of the proposed protocol**

After receiving the k numbers, the data provider is ready to publish the required data (this will be the step 4). The data provider will publish the data points having the k indices it just received (this will be the training dataset), together with the hash computed in step 2, the minimum expected accuracy (the minimum acceptable accuracy of the model), the base price, and the price factor. These concepts will be explained in the next paragraph.

Pricing a machine learning model seems to be in itself and interesting

Figure 1: The protocol diagram.

and important question, but we will not deal with it in the current paper, limiting our scope to a basic setting. We will assume that the data provider will fix a base price for the auction (see the description of step 5), p_b , the

maximum price that it is willing to pay for a model having the minimum test accuracy (a_{T_m}), and a price factor, α . If the model provider is able to produce a model with a test accuracy (a_T) at least equal with the minimum expected accuracy, the final price that the data provider is required to pay is:

$$p = p_m + \alpha(a_T - a_{T_m}), \quad (1)$$

where p_m is the price if the model has exactly the minimum accuracy on the test sample ($p_m \leq p_b$).

An important part of the protocol is a type of auction, and this is the concern of step 5. To be more explicit, we can consider that this is a first price sealed bid auction, but any other type of auction can be used (e.g. it can be an incentive compatible one, like the standard second bid auction). In this phase the model providers will bid for the service of creating a machine learning model. The winner- the agent that offers the lowest price- will use the available data to output an eligible model. This agent will be called, from this point, the model provider. Of course, if no agent offer a price above the threshold (base price, p_b), the process ends.

The next step, 6, encapsulates the "useful" computation: the model provider uses the available dataset to train a model, and outputs a predictor P. Then, the model provider will compute the hash of the predictor P, and sends it to the data provider (this is step 7).

In step 8, the data provider sends to the model provider the test sample, in the initial order, but without the labels. Having this data, the model provider can use the predictor to generate some labels for the test data points. The predicted labels will be sent to the data provider, which can evaluate the accuracy. If it is above the threshold (a_{T_m}), the process continue, otherwise it will stop without the transaction taking place (the process can, of course, be repeated, with the exclusion of the current model provider). These computations correspond to the step 9 in the protocol's flow.

After this, the data provider sends to the model provider the labels of the test dataset, in the correct order, and the computed accuracy (this is step 10). Now, in step 11, the model provider is able to reconstruct the initial (ordered) dataset, to compute it's hash, and to compare it with the one published by the data provider in step 4. If they are identical, the process continue. If the hashes are not the same, then the data provider very likely changed the test dataset, and the process stops without the transaction taking place.

The model provider can also compute now the accuracy of the predictor on the test dataset- it should be equal with the value received in step 10- then it sends the predictor to data provider (this is done in step 12).

Finally, in step 13, the data provider uses the predictor P to compute again the test error. It also computes the hash of the predictor and compare it with the one received in step 7. If the values are different, it means that the model provider sent a predictor different from the one used to generate the labels, and the process finishes without a transaction. If everything agrees, the transaction finishes with the data provider paying to the model provider the price p .

We end this section with a few remarks. The first one is that the split of the protocol into steps is, at least partially, arbitrary. It is possible to merge or split the steps in different ways. However, we have tried to keep the number of steps at a minimum, while also ensuring that they are simple and easy to understand.

We also observe that, as it is, the protocol is not designed for sensitive data. But it can be extended to cover such cases as well, using for instance different encryption techniques (Menezes, van Oorschot, & Vanstone, 2001).

5 Theoretical analysis

We are now ready to provide a more detailed analysis of the question why the direct market solution doesn't work, and how can the proposed protocol mitigate the issues. We start by establishing a formal setting, then analyse the naive approaches, and finally, show that the protocol attains it's goal, and has a modest overhead in terms of computation, memory and communication.

5.1 Formal setup

In order to provide a theoretical analysis of problem and the proposed solution, we will now introduce a basic formal setting. The aim of the model and data providers is to solve a machine learning problem. As evident also from the Introduction, we are concerned with supervised learning tasks (although the principles also apply to other types of problems).

For simplicity, we will restrict the formal analysis to the most basic case: binary classification. The following presentation is standard, and the details can be found in any introductory textbook on the subject, e.g. (Shalev-Shwartz & Ben-David, 2014) or (Mohri, Rostamizadeh, & Talwalkar, 2018).

Let X be an arbitrary set, $\mathcal{D}$ a probability distribution over $X \times \{0, 1\}$, with $\mathcal{D}_X$ its marginal over X . We fix a number $n \in \mathbb{N}^*$, and draw independently n points from $X \times \{0, 1\}$, according to $\mathcal{D}$. The result will be an *i.i.d.* (independent and identically distributed) sample S . The sample is randomly split into a training sample $S_{training}$, of size $k \in \mathbb{N}^*$, $k < n$, and a test sample S_{test} . We will denote by $S_{X,training}$ and $S_{X,test}$ the training and test sets, respectively, without the labels ².

In addition, we will also assume that the model will be used on data draw also independently from $\mathcal{D}$. These data points will form another *i.i.d.* sample, $S_{production}$.

Any function $h : X \rightarrow \{0, 1\}$ is called a (binary) hypothesis, and a class of functions $\mathcal{H} = \{h | h : X \rightarrow \{0, 1\}\}$ will be called a (binary) hypothesis class.

For some arbitrary sample $S = \{(x_i, y_i) | i \in [m], x_i \in X, y_i \in \{0, 1\}, m \in \mathbb{N}^*\}$, we can define the empirical risk as:

$$\hat{R}[h] = \sum_{i=1}^m l_{01}(h(x_i), y_i), \quad (2)$$

where $l_{01} : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$ is the 0-1 loss function, defined by $l_{01}(y_1, y_2) = 0$, if $y_1 = y_2$ and $l_{01}(y_1, y_2) = 1$, otherwise. If it is not obvious what is the sample, the notation $\hat{R}_S[h]$ can be used to specify it.

In classification, the ultimate goal is to have a small generalisation error on the specific distribution $\mathcal{D}$. The error for a hypothesis h , also called the true risk, is defined as:

$$R[h] = \mathbf{E}_{(x,y) \sim \mathcal{D}}[l_{01}(h(x), y)]. \quad (3)$$

Now it is feasible to rigorously define a learning problem as the task of finding a hypothesis h with low (ideally minimum) error for some predefined set X , and a fixed but unknown probability distribution $\mathcal{D}$, given access only to a finite *i.i.d.* sample S from $\mathcal{D}$.

Since the error is purely theoretical, and can not be directly computed we must rest on some form of approximation in order to be able to evaluate the quality of a hypothesis. Under some quite general conditions (see for example (Shalev-Shwartz & Ben-David, 2014) or (Mohri et al., 2018)), this can be achieved using the empirical risk computed on a test sample.

The next step is to define the economic incentives of the agents. We take the usual path of assuming that each agent has an utility that it wants to

²When the word "set" is used in reference to a dataset/sample we actually mean a multiset, since an element can appear multiple times; also the notation $S_1 \subset S_2$ for datasets will mean that each element from S_1 is also in S_2

maximize. If the transaction doesn't succeed (the final state in the diagram from Figure 1 is not "Done"), the utility is 0 for both agents. Otherwise, for the data provider, the utility is taken to be proportional to the accuracy of the model, while run in production. If

$$\hat{R}_{S_{production}}[h] = \sum_{i=1}^{m_p} l_{01}(h(x_{pi}), y_{pi}), \quad (4)$$

with $(x_{pi}, y_{pi}) \in S_{production}$ and $m_p = |S_{production}|$, is the empirical risk on the production sample, the data provider's utility can be written as

$$u_d[h] = \begin{cases} \alpha_d(1 - \hat{R}_{S_{production}}[h]), & \text{if the transaction is successful;} \\ 0, & \text{otherwise,} \end{cases} \quad (5)$$

where $\alpha_d > 0$ is some constant.

Notice that we have not taken into account the price paid for the model. This is just a slight simplification, since we can incorporate this dependence in the factor α_d .

For the model provider, when the transaction succeeds, the utility is considered to be proportional to the difference between the price of the model, and the cost to produce it, therefore:

$$u_m[h] = \begin{cases} \alpha_m(p - c), & \text{if the transaction is successful;} \\ 0, & \text{otherwise,} \end{cases} \quad (6)$$

where p is given by Equation 1, the cost c is known to the model provider, and α_m is another constant.

5.2 Why naive solutions fail

The current section is dedicated to a detailed exposition of the consequences of double information asymmetry. We will show why the market can not function if the model provider receives the test data, or if the data provider receive the predictor before it releases the test data. But first, let us introduce two notions that will be useful for the analysis.

Definition 5.1 (m-learning task). A m -learning task is a tuple $(S_{training}, S_{test}, l)$, where $S_{training}$ and S_{test} are two *i.i.d.* datasets, l is a loss function, and the size of the smallest of the two datasets is m .

The definition is just a formalisation of the general idea of a machine learning problem. It will also prove useful to have a notion of an "easy

learning task”, a task for which it is easy to get a model with a low expected loss (low true risk). The idea can be formalised in many ways. Here we choose a simple method: an easy task is one in which the samples are collections of vectors in some Euclidean space $\mathbb{R}^d$ (for some $d \in \mathbb{N}^*$), and a linear model with 0 true risk exists.

Definition 5.2 (Easy learning task). A learning task is called easy if the data points are vectors in an Euclidean space $\mathbb{R}^d$, and a linear model with 0 error exists.

5.2.1 Model provider has access to the whole dataset

In this case the model provider can intentionally overfit to get a low test error (Shalev-Shwartz & Ben-David, 2014). Without restrictions on the hypothesis class, it’s always possible to get 0 error. A simple model that does just that is the one that outputs for each test and training instance the correct label. Using code obfuscation techniques it’s easy to transform this memorization driven model in an apparently sophisticated predictor with the same behaviour on the sample.

In the case of deep learning the situation is complicated by the fact that the model can give very good results even in the interpolation regime (when the training error is 0 or near 0) (Goodfellow et al., 2016). In the interpolating regime, the model can perform very well, although it can give the impression of overfitting. It is difficult for the data provider to check if this is the case, if a separate test sample is not available.

The conditions in which, for deep neural networks, low training error imply low test error are not yet fully understood. An example of puzzling phenomenon is the fact that overparameterized deep neural networks can easily achieve very low training error even on datasets with random labels (Zhang, Bengio, Hardt, Recht, & Vinyals, 2017). All these considerations shows the difficulties experienced by someone trying to evaluate a modern machine learning model, without a test sample.

In this case, the problem is quite obvious, so no further investigations are necessary. In the next section, a more formal, in depth analysis is provided for the more difficult case of access to unlabeled test data.

5.2.2 Model provider has access to the unlabeled test data

A better alternative, used frequently in practice is for the model provider to have access to the test instances, but without the labels. In this case the

model provider can still use this information in his advantage, by incorporating the unlabeled data in the learning process. The co-training algorithm described in (Blum & Mitchell, 1998) is classic. More recent research has brought new approaches and theoretical insights (van Engelen & Hoos, 2020) (Kääriäinen, 2005), (Rigollet, 2007).

The incorporation of unlabeled training data is beneficial for learning, but in our case it gives rise to an issue. If the unlabeled test sample is used by the model provider, the test error may not reflect accurately the generalisation error.

In what follows we will show that in this case it is possible for the model provider to output a model with 0 error on the test set, but with a high generalisation error. To be more precise, we introduce the following definition.

Definition 5.3 ((γ, δ) -model forgeable learning task). A learning task is called (γ, δ) -model forgeable if, when the model provider receive a labeled training sample and an unlabeled test sample, it can efficiently find a hypothesis with the property that it has a true risk with at least γ bigger than the empirical risk computed on the test sample, with probability at least δ .

Here the "efficiency" is understood in the complexity theoretic sense (Arora & Barak, 2009)-the algorithms involved are polynomial in n and d (γ and δ are taken to be constants). The issue will be discussed in more details in Section 5.2.3, where it is more relevant, and the efficiency constraints are non-trivial.

Proposition 5.1. *For any $m \geq 500$, $(\frac{1}{5}, \frac{17}{20})$ -model forgeable m -learning tasks exists.*

Proof. We start by constructing an appropriate learning task. The set X is taken to be an arbitrary finite set, $X = \{x_1, x_2, \dots, x_l\}$, with $l \in \mathbb{N}^*$ to be chosen later. The distribution $\mathcal{D}_X$ is selected to be the uniform distribution, and the size of the test set satisfy the equality $n - k = \frac{l}{2}$. The labeling process is considered to be deterministic: there exists a function $c : X \rightarrow \{0, 1\}$, such that $c(x_i) = y_i, \forall i \in [l]$.

First, we will upper bound the probability that at least one test point is not in the training set, $\mathbf{P}(S_{X,test} \not\subseteq S_{X,training})$. In the worst case, all test points are distinct, therefore we can write:

$$\mathbf{P}(S_{X,test} \not\subseteq S_{X,training}) \leq \mathbf{P}\left(\bigvee_{i=1}^{\frac{l}{2}} (x_i \notin S_{X,training})\right). \quad (7)$$

Because we sample independently from the uniform distribution, it is true that $\mathbf{P}(x_i \notin S_{X,training}) = (1 - \frac{1}{l})^k$, and by union bound and (7):

$$\mathbf{P}(S_{X,test} \not\subset S_{X,training}) \leq \frac{l}{2}(1 - \frac{1}{l})^k. \quad (8)$$

If all points from the unlabeled test set are also in the training set, the model provider can output a hypothesis h that labels correctly all the test points, and for any other point it will choose the label randomly, each option with equal probability. Since the (true) labeling is deterministic, the expected value of the true risk will be $\mathbf{E}[R[h]] = \frac{1}{4}$, where the expectation is with respect to the random labels selected by the model provider. This is true because half of the points ($\frac{l}{2}$) are classified correctly, while for the other half, the label is chosen randomly.

Let us call the random labels $y_1^{(r)}, y_2^{(r)}, \dots, y_{\frac{l}{2}}^{(r)}$, and fix some arbitrary number $\epsilon \in (0, \frac{1}{4})$. By Hoeffding's inequality we have

$$\mathbf{P}(\frac{1}{2} - \frac{2}{l} \sum_{i=1}^{\frac{l}{2}} y_i^{(r)} \geq \epsilon) \leq e^{-2\frac{l}{2}\epsilon^2}. \quad (9)$$

We want to bound the probability $1 - \delta$ that any of the two "bad" events- not having all test points in the training set or deviating from the expected true risk by more than ϵ - will happen. To this end we apply union bound, and using the inequalities (8) and (9) to get:

$$1 - \delta \leq \frac{l}{2}(1 - \frac{1}{l})^k + e^{-l\epsilon^2}. \quad (10)$$

By choosing $k = 10000$, $l = 1000$, $\epsilon = \frac{1}{20}$, we have $1 - \delta \leq \frac{3}{20}$. In this case, with a probability of at least $\delta = \frac{17}{20}$, the empirical risk on the test sample will be 0, while the true risk will be at least $\gamma = \frac{1}{4} - \frac{1}{20} = \frac{1}{5}$, as desired. $\square$

The constants appearing in the proposition are not particularly important. The essential point is that even with large samples, it's still possible to manipulate the model such that with a significant probability, the result is strongly affected. The problem doesn't manifest itself only for samples smaller than some threshold. When the samples grow, the issue persists.

5.2.3 Data provider receive the predictor before it releases the test data

In this case the data provider, trying to minimize it's cost, can forge the test set to reduce the observed accuracy, and hence the price it must pay. It's

easy to see how one can achieve this: on the test set, it can change the label for some of the data points that are correctly classified by the predictor, or it can increase the number instances on which the predictor fails. However, it's not immediately obvious that such a process can remain undetected³. In the following paragraphs we will show that this is indeed the case, at least in some situations.

The main idea is to use a lower bound on the sample complexity of identity testing for probability distributions. Let us assume that some data is sampled independently from a fixed but unknown probability distribution $\mathcal{P}$. The distribution is supported on a set $X \subset \mathbb{R}^d, d \in \mathbb{N}^*$. The following result (Theorem 5.2), which is Theorem 3.4 in (Diakonikolas, Kane, & Peebles, 2019), gives an exponential lower bound on the number of samples needed to test if a distribution belonging to a large class is the uniform distribution ($\mathcal{U}$) or not. The class of distributions is the set of q -histogram distributions, that are distributions $\mathcal{P}$ on $[0, 1]^d$ with the property that there exists a partition of the domain in q axis-aligned rectangles such that $\mathcal{P}$ is constant within each such rectangle.

Theorem 5.2. *(Diakonikolas et al., 2019) If there exists an algorithm that given s independent samples from a q -histogram distribution $\mathcal{P}$ on $[0, 1]^d$ with $q > 2^{100d}$ and accepts with at least $2/3$ probability if $\mathcal{P} = \mathcal{U}$ and rejects with at least $2/3$ probability if $d_{TV}(\mathcal{P}, \mathcal{U}) \geq \epsilon$, then $s = (\sqrt{k}/\epsilon^2)\Omega(\log(q)/d)^{d-1}$.*

In the next paragraphs we will call a problem "computationally easy" if it can be solved using a polynomial amount of resources (time, random bits, etc.). This means that they essentially belong to the complexity classes **P**, **BPP** or the related ones (Arora & Barak, 2009).

The following definition tries to capture in a quantitative way the data forging process.

Definition 5.4 ((γ, δ) -data forgeable learning task). A learning task is (γ, δ) -data forgeable if an efficient algorithm exists to generate a new test sample such that, with probability at least δ , the error of the best predictor is with at least γ greater on the generated sample than the generalisation error of the predictor, and no algorithm can decide, using only a sample of sub-exponential (in d) size, with probability at least $2/3$, if the generated sample is from the same distribution or not.

³If the forgery is detected, and it can be proven, the transaction will not take place and the utilities will be 0.

Any attempt to apply the Theorem 5.2 for our purpose faces two difficulties. The main difficulty is that the model provider has limited information and computational power. This situation is modeled by assuming it has access only to the training and test samples, and does not know anything else about the underlying probability distribution, and in the process of cheating it must use efficient (polynomial time), possibly probabilistic, algorithms. The second obstacle came from the fact that the distribution must be such that the model provider should be able to generate a good predictor.

Proposition 5.3. *For any $m \geq 1000$, easy m -learning tasks that are also $(\frac{2}{5}, \frac{99}{100})$ -data forgeable exists.*

Proof. The proof is constructive. We will design a class of learning tasks and a probability distribution from which the forged sample will be drawn, with the following properties:

1. the class contains easy learning tasks;
2. differentiating the original test sample from a forged one is in general hard;
3. with high probability (bigger than $\frac{99}{100}$), the error of any predictor on the forged test sample is big (above $\frac{2}{5}$).

We choose as the distribution from which the original sample is drawn to be a q -histogram distribution on $[0, 1]^d$, with $q > 2^{100d}$ (the variable that we want to predict is included as the last entry of the vector, $y_i = x_i^{(d)}, \forall i \in [n]$). We assume that the test sample size is (at least) 1000 ($n - k \geq 1000$), and training sample size is 2000 ($k = 2000$).

The forged test sample will be drawn from the uniform distribution on $[0, 1]^d, \mathcal{U}$. If the data provider generates a new sample from this distribution, using Theorem 5.2 we can conclude that no algorithm can decide, using only a sample of sub-exponential (in d) size, with a probability of at least $2/3$, if the sample it received is from the same distribution as the training sample, or from the uniform distribution. Let us observe that sampling from the uniform distribution on the unit d -cube ($[0, 1]^d$) is computationally easy.

Now we will show that, with high probability, the error on the test sample is greater than the generalisation error of the best predictor by at least $\frac{2}{5}$. Note that we only want to show that this is true for at least one learning task. Such a learning task can be the one that has the label always 0, $\mathbf{P}(y = 0) = 1$ (it is easy to see that q -histogram distributions with this

property exists). The best predictor in this case is the function that assigns 0 to any instance, and its generalisation error will be also 0. On the other hand the generalisation error of this predictor on the uniform distribution is $\frac{1}{2}$. According to Definition 5.1, this is an easy learning task.

Let $h : [0, 1]^d \rightarrow \{0, 1\}$ be the best predictor for the learning task, and $x_i^{(d)}$ the label of the i -th (forged) test point ($i \in [n-k]$). Note that $x_i^{(d)}$ will be a Bernoulli random variable, and the errors for each point, $l_{01}(h(x_i), x_i^{(d)})$, will be independent Bernoulli random variables.

Using the Hoeffding bound for Bernoulli random variables, and the fact that $n - k \geq 1000$ we have:

$$\mathbf{P}\left(\frac{1}{2} - \frac{1}{n-k} \sum_{i=1}^{n-k} l_{01}(h(x_i), x_i^{(d)}) \geq \frac{1}{10}\right) \leq e^{-2(n-k)(\frac{1}{10})^2} \leq e^{-2 \cdot 1000/100} < \frac{1}{100}. \quad (11)$$

Therefore, with probability greater than $\frac{99}{100}$, the error of any (fixed) predictor on the (forged) test sample is with at least $\frac{1}{2} - \frac{1}{10} = \frac{2}{5}$ greater than the generalisation error of the best predictor. Since on any learning task the algorithm used for forgery uses a polynomial amount of resources, we have arrived at the conclusion. $\square$

A similar remark with the one made for Proposition 5.1 is valid for the new result: even for large samples, it's still possible to manipulate the data such that with a significant probability, the result is strongly affected, but the forgery is difficult to prove.

The main weakness of this result comes from the fact that it shows only the existence of such learning tasks, not that they are in any way "natural" or "typical". For example, almost all q-histogram distributions produces learning tasks that are difficult (in the sens that it is computationally or statistically hard to get a predictor with low generalisation error), so they are hard to forge, by our definition. On the other hand, this notion of forging is quite strong since it is relative to the best predictor, not the one generated by the model provider. Note that we also look at the absolute difference of the errors- again, a stringent requirement.

At a first glance, it may seems that there is an easier way to check if the test sample is from the same distribution as the training sample by using tools from statistical learning theory, more specifically error upper bounds (Shalev-Shwartz & Ben-David, 2014). The problem with this approach is that the bounds involve, at a minimum, a complexity measure for the hypothesis class (VC dimension, Rademacher complexity, etc.). It will be

difficult and undesirable for the model provider and data provider to agree on the maximum value of such a quantity.

In the proof we assumed that the forged sample is generated from the uniform distribution over $[0, 1]^d$. In reality this is of course not attainable, since computers work with finite quantities. Nevertheless, the finite approximation of the uniform distribution will have the same basic properties as the true one, and a slightly modified version of Theorem 5.2 can be applied.

5.3 Protocol analysis

This part is dedicated to analysing the viability of the market based on the proposed protocol (in other words, the protocol's correctness), and the computational complexity of the steps of the protocol.

5.3.1 Correctness

In this section we will formally show that the protocol described in Section 4 (see also Figure 1 and Table 1) works as expected. What we mean by this is that the two parties can both fulfill their goals: having a good predictor at the agreed price, for the data provider, and receiving the price based on the model's performance, for the model provider.

Beside the general setting described in Section 5.1, for a formal analysis we make the following assumptions:

- **Assumption 1.** The agents are rational and self interested (they are expected utility maximizers).
- **Assumption 2.** The hash functions are secure.
- **Assumption 3.** If the data provider changes a subset of the data points (by this we mean changing the features or the label), the lost utility if the points fall in the training set is greater than the gain obtained if they end up in the test set.
- **Assumption 4.** The initial dataset is *i.i.d.*. Moreover, the size of all datasets (for training, testing, and production) are large- say, larger than 1000.
- **Assumption 5.** Let δ and ϵ be some positive subunitary numbers, $0 < \delta < 1, 0 < \epsilon < 1$ (for concreteness, we take $\delta = 0.01$, and $\epsilon = 0.1$). We assume that the model provider can produce, with probability at least $1 - \delta$, a model h^+ such that $R[h^+] \leq err_{T_m} - \epsilon$ ($err_{T_m} = 1 - a_{T_m}$), err_{T_m} being the maximum allowed test error.

- **Assumption 6.** If the model provider doesn't apply the strategy described by Assumption 5, the only other option is to output a model h^- , possible at a much lower cost, but with the property that $R[h^-] \geq \text{err}_{T_m} + \epsilon$ (the symbols have the same meaning as before).
- **Assumption 7.** The following inequality is true ($c[h^+]$ is the cost of the model h^+):

$$\delta(p_m + \alpha) < (1 - 2\delta)(p_m - c[h^+]) - 2\delta c[h^+]. \quad (12)$$

Some of the assumptions are self explanatory and standard for such an analysis, but the assumptions 1, 3, 5, 6, and 7 requires some clarifications.

In the Assumption 1 we talk about the "expected utilities". These are defined as the expected values of the utilities for data and model provider, with respect to the drawn samples. Remember that the utilities are 0, if the transaction doesn't succeed, and are given by equations 5 and 6. For a particular model h , we denote the expected utilities by $U_d[h]$ and $U_m[h]$, for data provider and model provider, respectively.

The Assumption 3 says that the utility gained by changing a part of the sample, if it is used in the test sample, is less than the utility lost if that sub-sample is part of the training set. This assumption will ensure that the data provider will not have incentives to alter the data before the protocol is applied.

The Assumption 5 says that the model provider is able to generate a model with good expected accuracy. This is possible, for example, if empirical risk minimisation is performed with success for a class with finite VC dimension, using a large enough training sample (Shalev-Shwartz & Ben-David, 2014).

The message of Assumption 6 is that if the model provider tries to output a model at a significantly lower cost (e.g. by selecting a random predictor) than that required to properly train and validate a model, the expected accuracy of the model will be lower than some threshold. We can expect this to happen for any reasonable complex learning task.

The last assumption is a bit more technical. However, let us observe that when the samples sizes become larger, the probability δ can become smaller and smaller, and at some point the inequality will become true if $p_m - c[h^+] > 0$.

Proposition 5.4. *If the assumptions 1-7 are true, with probability at least $1 - 2\delta$, the protocol will finish successfully: the data provider will receive*

a model h^+ , and the model provider will get paid. Moreover, the expected utilities of the two agents will be lower bounded as follows:

$$U_d[h^+] \geq \alpha_d(1 - 2\delta)(1 - \text{err}_{T_m}), \text{ and} \quad (13)$$

$$U_m[h^+] \geq (1 - 3\delta)\alpha_m - 3\delta\alpha_m c[h^+]. \quad (14)$$

Proof. We split the proof into 4 steps.

Step 1. Data provider will participate in the transaction.

The data provider utility will be 0 (see Section 5.1) if it doesn't participate in the transaction. On the other hand, if it participates, and the agents behave properly, the model will have a big enough expected accuracy (it will be shown that this is true in the next steps). Since its utility is dependent on the accuracy on the production sample (see Equation 5), which is *i.i.d.*, with a non-zero probability, the utility for data provider will be higher than 0. Being a utility maximizer (Assumption 1), it will choose to participate.

Step 2. Training and test data received by the model provider will be i.i.d.. We will show that the data provider will not alter the data during the protocol execution.

The Assumption 1 and Assumption 3, together with the fact that the test sample is selected randomly (see Step 3), will ensure that both the training set and the test set will be initially *i.i.d.*. The steps 2, 4, 11 and 12 make it impossible for the data provider to alter the test set for its own benefit, without being detected (because of the hash computations and Assumption 2). Let us observe that if the data provider doesn't follow the rules, and this behaviour is detected, its utility will be 0. Therefore, the training and test datasets received by the model provide will be indeed *i.i.d.* samples.

Step 3. Model provider will participate in the transaction, and act to minimize the generalisation error.

By Assumption 5 and Assumption 6, the model provider has 3 options:

1. To not participate in the protocol;
2. To participate and produce a bad model (h^-);
3. To participate and produce a good model (h^+).

If the first strategy is chosen, the utility will be 0. We will now focus on the next two options.

Let us observe that the price is based on the accuracy computed on the test set, and the model provider does not have access to it (before Step 8).

Because the hash of the predictor is computed and sent to the data provider in Step 7, it will be impossible to change the model later (after steps 8 or 10), using the received test data. Therefore, the model provider can only use the training data.

If the model provider produces the model h^- , in the most optimistic scenario, when $R[h^-] = err_{Tm} + \epsilon$ (see Assumption 6), the probability the error on the test sample will be below the threshold ($\hat{R}_{S_{test}}[h^-] \leq err_{Tm}$) can be expressed in the following way:

$$\mathbf{P}(\hat{R}_{S_{test}}[h^-] \leq err_{Tm}) = \mathbf{P}(err_{Tm} + \epsilon - \hat{R}_{S_{test}}[h^-] \geq \epsilon) = \mathbf{P}(R[h^-] - \hat{R}_{S_{test}}[h^-] \geq \epsilon). \quad (15)$$

Using Hoeffding's inequality, we can write:

$$\mathbf{P}(\hat{R}_{S_{test}}[h^-] \leq err_{Tm}) \leq e^{-2(n-k)\epsilon^2} \leq \delta, \quad (16)$$

since $e^{-2(n-k)\epsilon^2} = e^{-2 \cdot 1000 \cdot 0.1^2} \leq 0.01 = \delta$.

If the error is above the threshold, the utility will be 0. Otherwise, it will be upper bounded by $\alpha_m(p_m + \alpha)$ (the best case will occur when the cost is 0, and the accuracy is 1). Therefore, using Equation 16, we can conclude that the expected utility will be upper bounded as:

$$U_m[h^-] \leq (1 - \delta) \cdot 0 + \delta \alpha_m(p_m + \alpha) = \delta \alpha_m(p_m + \alpha). \quad (17)$$

If the model provider tries to generate h^+ , by Assumption 5, it will succeed with probability at least $1 - \delta$. In the most pessimistic scenario, when $R[h^+] = err_{Tm} - \epsilon$ (see again Assumption 5), we have

$$\mathbf{P}(\hat{R}_{S_{test}}[h^+] \geq err_{Tm}) = \mathbf{P}(\hat{R}_{S_{test}}[h^+] - (err_{Tm} - \epsilon) \geq \epsilon) = \mathbf{P}(\hat{R}_{S_{test}}[h^+] - R[h^+] \geq \epsilon), \quad (18)$$

and using one more time Hoeffding's inequality we have

$$\mathbf{P}(\hat{R}_{S_{test}}[h^+] \geq err_{Tm}) \leq e^{-2(n-k)\epsilon^2} \leq \delta. \quad (19)$$

Observing that in the worst case the price will be just p_m , and if the model will fail to achieve the minimum accuracy (the training or the test fails), the utility will be negative ($-\alpha_m c[h^+]$), using the union bound, the expected utility can be lower bounded as:

$$U_m[h^+] \geq (1 - 2\delta) \alpha_m(p_m - c[h^+]) - 2\delta \alpha_m c[h^+]. \quad (20)$$

Using the Assumption 7, we can conclude that $U_m[h^+] \geq U_m[h^-]$, and by Assumption 1, the model provider will choose to try to produce the model h^+ .

Step 4. Computing the minimum expected utility for the data provider

Taking into account the previous step, the fact that the production sample is *i.i.d.*, and large enough (> 1000 , by Assumption 4), we can apply the Hoeffding's inequality in the following way:

$$\mathbf{P}(\hat{R}_{S_{production}}[h^+] - R[h^+] \geq \epsilon) = \mathbf{P}(\hat{R}_{S_{production}}[h^+] - (err_{T_m} - \epsilon) \geq \epsilon) \leq e^{-2(n_p)\epsilon^2} \leq \delta, \quad (21)$$

where n_p is the size of the production sample, and we have considered the least favorable scenario, that is $R[h^+] = err_{T_m} - \epsilon$. This means that the union bound will imply that the expected utility for the data provider will be lower bounded as

$$U_d[h^+] \geq \alpha_d(1 - 3\delta)(1 - err_{T_m}). \quad (22)$$

The union bound was applied for three events: training failure, test failure, and production failure (by "failure" we mean that the error is above the threshold), each one having a probability of at most δ .

For the protocol to succeed, only two events needs to be avoided, training failure and test failure, hence the probability of success is at least $1 - 2\delta$. $\square$

The Figure 2 presents, in a simplified way, the interaction mediated by the protocol as a 2 players game. If one of the players tries to cheat, the other player will be able to detect this behavior (see the proof of the Proposition 5.4), and the outcome will be 0 for both players (in fact, we can argue that it will be negative, since the participation will involve some resources). On the other hand, if both players are honest, the result will be positive for both of them. The game has two pure strategy Nash equilibrium points. They arise when both players have the same behaviour. But one equilibrium point has a 0 payoff for both players, while the other has strictly positive payoffs.

5.3.2 Computational complexity

In this part we will analyse the computational complexity of the processings required by the protocol. Note that we are only interested in those computations that are inherent to the protocol itself, not the "useful" computations (e.g. training the model, evaluating the error, etc.). The complexity of

		Model Provider	
		Doesn't participate or cheat	Participates and doesn't cheat
Data Provider	Doesn't participate or cheat	(0,0)	(0,0)
	Participates and doesn't cheat	(0,0)	$(U_d[h^*], U_m[h^*])$

Figure 2: Schematic view of the protocol’s outcome as a 2 players game.

model training and evaluation depends on the specific approaches (e.g. the learning algorithm, etc.), are not our concern (they are necessary even if the protocol is not used). Therefore, the theoretical complexity of the protocol will be defined as the complexity of the most expensive protocol step.

Usually, computing the cryptographic hash can be done in time $O(n)$. This is also the case for the hash used in our implementation, SHA-2 (Lilly, 2004; Cormen, Leiserson, Rivest, & Stein, 2009).

To get k distinct numbers from the set $\{1, 2, 3, \dots, n\}$, $k \leq n$, one can use the following approach: shuffle the set $\{1, 2, 3, \dots, n\}$, then take the first k values. The shuffling can be achieved using the classic Fisher–Yates algorithm (Fisher & Yates, 1963), therefore the entire step has a complexity of $O(n)$.

All the other steps are trivial, requiring constant or linear time (if we need to iterate over the entire dataset). We will summarize the conclusions of this sub-section in the following proposition:

Proposition 5.5. *The computational complexity of the protocol is $O(n)$.*

Since the computational effort required by most machine learning algorithms is super-linear (Shalev-Shwartz & Ben-David, 2014), we can conclude that the proposed protocol is indeed lightweight.

6 A reference implementation and case studies

We implemented⁴ the protocol using a REST API (Fielding & Taylor, 2000). The API is based on the lightweight Python web framework Flask⁵ (Dwyer,

⁴The code will be made publicly available on Github.

⁵<https://flask.palletsprojects.com/en/1.1.x/> (Accessed: 2021-02-10)

Aggarwal, & Stouffer, 2017), and the implementation follows the general client-server architecture.

The model providers and data providers act as clients, and the server will offer what is needed to make the interaction possible. The Table 2 presents all the endpoints of the server, together with their descriptions. The endpoints correspond, roughly, to protocol’s steps.

Although a server is present, the architecture does not enforce a strictly centralised model of interaction. The model provider and data provider can easily establish a secure communication tunnel- the server acts as a facilitator. A completely decentralised setup, without a server, is also possible, but undesirable from a practical perspective. In that case the data provider should know the addresses of all the model providers to be able to launch the auction.

Endpoint	HTTP Method	Query Parameters	Description
/upload_training	POST	Data , Hash	Load training data and hash
/randomgen	GET	Count,Limit	Generate random indices
/datasets	GET	-	Get all available datasets
/bid	PUT	Dataset, Price	Place a bid
/training	GET	Dataset	Get the dataset
/upload_test_nolabels	POST	-	Upload the test data without labels
/test_nolabels	GET	Dataset	Get the test dataset, without labels
/upload_test_labels	POST	-	Upload the test labels
/upload_predictor_hash	POST	-	Load the predictor hash
/predictor_hash	GET	Dataset	Get the predictor hash
/upload_predictor	POST	-	Load the predictor
/predictor	GET	Dataset	Get the predictor

Table 2: The API endpoints

6.1 Experimental results

We present a few case studies to illustrate from a practical standpoint how the protocol can be used, and what are the associated costs.⁶ For this purpose, we have designed three experiments. They are based on publicly available datasets and consists of simulating the application of the protocol. The most important metric is the time overhead, that is the time needed to do the computations required by the protocol. The time overhead is compared with the time consumed to do the actual task: constructing and evaluating the model. Since the training is overwhelmingly dominating in all cases, we have only included the training time.

There is also an overhead in terms of memory and communication, but since we only store and transfer integer numbers and fixed length hashes, it is negligible in all cases.

The datasets and learning algorithms were selected to reflect both extreme (unfavourable) cases (e.g. small dataset, very fast learning algorithm), and more typical ones. The experiments were performed on a machine with an Intel(R) Core(TM) i7-9850H CPU, 2.6 GHz, processor, with 16 GB of RAM, running Windows 10 Pro.

For the first experiment, we choose a text classification task. The dataset is public and consist of around 200000 news headlines from the year 2012 to 2018 obtained from HuffPost⁷ (in the rest of the paper, we call it News Dataset). The learning task requires to predict the category to which an article belongs (e.g. entertainment, politics, travel, sports, etc.). Therefore it is a multiclass classification problem, and it can, in our opinion, be considered an average, typical machine learning problem.

We set the size of the training set to be $k = 160000$ (80%), and the rest of the data was devoted to the model testing. We have tried to find a simple scenario that retains some practical relevance. In our case this mean a learning algorithm that is known to be very fast and is used in practice (although it is not necessary the most accurate one). Based on this reasoning, we chose the Multinomial Naïve Bayes classifier (Mitchell, 1997), which is very simple and fast and, in addition, it is used often for text classification (Manning, Raghavan, & Schütze, 2008). Other classifiers, like Random Forest and Deep Neural Networks provide, of course, better results(Shalev-Shwartz & Ben-David, 2014; Goodfellow et al., 2016), but the training time is also higher, and the overhead in that cases becomes less

⁶The code used for the experiments will be made publicly available on Github.

⁷https://github.com/nomadotto/News_Classifier/blob/master/News_Category_Dataset_v2.json
((Accessed: 2021-02-10))

important.

As mentioned before, we measured the time spent on the intrinsic ("useful") computation, that is the training and validation (including the pre-processing), and the total time overhead added by the protocol steps- the time not required if we have used a direct approach. In order to get more robust results we repeated the experiment 5 times. The results are presented in Figure 3. The time spent on training (89.12 seconds on average) surpasses by far the additional time required by the protocol (4.07 seconds on average).

Based on the previous observations we have chosen two smaller datasets. They are both from UCI Machine Learning Repository⁸(Dua & Graff, 2017). The first one is the classic "Ionosphere Dataset" (Sigillito, Wing, Hutton, & Baker, 1989), and the second one is newer and consist of medical records of patients having heart failure (Chicco & Jurman, 2020) (in the rest of the paper, we will call it "Hart Failure Dataset").

Our experiments show that in the extreme case of Naïve Bayes on small datasets, the time needed for protocol execution can be larger (up to 4 times) than the time needed to actually build the model (see Figure 4 and Figure 6). Although relatively high, the cost of the protocol is still manageable. In the case of a Neural Network (implemented using the Tensorflow framework⁹), the time overhead becomes once again negligible (see Figure 5 and Figure 7).

7 Conclusion

Through this work we provided an analysis of a potential market for on-demand machine learning models. The main identified challenge was the double information asymmetry. In order to solve this issue a simple protocol was designed. The theoretical and empirical analysis shows that, under some basic assumptions, the protocol achieve it's goal, with a typically small overhead.

We describe a reference implementation of the protocol in the Python programming language. The protocol is flexible, and easy to implement on any platform, using any programming language. It can be easily adapted for different data and machine learning tasks, and we hope it can be a valuable resource in making the process of machine learning solutions development more efficient and safe.

⁸<https://archive.ics.uci.edu/> (Accessed: 2021-02-10)

⁹<https://www.tensorflow.org/> (Accessed: 2021-09-15)

Figure 3: A comparison between the time spent on the actual computation and the time needed to perform all the protocol's steps for News Dataset when Naïve Bayes is used.

Figure 4: A comparison between the time spent on the core computations, and the time needed to perform all the protocol's steps for Hart Failure Dataset when Naïve Bayes is used.

Figure 5: A comparison between the time spent on the core computations, and the time needed to perform all the protocol’s steps for Hart Failure Dataset when Neural Networks are used.

The current work rises some additional questions that were not fully addressed in the paper. One such problem is that of machine learning models cost estimation. We have assumed that the model provider is able to estimate the cost for generating a model by looking at the training dataset, and being able in this way to participate in the auction. The assumption seems to be reasonable, but we have not investigated how this can be achieved, and at what cost.

We have also taken for granted that the agents have some utility functions that they want to maximize. We assumed some simple functions, but it might be worth investigating further this aspect. Moreover, in this work we have not investigated the behaviour of the market as whole (e.g. equilibrium, efficiency, etc.). These seems to be important problems, worth pursuing.

Figure 6: A comparison between the time spent on the core computations, and the time needed to perform all the protocol’s steps Ionosphere Dataset when Naïve Bayes is used.

Figure 7: A comparison between the time spent on the core computations, and the time needed to perform all the protocol’s steps for Ionosphere Dataset when Neural Networks are used.

8 Declarations

- **Funding.** No founding was received for the present research.
- **Conflict of interest/Competing interests.** The author has no conflict of interest or competing interests to declare.
- **Ethics approval.** Not Applicable.
- **Consent to participate.** The author gives his consent for participating in the review process.
- **Consent for publication.** The author gives his consent for publishing the paper in the journal, if the decision is favorable.
- **Availability of data and materials.** All data used in the experiments is publicly available (see Section 6.1 for details).
- **Code availability.** The code will be released publicly on Github. Until then it can be made available to the editors or reviewer on request.
- **Authors' contributions.** The paper has a single author which was responsible of initiating the research, performing the theoretical and empirical work, and writing the paper.

References

- Abernethy, J., Chen, Y., Ho, C.-J., & Waggoner, B. (2015). Low-cost learning via active data procurement. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, p. 619–636, New York, NY, USA. Association for Computing Machinery.
- Akerlof, G. A. (1970). The market for 'lemons': Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3), 488–500.
- Amershi, S., Begel, A., Bird, C., DeLine, R., Gall, H., Kamar, E., Nagappan, N., Nushi, B., & Zimmermann, T. (2019). Software engineering for machine learning: A case study.. *ICSE-SEIP '19*, p. 291–300. IEEE Press.
- Archak, N., & Sundararajan, A. (2009). Optimal design of crowdsourcing contests. In *ICIS 2009 Proceedings*. 200.

- Arora, S., & Barak, B. (2009). *Computational Complexity: A Modern Approach* (1st edition). Cambridge University Press, USA.
- Arrow, K. (1962). Economic welfare and the allocation of resources for invention. *1962*, 5.
- Blum, A., & Hardt, M. (2015). The ladder: A reliable leaderboard for machine learning competitions. In *Proceedings of the 32nd International Conference on Machine Learning*.
- Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory, COLT' 98*, p. 92–100, New York, NY, USA. Association for Computing Machinery.
- Cavallo, R., & Jain, S. (2012). Efficient crowdsourcing contests. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2, AAMAS '12*, p. 677–686, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
- Chawla, S., Hartline, J., & Sivan, B. (2019). Optimal crowdsourcing contests. *Games and Economic Behavior*, 113, 80–96. Funding Information: Supported in part by NSF award CCF-0830494 and in part by a Sloan Foundation fellowship. Supported in part by NSF award CCF-0830773. Supported in part by NSF award CCF-0830494. Publisher Copyright: © 2015 Elsevier Inc.
- Chen, Y., Immorlica, N., Lucier, B., Syrgkanis, V., & Ziani, J. (2018). Optimal data acquisition for statistical estimation. In *Proceedings of the 2018 ACM Conference on Economics and Computation, EC '18*, p. 27–44, New York, NY, USA. Association for Computing Machinery.
- Chen, Y., & Zheng, S. (2019). Prior-free data acquisition for accurate statistical estimation. In *Proceedings of the 2019 ACM Conference on Economics and Computation, EC '19*, p. 659–677, New York, NY, USA. Association for Computing Machinery.
- Chicco, D., & Jurman, G. (2020). Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. *BMC Medical Informatics Decis. Mak.*, 20(1), 16.
- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, Third Edition* (3rd edition). The MIT Press.
- Diakonikolas, I., Kane, D. M., & Peebles, J. (2019). Testing identity of multidimensional histograms. In Beygelzimer, A., & Hsu, D. (Eds.), *Pro-*

ceedings of the Thirty-Second Conference on Learning Theory, Vol. 99 of *Proceedings of Machine Learning Research*, pp. 1107–1131, Phoenix, USA. PMLR.

- Dua, D., & Graff, C. (2017). UCI machine learning repository..
- Dwyer, G., Aggarwal, S., & Stouffer, J. (2017). *Flask: Building Python Web Services*. Packt Publishing.
- Fielding, R. T., & Taylor, R. N. (2000). *Architectural Styles and the Design of Network-Based Software Architectures*. Ph.D. thesis.
- Fisher, R. A., & Yates, F. (1963). *Statistical tables for biological, agricultural and medical research* (6rd edition). Edinburgh: Oliver and Boyd.
- Gast, N., Ioannidis, S., Loiseau, P., & Roussillon, B. (2020). Linear regression from strategic data sources. *ACM Trans. Econ. Comput.*, 8(2).
- Goldreich, O., Micali, S., & Wigderson, A. (1991). Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3), 690–728.
- Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, STOC '85, p. 291–304, New York, NY, USA. Association for Computing Machinery.
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. The MIT Press.
- Hillier, B. (1997). *The Economics of Asymmetric Information*. Palgrave, London.
- Hoppe, E. I., & Schmitz, P. W. (2013). Contracting under Incomplete Information and Social Preferences: An Experimental Study. *Review of Economic Studies*, 80(4), 1516–1544.
- Johnston, D. (2018). *Random Number Generators Principles and Practice: A Guide for Engineers and Programmers*. De—G Press, Berlin, Germany.
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260.
- Kääriäinen, M. (2005). Generalization error bounds using unlabeled data. In *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005*, pp. 127–142. COLT.

- Lambert, R. A., Leuz, C., & Verrecchia, R. E. (2011). Information Asymmetry, Information Precision, and the Cost of Capital*. *Review of Finance*, 16(1), 1–29.
- Lavalle, S., Lesser, E., Shockley, R., Hopkins, M., & Kruschwitz, N. (2011). Big data, analytics and the path from insights to value. *MIT Sloan Management Review*, 52, 21–32.
- Lilly, G. M. (2004). U.s. patent us6829355b2, device for and method of one-way cryptographic hashing..
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.
- Marwala, T., & Hurwitz, E. (2015). Artificial Intelligence and Asymmetric Information Theory. *ArXiv*, -(–), –.
- Marwala, T., & Hurwitz, E. (2017). Information Asymmetry. *Artificial Intelligence and Economic Theory: Skynet in the Market. Advanced Information and Knowledge Processing*, -(–), –.
- Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (2001). *Handbook of Applied Cryptography*. CRC Press.
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York.
- Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of Machine Learning* (2nd edition). The MIT Press.
- Prashant Johri, Jitendra Kumar Verma, S. P. e. (2020). *Applications of Machine Learning*. Springer Singapore, Singapore.
- Rigollet, P. (2007). Generalization error bounds in semi-supervised classification under the cluster assumption. *J. Mach. Learn. Res.*, 8, 1369–1392.
- Rothschild, M., & Stiglitz, J. (1976). Equilibrium in competitive insurance markets: An essay on the economics of imperfect information. *The Quarterly Journal of Economics*, 90(4), 629–649.
- Segev, E. (2020). Crowdsourcing contests. *European Journal of Operational Research*, 281(2), 241–255.
- Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, USA.
- Sigillito, V., Wing, S., Hutton, L., & Baker, K. (1989). Classification of radar returns from the ionosphere using neural networks..
- Spence, M. (1973). Job market signaling. *The Quarterly Journal of Economics*, 87(3), 355–374.

- van Engelen, J. E., & Hoos, H. (2020). A survey on semi-supervised learning. *Machine Learning*, 109, 373–440.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization..
- Zhang, J. M., Harman, M., Ma, L., & Liu, Y. (2020). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, -(), 1–1.
- Zhang, M., Arafa, A., Wei, E., & Berry, R. (2021). Optimal and quantized mechanism design for fresh data acquisition. *IEEE Journal on Selected Areas in Communications*, 39(5), 1226–1239.
- Zhao, Y., Yu, Y., Li, Y., Han, G., & Du, X. (2019). Machine learning based privacy-preserving fair data trading in big data market. *Information Sciences*, 478, 449–460.