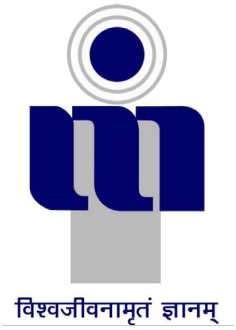
Collaborative Artificial Intelligence in a Public Blockchain Network

A project report submitted in partial fulfillment of the requirements for B.Tech. Project

IPG M.Tech.

by

Rathin R (2019IMT-081)



ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT
GWALIOR-474 015

2022

CANDIDATES DECLARATION

I hereby certify that the work, which is being presented in the report, entitled **Collaborative Artificial Intelligence in a Public Blockchain Network**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the institution is an authentic record of my own work carried out during the period *June 2022* to *September 2022* under the supervision of **Dr. Saumya Bhadauria**. I have also cited the references about the text(s)/figure(s)/table(s) from where they have been taken.

Date: Signatures of the Candidates

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Date: Signatures of the Research Supervisor

ACKNOWLEDGEMENTS

I am highly indebted to Dr. Saumya Bhadauria and thank her for allowing me the independence to work and try out new concepts. I want to take this opportunity to express my sincere gratitude to her for her academic guidance, interest in this project, and ongoing support, along with the sessions she held to boost my confidence and motivate me. These sessions proved to be very successful and were crucial in helping to instill self-assurance and trust in me. She provided invaluable direction, suggestions, wise judgment, constructive criticism, and an eye for perfection that helped the current work grow and flourish. My mentor never let me feel like a rookie by always listening to my opinions, recognizing and refining them, and giving me free rein in my project. She always responded to all of my doubts with smiling graciousness and prodigious patience. Because of her intense interest and friendly demeanor, the current work has reached this point. Finally, I want to express my gratitude to our institution and my coworkers, whose unfailing support enabled me to finish this work by reviving my spirit and refocusing my attention and energy.

ABSTRACT

Significant advancements in artificial intelligence have recently been made possible through machine learning. However, these tend to be highly centralized. Predictions are frequently sold on a per-query basis, the enormous datasets needed are typically proprietary, and published models can soon lose their usefulness with little to no work on the user's part (without continuously training and updating the model). It is a system that enables users to construct a dataset jointly and host a model that is constantly updated using smart contracts. Everyone will be able to access this model on a blockchain. Personal assistants, playing games, recommender systems, and other situations where a model is repeatedly utilized for comparable input qualify as ideal learning problems.To maintain the model's accuracy for some test sets, we propose monetary and non-monetary incentive mechanisms for contributing high-quality data.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IM	Incentive Mechanism
DCAI	Decentralized and collaborative Artificial Intelligence
ML	Machine Learning
POW	Proof-of-Work
API	Application Programming Interface
AI	Artificial Intelligence
Dapp	Decentralized Application

NOTATIONS

Loss(x)	function to calculate loss (incentive mechanism)
Avg(t)	function to calculate average time
W(t)	weight at time t vector/list of numbers
P(c|x)	probabilty of c given x
N(p)	number of reimbursed data sets of contributors
F(p_f, s)	payment made to contributor for submitting data with deposit f

CHAPTER 1

Introduction

We propose a technology that utilizes Blockchain and Artificial Intelligence for partaking and updating a ML model. Under this approach, anybody can use the model to obtain prdeictions and provide data to the model for training. The system must be dependable to promote participation and prevent manipulation. Our system [1] is divided into categories, and we suggest and defend various examples of "incentive mechanisms" and their benefits. There are several ideas for fusing blockchain and machine learning systems. Permission to the trained model is restricted to a market in systems like DInEMMo [1].

This limits access to those who can afford it while allowing contributors to profit from how a model is used. DanKu [7] offers storing previously learned models for competitions using smart contracts. This, however, prevents ongoing training and group instruction. This initiative seeks to counteract the current concentration of artificial intelligence by promoting model sharing. This centralization includes centralized private data, access to machine learning model projections, and machine learning expertise.(e.g., charged on a per-query basis).

1.1 Problem/Motivation

Difficult to set up AI/ML systems :

- Lack of Quality/Inaccessible Data: Data being one of the most valuable entity in this era is quite inaccessible to common people or those that we get our hands into are poor quality data.
- Inadequate Infrastructure : To train Large data sets a computer requires high performance GPU, which is apparently quite expensive.

Model Decay : A Model looses its accuracy if it is not updated regularly. For example considering a model which predicts the rent of houses in an area. If the model is not

updated regularly it will still calculate rents based on previous values not considering the new rents as time goes by(rent prices increases in real time).

AI Skills are centralized : Data sets are largely owned by certain organizations (eg.Google) which then sell those data collected from the users on per query basis. Google generates a revenue of 92M$ every 24hrs. By storing models in smart contracts [15], we can host models that evolve and use them to help build a dataset that is used to train other models on or off-chain.

The output of our system is effectively two things: the dataset and the model. The model can be used by production applications or just used to help validate data. The data can then train other models off-chain or even on-chain. Instead of getting people to add skills, people can improve the base Cortana model.

1.2 Objectives

The objective is to create a decentralized application which can train models using blockchains computation power and use artificial intelligence to predict outcomes in the Dapp.To encourage people to put data into our system there is an incentive mechanism formulated which rewards users who contribute good data to the model and punishes users those who contribute incorrect data by charging them penalty.As people start using this system a large and authentic datasets will be available publicly in the blockchain till eternity. People can use these datasets available in the blockchain or get predictions from the models that are already present in the Dapp.

- Store Models in the Blockchain. Put Models in Smart Contracts so that anyone can update them, they can be used for free since they only read the contract's state.
- Now that we have a public Model , we'll let people update it use those updates to build a dataset.
- To create an immutable warehouse of datasets that, once stored in the blockchain as "good data," cannot be manipulated—exploiting the hackproof property of blockchain to secure the datasets from manipulators.
- The main goal is to create a "Wikipedia" of datasets. Any common man should have access to data for free at the ease of technology at our fingertips.Models can be trained in this Dapp by giving a small amount of gas fees to the network.This can be then trained using blockchains computation power.

CHAPTER 2

Literature review

2.1 Background

Significant advancements in artificial intelligence have recently been made possible through machine learning. However, these tend to be highly centralized. Predictions are frequently sold on a per-query basis, the enormous datasets needed are typically proprietary, and published models can soon lose their usefulness with little to no work on the user's part (without continuously training and updating the model). It is a system that enables users to construct a dataset jointly and host a model that is constantly updated using smart contracts. Everyone will be able to access this model on a blockchain. Personal assistants, playing games, recommender systems, and other situations where a model is repeatedly utilized for comparable input qualify as ideal learning problems.To maintain the model's accuracy for some test sets, we propose monetary and non-monetary incentive mechanisms [10] for contributing high-quality data.

2.2 Overview

We demonstrate the capacity of a novel framework [4] to massive aggregate amounts of data, allow contributors to possibly profit, and host a shared machine learning model as a public resource by leveraging developments in AI, prediction markets, and blockchain platforms.

Several programmable components help achieve this:

- *the incentive mechanism*
- *the data handler*
- *the machine learning model*

Options for these elements are selected while creating and initializing a smart contract. It then accepts "add data" actions from participants, with the incentive mecha-

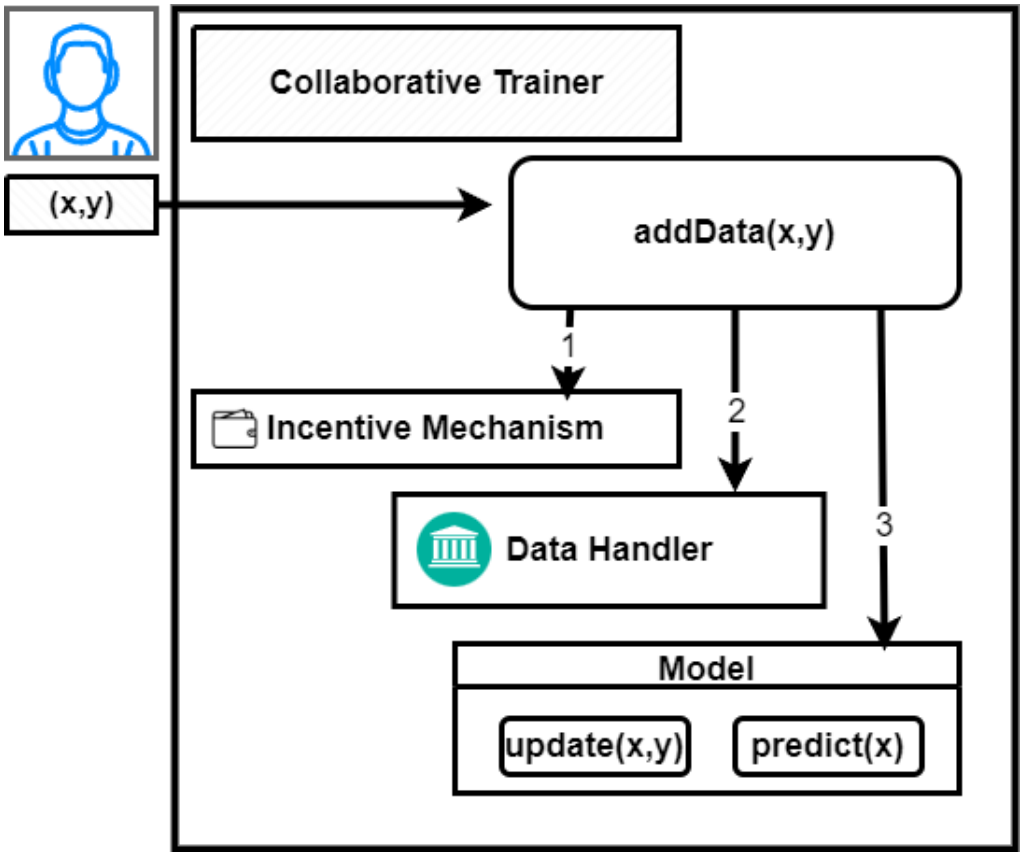


Figure 2.1: Diagram Showing the components of DCAI

nism potentially resulting in compensation or opening the door to more functionalities. Adding data entails validating the incentive mechanism as shown in **Fig.2.1**, before putting it in the data handler and then invoking the update procedure on the model's contract.

Our method aims to create valuable shared resources, not to make money for the creators. It is feasible for data contributors to get money (depending on the incentive system), but this is primarily due to controls that punish those who submit inaccurate data.The dataset is also public because it may be accessed through emitted events or the blockchain's transaction history (if this feature is available to the blockchain framework). Using standard crowdsourcing [3] services like Figure Eight (formerly Dolores Lab, CrowdFlower) and Amazon Mechanical Turk can be expensive when collecting massive datasets. Filtering out "bad data" in crowdsourcing is an ongoing struggle because spammers can submit low-effort or nonsensical data and still be paid for their work . Our incentive systems don't provide contributors with any benefits.Bad data submissions can even result in payment of the fine, whereas sincere contributors are actively rewarded for pointing out others' errors.

2.3 Machine Learning and Blockchain Background

In our analysis, the main focus was supervised learning problems with labeled samples in the dataset [5]. For instance, a restaurant or movie may receive a label of 1 to 5 stars in our recommender system.A model [2] is a machine learning algorithm that has been trained on data. It is employed to make predictions, such as determining the label of an example that has been provided. It can be represented as a neural network, decision tree, or other type of data structure.

On platforms where decentralized networks have selected a typical computational order, our method [1] is applicable. One example is the blockchain used by Ethereum. A smart contract is an object in this shared code in the sense of object-oriented programming.It has data fields and uses method calls to communicate with new code and events. A computation that is performed inside of a smart contract is referred to as being on-chain. On the blockchain, the computation's input and output are typically kept. Off-chain, on the other hand, allows the computation to be performed locally on the client's computer and is not required to be made public.

2.4 Staking a Deposit

Using a smart contract to set a sentence is challenging because a user cannot be made to pay. Instead, several blockchain-based [11] solutions demand that users "stake" deposits that can be later reclaimed if they follow the rules. We will also suggest taking a deposit to streamline some "reward" for supplying new data, similar to those methods.

CHAPTER 3

Methodology

The hypothesis and the analytical support for the suggested solution are presented in this section.The suggested incentive mechanisms (IM) [4] incentivize participants to provide information that will increase the model's precision. There are several ways to measure this. The most transparent proxy is to evaluate performance concerning a particular test set. If a test set is unavailable, we will also review other performance evaluation techniques. This paper aims to provide the overall framework with compelling examples; hence, each incentive mechanism will be thoroughly examined in subsequent work. We refer to good data as factually accurate information in the following parts. For instance, the label "1" is unquestionably preferable to the label "0" for an image of the number 1. Additionally, data can be inaccurate or unclear.

3.1 Gamification

To [10] lower the entry barrier, we first suggest a baseline with no financial incentives. This is the Models Datasets section of Wikipedia. This idea exclusively depends on contributors' desire to work together for free and for the common good. Additionally, data producers may be rewarded with points and/or badges via stackexchangification.Badges on Stack Exchange have been demonstrated to be valid. The contributor's wallet address can be used as a key or identifier to record the points and badges in a smart contract on the blockchain. Here are some illustrations of measures for granting a user points or badges:

- A specified number of data samples has been contributed.
- Submitting diverse data samples.
- Submitting data with different labels
- Submitting data routinely (e.g. weekly)

Table 3.1: Perceptron models are only updated when h(x) ≠ *y*.

Action	Gas fees	Cost($)
Deployed smart contract	3,845,840	$ 4.06
Add data with 15 words (h(x)=y)	177,693	$ 0.19
Add data with 15 words (h(x)≠ *y*)	249,037	$ 0.26

It is necessary to do additional tests to determine how these metrics can be expanded off-chain or specifically computed efficiently on-chain.

3.2 Rewards Mechanism Based on Prediction Markets

In this part, we outline a mechanism that rewards contributors of accurate data with money [10]. An outside source, such as an academic institution or business, provides a pool of reward monies and a test dataset. Participants are compensated based on how well they raise the performance of the model as determined by the test results. We will have the ability to offer extremely substantial incentives for participation once this source is accessible. The system is also resistant to deceitful or manipulative participants and providers.

1. *Overview:* **Fig. 2.1** shows how the system works. There are three stages. During the commitment phase, the provider deposits the bounty and creates a loss function L. (M, S). This is either a substitution or proxy metric or a measurement of loss for any model h on any dataset S. (typically the average loss on points in the dataset). In a similar manner, the provider then cryptographically commits to a test dataset, of which only a tiny random subset is initially made available. People input data to the model or alter it in other ways throughout the participation period.

   A.Commitment Phase

   (a) Provider deposits C units of currency

   (b) Provider defines a loss function Loss(m, S).

   (c) Provider secretly divides a test dataset into 100 equal parts and uploads their 100 cryptographic hashes.

   (d) Smart contract randomly selects 10 of these hashes.

(e) Provider uploads 10 partial datasets. If they do not match the 10 hashes, abort.

(f) Provider specifies end condition (e.g. time limit).

B. Participation Phase

(a) Smart contract contains an initial model, m_0.

(b) for each participant t = 1, 2, . . . , T until end condition is met: do

(c) Participant deposits a stake of 1 unit of currency

(d) Participant provides data.

(e) Model is updated from $m_t - 1$ *to* m_t.

C. Reward Phase

(a) Provider uploads 90 partial datasets; call them S. If they do not match the remaining 90 hashes, abort.

(b) Let $c_t = 1$ for all t // balance initially equals stake

(c) Let list L = (1, . . . , T) // list initially contains everyone

(d) **for** i = 1, . . . , C **do**
for each participant t in L **do**
Let t' be previous participant in L, or 0 if none.
Participant t's balance is changed:

$$c_t \leftarrow c_t + Loss(m_{t'}, D) - Loss(m_t, D)$$

(e) Let list $S = (t \in L : c_t \geq 1)$. // all who can re-stake 1 stay in L

(f) Each participant t is paid b_t.

The supplier uploads the test dataset during the reward phase, and the smart contract verifies that it fulfills the agreement. 3 The smart contract then decides on rewards for each participant, as will be covered later.

2. *Reward Calculation:* Consider for a moment that the bounty B = 1, meaning that the payout for each player t is their stake plus the following number:

$$Loss(m_{t-1}, D) - Loss(m_t, S) \tag{3.1}$$

This is precisely the reward function suggested in , which is based on prediction markets with automated market makers or scoring rules .It can be visualised like this: The first participant receives $Loss(m_0, S)$ initially from the smart contract.They compensate [13] the second participant with $Loss(m_1, S)$ after their

data updated the model to h_1.This continues until $Loss(h_t, s)$ is returned to the smart contract by the final participant. The more effectively ht performs, the less each member is required to pay back,therefore they are incentivized to offer information that is as helpful in relation to the (anticipated) test set as possible.(T loses part or all of its stake if ht performs worse than the prior model.) The smart contract pays the entire improvement from all contributions as a net amount, equal to $Loss(m_0, S) - Loss(m_T, S)$.

By assuming the loss function, it can only be 1 or less. In order to scale this technique for a bounty of B>>1, we must do so last. The method of calls for everyone to stake B, which is impossible. As a result, we take the method of repeatedly iterating mechanism B. The player stakes one unit for each round, after which they are rewarded.She withdraws if she cannot stake 1 unit any longer due to losses. 4 Even though this is a little complicated, the better ht, the greater reward t receives, still holds, and we think this strongly aligns with the incentives for involvement.

3. *Untrusted provider, commitment, and value burning:* The supplier can manipulate by selecting the dataset and then participating in the participants' partnerships or collaborations. The cryptographic commitment system and value "burning," which happens when some value transferred to the smart contract is not returned to anyone, are the defenses against manipulation, as we will explain in more detail below.

   The provider must provide a random 10 % of the dataset in advance under the cryptographic commitment mechanism (of course, the numbers of 10 and 100 can be adjusted as desired). The process is stopped, and incentive B is kept in the contract if the provider doesn't comply (not refunded). Participants discover something regarding the final dataset, assuming the provider complies. This avoids the problematic attack described below, in which a provider secretly selects a dataset with inaccurate labels before contributing anonymously with the necessary modifications.The provider might then not only recoup the majority of the initial payment but also profit significantly from the honest participants' stakes, which they would otherwise forfeit because their information is useless for this test set. As a result of the commitment scheme's [12] upfront disclosure of a representative portion of the test setup, participants are more likely to be made aware of this attack and decline to take part. After the procedure, a sizeable portion of the initial prize is probably still locked in the contract. Only the following sum is distributed in detail:

$$C.[Loss(m_0, S) - Loss(m_t, S)] \tag{3.2}$$

While not ideal, the advantage is that the provider has no motivation to pick the wrong dataset in the hopes of having a lot of value left over. Instead, the smart contract may give any leftover money to a specified account (like a charity), but it could be tricked. In conclusion, the supplier must assume that the entire bounty will be lost. To ensure that only trustworthy providers are motivated to join, it will only participate if the profit from the trained model justifies this expense.

3.3 Deposit, Refund, and Take: Self-Assessment

It would be ideal if those who provide inaccurate data were subject to a fee or punishment. Having additional participants check the data, as is usual in traditional crowdsourcing methods, is one way to discover if the data is bad. It is challenging to enforce a fine through a smart contract at a later date. This idea mandates a deposit when giving data in order to facilitate fines.

This IM [4] is a substitute where one need not rely on a kind agent to submit a test set as previously mentioned. Instead, one may rely on the data contributors to remunerate and indirectly validate one another. We suggest leveraging the deployed model, h, to validate new contributions as a stand-in for data verification. The restriction is that the model must have already been trained and must, in general, have already correctly classified samples with a level of accuracy that is considered to be acceptable. The proposal's salient features are as follows:

- *Deploy:* a model, m, already trained with some data
- *Deposit:*A deposit, c, is also necessary for each data contribution that includes data x and label y. Each contribution's data and meta-data are kept in the data handler.
- *Refund:*If the current model, h, agrees with the classification that was initially submitted after a period of time t, i.e., if m(x) == y, then the contributor may request a refund of their whole deposit, c.
- *Take:*Finding a data point (x, y) for which m(x) $\neq$ y allows a contributor who had already had their data vetted in the Refund stage to ask to receive a portion of the deposit (s), which was initially supplied when (x, y) was submitted.

If the sample provided, (x, y), is false or invalid, then additional contributors should provide (x, y′), where y′ is the true or at the very least commonly accepted label for x and $y' \neq y$, within the time limit specified. Similar to how one typically anticipates that incorrect edits to well-known Wikipedia articles will be promptly fixed.

1. *Time to Wait for Refund:* The contract's author must decide [9], t, how long donors must wait before requesting a refund of their money. As a general rule, we recommend selecting t so that other authors will have enough time to submit revisions with alternate labels if they disagree with the data. For instance, t equals one week. To train a model for a new use case, models that are not highly sensitive may need to wait until enough examples have been delivered.

   Very delicate models might allow dishonest contributors to request refunds for "poor" data before another contributor has a chance to "fix" their flawed input. Such models ought to demand deposits that are big enough to deter spikes in fraudulent data submissions. Before setting t, further precautions must be taken, and studies must be conducted. t need not always remain constant. It might be influenced by the sample of data provided, the quantity of data submitted, or how confidently the model fits the data point. In other words, if the model includes a measure of the likelihood that the submission (x, y) is correct, P(m(x)=y), then it can be used to lower t because it is unlikely to be modified in the future.

$$t \propto \frac{1}{P(m(x) = y)} \tag{3.3}$$

2. *Varying Deposit:* Requiring a deposit has a few goals:

   - Add value to the system so that after others have submitted accurate data, they can profit.
   - Prevent the model from undergoing too many alterations.
   - Lessen spam (incorrect or invalid data). To attain these objectives, we suggest

$$s \propto \frac{1}{time\ since\ last\ update} \tag{3.4}$$

   In other words, it is expensive for contributors to send several updates quickly. As a result, users of the models' prediction function ought to have a more consistent experience. Consider, for instance, a personal assistant in one's house that responds inappropriately to a verbal command that is repeated frequently throughout the day, such as "Play the news."

3. *Taking Another's Deposit:*To take a portion of the deposit from the original contributor, p., a contributor reporting "poor" data must follow specific rules that we establish. It should be noted that the data handler or emitted events may contain contributed data or meta-data about it.

   First some definitions:

- Let f(p_f, s) represent the payment made to contributor reporter c_f for submitting data (x, y) with deposit f.
- Let n(p) represent how many data samples contributor c was reimbursed for (assumed good data).

Guildelines:

- The existing model does not agree with the statement that $m(x) \neq y$. So we assume that the data is "bad."
- If n(($p_f > 0$) , the reporter ought to have already received their data back. This makes it mandatory for them to have, ideally, already submitted "excellent" data before attempting to take advantage of the system.
- The reporter is not permitted to be the original contributor ($p_f \neq p$). If not, contributors may easily try to get their deposits refunded for "bad" data.
- The reward should be shared amongst “good” contributors.

$$f(p_f, s) \propto s \propto \frac{n(p_f)}{\sum_{all} n(p)} \tag{3.5}$$

- The reward ought to be at least some minimal amount to cover potential transaction costs, according to the formula

$$f(p_f, s)\, f > \epsilon > 0 \tag{3.6}$$

- The remaining deposit that can be reclaimed, $s_f \leq s$, must be tracked by the data handler.

$$s_f \longleftarrow s_f - f(p_f, s) \tag{3.7}$$

$$f(p_f, s) \leq s_f \leq s \tag{3.8}$$

4. *Biasing the Model:* Contributors may be enticed by the idea only to provide data that the model already predicts (m(x) = y) at submission time and then hope that the model still predicts at refund time. Due to this, the model might be biased towards information it already "knows." Contributors still pay a small price to deposit money and get their refund because they typically have to pay a transaction fee [7]. The designer might consider handling duplicate or similar data while selecting the model and training approach. Even too many replication or similar data may cause the IM to reject it.

5. *Preventing Lock-ups:*We go over methods to prevent money from becoming "locked-up" or "trapped inside" the smart contract in this part. Contributors may fail to obtain their reimbursements or choose not to withdraw their share of the deposit,

leaving money "trapped inside" the contract. We add two new settings to prevent this:

- t_p :The period the author must wait before claiming the total refund (s_f) for a particular contribution. Where $t_p > t$, since there's a chance they could get to keep a sizable chunk of d, this encourages developers to use a model. Contracts may want to specify that this time is significantly longer than the time for trying a refund to offer contributors more time to receive their deposits back and prevent the creator from taking an excessive amount ($t_p >> t$).
- t_a : The period that somebody must wait before taking the total balance of the refund (d_r). If the creator forgets to subtract the "stuck" value from the contract, then $t_a \geq t_p > t$. There may be more variations of these as well, such as a value, t_d, for data contributors with refunds ($n(p) > 0$), where $t_a \geq t_s \geq t_p$.

6. *Experiment:*We created simulations to test various parameters for models and incentive mechanisms. We first used 8% of the IMDB reviews training dataset introduced in for one simulation to train a Perceptron model . The presence of the 1000 most common terms in the dataset is one of the model's 1000 binary features. Still, these contributors represent several contributors who gradually provided the remaining 92% of the training data. With our source code, you may find information about the simulation.

3.4 Floating Point Numbers

Because Ethereum does not allow floating point numbers, we use integers as data representations in our examples. When a floating point number is anticipated, we enter an integer that has previously been multiplied by a certain amount, such as 10^9 (9 decimal places of precision). Every action is taken with this transformation in mind.

3.5 Inversion of Control

Our examples apply the Inversion of Control principle [9], which favours composition over inheritance, to promote clarity and ease development.According to , inheritance-based smart contracts on Ethereum can be less expensive than those that use composition. There are certain fine points to take into account, such as ownership and advertisement of included contracts.

Since the model will function as a separate contract, its updating mechanism must be made available to the public. The model's update method can only be called by the model's owner (the CollaborativeTrainer), who is also in charge of utilising the incentive mechanism.

CHAPTER 4

Experiments and results

The suggested system is modular, allowing for many models or incentive mechanisms that may be effortlessly switched between; however, some Incentive Mechanisms may function better for specific models than others, and vice versa. These models are suitable for implementation on Proof-of-Work (PoW) blockchains , such as the current public Ethereum blockchain [5], because they can be updated effectively with one sample at a time. The first is a Naive Bayes classifier since it may be used to solve numerous different kinds of issues. The nearest centroid classifier is then used. A single-layer Perceptron model is the last one.

We assessed the models using three datasets [5] selected to illustrate issues that would profit from cooperative scenarios. Many contributors could enhance a model to produce a shared public resource. The situations included predicting a sport using Endomondo user activity, analyzing the sentiment of movie reviews from IMDB , and assessing whether a news piece is fake. In each of these cases, users gain from directly influencing the development of a model they frequently use and from not having to rely on a centralized authority to host and manage the model. Since these costs might be substantial for the open Ethereum blockchain, transaction fees (also known as gas) for each operation were also contrasted.

Our research mostly utilizes supervised classifiers since they can be applied to various tasks and are simple to assess using test data. We first suggest leveraging the work in the incremental learning field by adopting models capable of updating well with just one sample to minimize transaction costs. Most public blockchains rely on transaction expenses, or "gas," as it is known in Ethereum, to cover the compute costs associated with executing a smart contract.

4.1 Naive Bayes

First, a Naive Bayes classifier is used in the model since it can be used for many issues. The Naive Bayes classifier works quickly for updating and predicting because it considers that each feature in the model is independent.

- Find the class with the most likely class for the data.
- x:list of features
- c:the classification/label
- Update: increment various counts

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \tag{4.1}$$

4.2 Nearest Centroid

The nearest centroid classifier assigns new points the label of the centroid that they are closest to by computing the average point (or centroid) of all the points in a class .

- Find the most similar point to your data
- Easy to update moving average:

$$avg(t+1) = \frac{x + n.avg(t)}{t+1} \tag{4.2}$$

- Enforce normalized: no one can move a centroid too much
- Encode “off-chain” using a known encoder: tested with 512 dimensions

4.3 Perceptron

For binary classification, a single-layer perceptron model [8]is an excellent linear model. We assess this approach since it can be applied to dense and sparse data, including text. Perceptron’s update mechanism only modifies the weights if the sample is currently incorrectly classified by the model.The model may be efficiently updated by simply adding or removing values for the sample’s features with the model’s weights based on the sample’s label.

- Only update when expected class predicted class
- Predicted classification: $\hat{y} = w(t).x + b$

- Easy to update: w(t+1) = w(t) + r.(y- $\hat{y} = w(t).x + b$).x
  - x : data (vector/list of numbers)
  - y expected classification/label (number)
  - w(t): weights at time t vector/list of numbers)
  - r: learning rate (number)
  - b: bias/offset (number)

4.4 Datasets

Two datasets were selected to illustrate issues that would profit from cooperative situations in which numerous contributors could enhance a model to provide a shared public resource. In every case, the users gain direct control over the model they regularly use and not depend on a centralized authority to host and control the model.

4.4.1 Fake News Detection

The aim is to judge whether or not the news story is reliable, given the language of the article . Only the top 1000 count in the training set are taken into account when we convert each text into a sparse representation utilizing the word frequency of the bigrams. While tackling false news detection [5] is probably too complex for straightforward models, decentralization would tremendously improve a detector by preventing it from being influenced by a centralized authority.

4.4.2 IMDB Movie Review Sentiment Analysis

The goal of the dataset [5] for sentiment analysis, which consists of 25,000 IMDB movie reviews, is to determine if the English text of a review is positive or negative. We only used the top 1000 most frequent words in the dataset for our word-based characteristics. Because of its size and popularity, this particular sentiment analysis [13] dataset was chosen for this study. Despite the fact that this dataset primarily consists of movie reviews, a sentiment analysis model developed through collaboration can be used in a variety of contexts, such as a system to track social media posts.

4.4.2.1 Initial Model

The initial deployed model's degree of training is frequently helpful. For instance, if the model already has some level of usefulness, it is more likely to be used for inference by users, who are more inclined to produce and supply data to develop the model. Additionally, having an initial degree of accuracy enables more significant validation of provided data for sure of our incentive mechanisms, but not all of them. We note that while this can be useful, the model does not always need to be highly precise or pre-trained.

4.4.2.2 Limitations

Our examples usually concentrate on applications related to managing little input that may be easily compressed, such as text, due to limitations like the cost of RAM in the Ethereum network. Word dictionaries and popular shared encoders like the Universal Sentence Encoder in can be used to quickly and easily compress text. When using Ethereum, the cost of storing and updating complicated models, such as deep neural networks that can understand images, may be costly. The cost of uploading just the raw images would be high. For instance, according to, at a moderate gas price of 4gwei, downloading the full MNIST collection of handwritten digits would cost about 275 ether.

4.5 Fake News Detection

Table 4.1: For the Fake News dataset, the costs of Ethereum gas for each model.

Action	Naive Bayes Sparse	Nearest Centroid Sparse	Perceptron Code
Deployment	55,511,446	67,139,037	30,967,145(46.20 $)
Update	281,447	356,345	263,517 (0.39 $)
Refund	172,216	176,797	138,028 (0.21 $)
Reward	136,800	141,253	102,484 (0.15 $)

Each model allowed the "good" agent to make money while the "bad" agent suffered a financial loss.The Naive Bayes model outperformed the Perceptron model's baseline accuracy, which has the highest accuracy.

According to **Table 4.1**, the Perceptron model has the lowest gas cost. The Naive Bayes and Sparse Nearest Centroid models have significantly higher deployment costs

because virtually every 1000 features needed to be configured twice (once for each class). The Sparse Nearest Centroid Classifier did not need prediction (which happens in Refund and Reward) to go through each dimension because the distance to each centroid can be calculated by storing the magnitude of each centroid and using the sparse input data to find the difference from the extent just for the few features in the spare input.
When we update each feature, we keep extra data (mostly the new denominator; upgrading the Sparse Nearest Centroid Classifier does not necessitate updating every feature). The Sparse Nearest Centroid Classifier does not require updating every feature since we keep some extra information (mostly the new denominator) when we update each feature. Instead, we compute and use the appropriate denominator at prediction time.

4.6 IMDB Movie Review Sentiment Analysis

Table 4.2: shows the prices of Ethereum gas for each model in the IMDB Movie Review Sentiment Analysis dataset.

Action	Naive Bayes Sparse	Nearest Centroid Sparse	Perceptron Code
Deployment	55,423,682	67,136,669	30,875,193(46.07$)
Update	332,636	422,476	332,927(0.5 $)
Refund	189,954	196,375	145,601(0.22 $)
Reward	154,538	160,831	110,157 (0.16 $)

The "bad" agent loses most or all of the initial investment, whereas all "good" agents can make money. With this data, all models maintained their accuracy, with the Naive Bayes model performing the best. This is probably due to a large number of features. For the Update technique with the lowest gas cost, the Naive Bayes model narrowly defeats the Perceptron model. **Table 4.2** displays the gas prices for all acts. Because we may skip many dimensions, the Sparse Nearest Centroid Classifier's Update cost was cheap, just like it was for the Fake News dataset. Since each feature needed to be configured for each of the two classes, the deployment costs for the Sparse Nearest Centroid Classifier and Naive Bayes models were significantly greater.

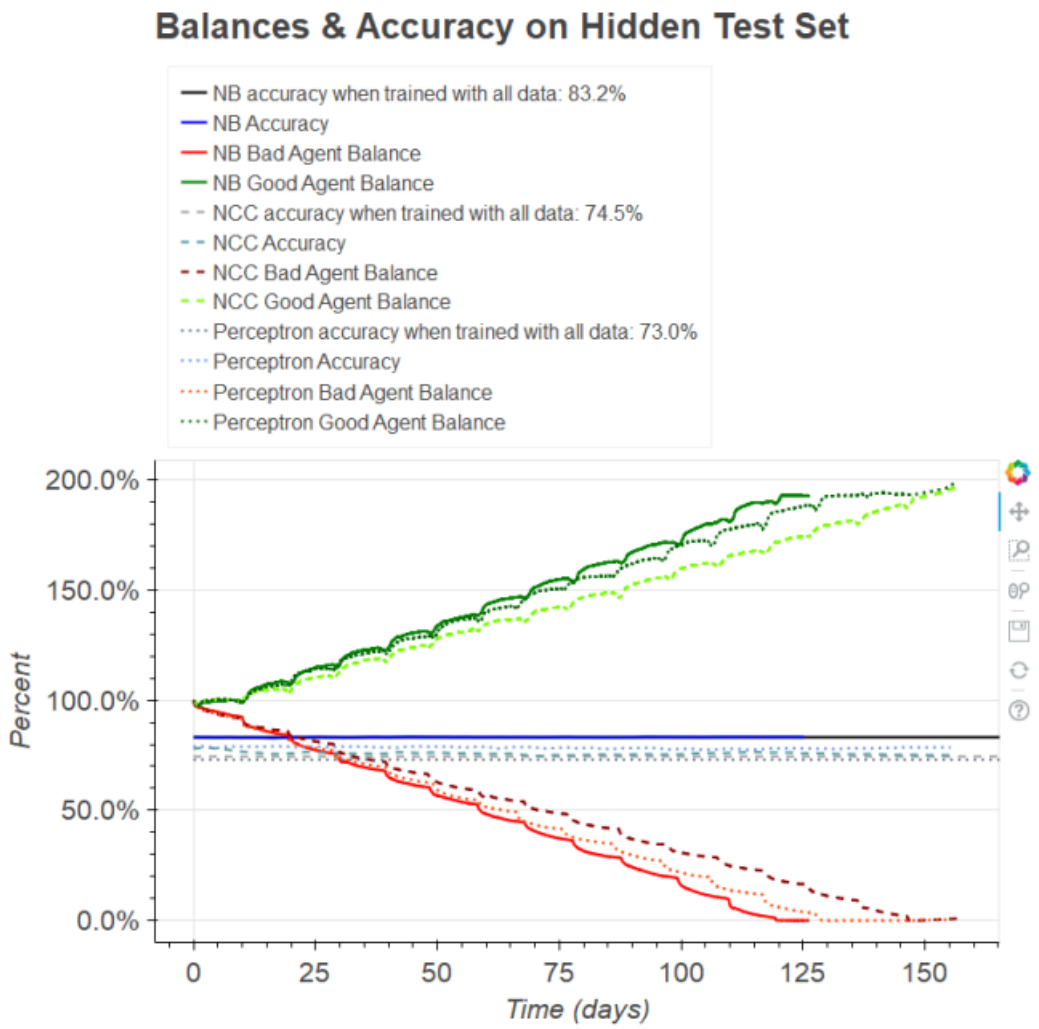


Figure 4.1: Graph simulations of IMDB Movie Review Sentiment Analysis dataset.

4.7 Conclusion

The Perceptron model was consistently the most affordable to utilize across all studies. This is clear from the simulation as shown in **Fig.4.1**. This was primarily due to the model's smaller size compared to the other two models, which had to store effectively twice as much data as the perceptron does for each class. Although it cost a lot to implement each model, this was a one-time expense. This expense is far lower than what it would cost to run a web service using the concept for a number of months.

Most models kept their accuracy, with the volatile Perceptron for the Activity Prediction dataset being the exception. The model can be forked from a previous point when its accuracy on a concealed test set is higher, even if the model becomes damaged with false data. Additionally, it can be retrained using data deemed "good" during deployment. Consumers must know the model's accuracy on a few concealed test sets.

Users can keep their own private test sets or use a service offered by an organization that would publish a model rating based on the test sets they have.

Due to the "good" agent's previous diligence and the fact that we established a standard wait period of 9 days before either agent could claim the remaining deposit for a data donation, the balance graphs appeared to be relatively consistent between experiments. Since they only provided accurate data when they believed the model to be trustworthy, the "good" agent was able to recover their deposits and profit from disclosing many contributions from the "bad agent." A subset of the assistance from the "good" agent can be successfully reported as bad when the "bad" agent is able to corrupt since the model would disagree with those contributions.Since the "bad" agent has fewer "confirmed" donations than the "good" agent, they are unable to claim the majority of these deposits when reporting the contribution. This leaves a balance for which either agency must wait nine days before taking the total remaining deposit. As a result, the balance plots display periodic patterns every nine days. Because there is always data for which either agent cannot collect the deposit after simply the initial refund wait time of 1 day, the pattern persists throughout the simulation.

CHAPTER 5

Discussions and conclusion

We have developed a customizable framework for training and data gathering on a blockchain using numerous fundamental incentive mechanisms and various existing machine learning models for gradational learning. The best-case scenarios have a variety of facts with widely accepted classifications. This methodology is primarily intended for models that can be updated effectively at the moment, although we anticipate further research on scaling to more complicated models.

5.1 Advantages Of Blockchain

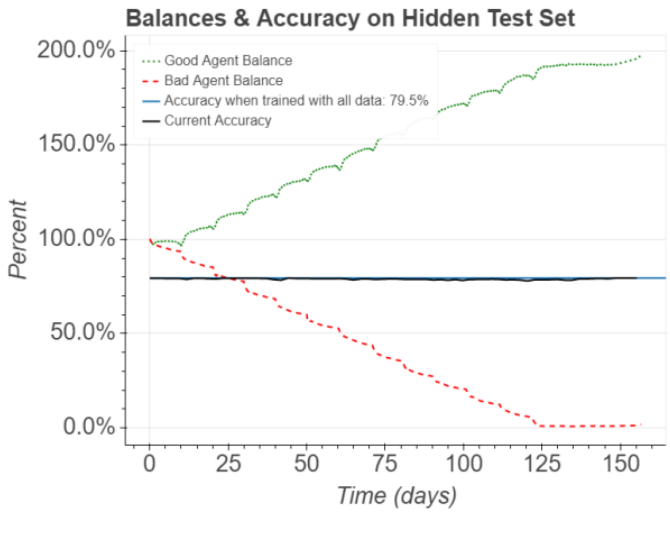


Figure 5.1: Calculate the model precision in a simulation where the "Bad Agent" rival is willing to deposit nearly the money as the "Good Agent," a trustworthy participant.

As shown in **Fig.5.1**, we can see a rise in accuracy and the Balance of the "Good agent" in the network and quite the opposite effect for the "Bad Agent."

5.1.1 Public, Persistent & Decentralized

The model is a available to everyone in a public blockchain [15]. Essentially, one can trust that the model will remain and can be accessed via a large number of nodes.

5.1.2 Versioning

Simply updating your node's version to the same block where your application is satisfied with the model and maybe assessing performance in relation to one's secret test set makes it simple to go back to a prior version of the blockchain.

5.1.3 Models Transform Over Time

Blockchain-based programs enable models to adapt to changing conditions, such as the ability to recognize new words.

5.1.4 Transparency Trust

A machine learning or data collection service generally uses a REST API or similar approach that enables users to interface with code on other servers. There is no mechanism for users to verify that the source code running on those servers is accurate, even though they can submit data or request it from the servers. An Ethereum smart contract's complete source code is accessible to everyone, and it is feasible to foresee the outcome of a particular execution. The model will be updated, and data contributors may rely on their efforts being acknowledged.

5.2 Limitations

5.2.1 Introducing Bad Data

A wealthy and tenacious actor can ruin a model that has been implemented by providing false or absurd training data.

In response, the incentive system should make it expensive and disadvantageous to provide false data. Additionally, many blockchains charge transaction fees, which makes submitting a lot of data quite expensive. Users can go back to a previous version of a model even if it becomes corrupt because it is stored on a blockchain. Additionally, analysis can be performed on already submitted "good" data.

5.2.2 Obscure Data

Users should decide the model carefully, Incentive Mechanism, and the potential consequences of entering ambiguous data when implementing this architecture. For instance, the sentiment "The play was okay" might be appropriate if the label options were "joy" or "sorrow." Obscure Information is always a problem when crowdsourcing because contributors could lose money, but it is particularly difficult in this situation. Contributors should keep their data as clear as possible.

5.2.3 Overwhelming the Network

Due to network congestion, some apps that rely on public blockchains have experienced reliability concerns. When data is being added , which necessitates initiating new blockchain transactions, this can be problematic for the framework.Running inference requires extensive reading, which can be performed locally using the most recent model version and is often inexpensive, so it shouldn't be impacted.

5.2.4 Requiring a Deposit

The common practice of requesting a deposit for blockchain applications may be unfamiliar to many new participants. Most well-known blockchains, including Bitcoin and Ethereum , already charge small fees to conduct any transaction. These fees can sometimes be concealed from customersA third party can hide the deposit cost by offering their interface to a public system and independently evaluating data submissions. This external party may believe that their methods and techniques for validating data are superior. After that, they might reward clients in accordance with their strategy, which might not even involve financial rewards.

5.3 Future scope

The offered framework can be customised and expanded upon in several ways.

5.3.1 Models

More research is required on the models that will operate effectively within this framework.

1. Unsupervised Models: Because we concentrate on verifying data with labels, the examples are given primarily employ supervised classifiers. Unsupervised models can use test data and incentive systems to verify data contributions. Examples include:

    - Clustering:Outlier identification can be aided greatly by the development of clustering models.
    - Generative models like GANs and autoencoders. For instance, a model that tries to produce text or create images could be developed.

2. Complex Models:More research is required to find out how to pre-calculate as much as possible off-chain and carry out necessary processes in a smart contract. Techniques like fine-tuning or off-chain training, as recommended for DeepChain, can be beneficial. A simple neural network, one layer, and a typical encoder can be used to fine-tune the encoded result on-chain (shared in more traditional methods such as via online APIs or publicly uploading source code to a website).

    Although the encoder should be static, significant change is possible if the input to the fine-tuning component is sufficiently similar to previously encountered training data. A system like Provable can even use complex models via an API through a smart contract. The idea of this approach is not served by hiding the model behind an API, which would make the model potentially private. This framework may also create sophisticated models by integrating many "layers" of smart contracts.

3. Recovering iniquitous Models:It is possible to repair a model damaged by inaccurate data.A dataset can be improved using various techniques once it has been gathered. A new model can be trained using the cleaned dataset. Those who coordinate cleaning efforts and implement this new model in their production system could keep it a secret. The collaborative training procedure could be restarted using the model to gather further training data.

5.3.2 Incentive Mechanisms

Further research, analysis, and experimentation in this area are required to highlight the types of models with which each incentive mechanism works well with. Third-party providers that develop services based on this suggested structure may be able to conceal from end users the incentive mechanisms imposed by the smart contract. If consumers of these services don't want to deal with these smart contracts, they can still receive rewards for validating data contributions on their own platforms.

5.3.3 Privacy

Contributors might not want their information to be shared on a public blockchain. We first suggest using this framework solely for data that can safely be made public. For instance, personal assistant requests, such as "What day is tomorrow?" don't contain personal information.In the future, transmitting only model updates to the smart contract may be possible rather than actual information.

CHAPTER 6

Results

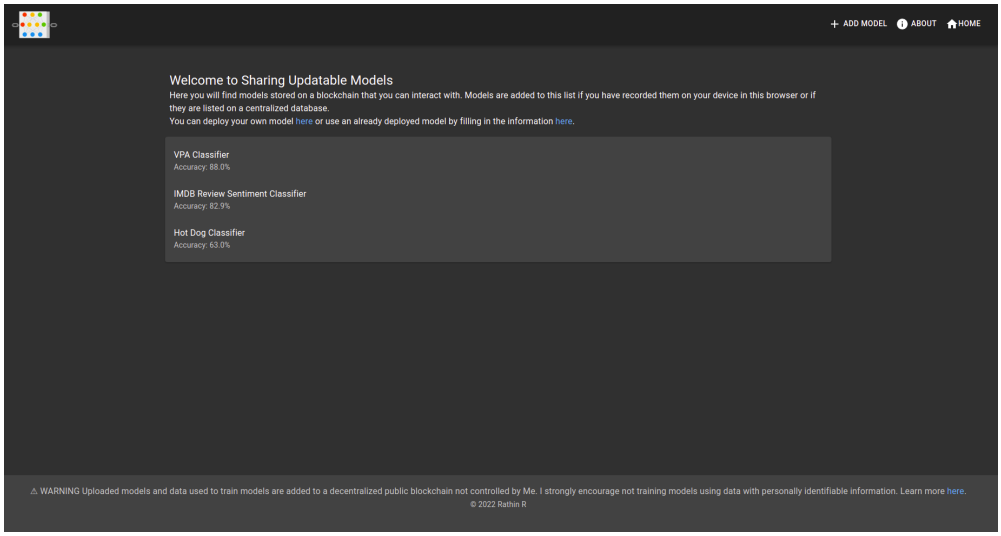


Figure 6.1: Shows available models that have been trained in the blockchain network

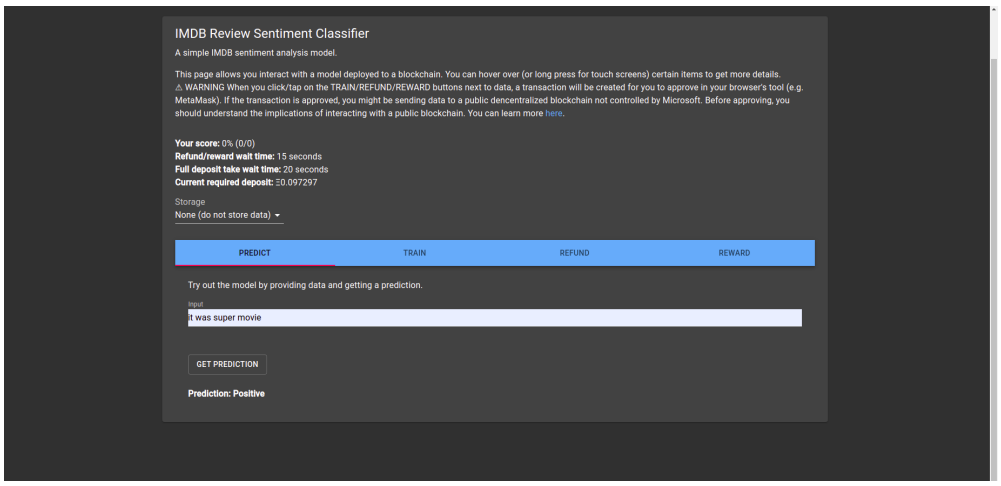


Figure 6.2: Get prediction(positive or negative) for your movie review from the model.

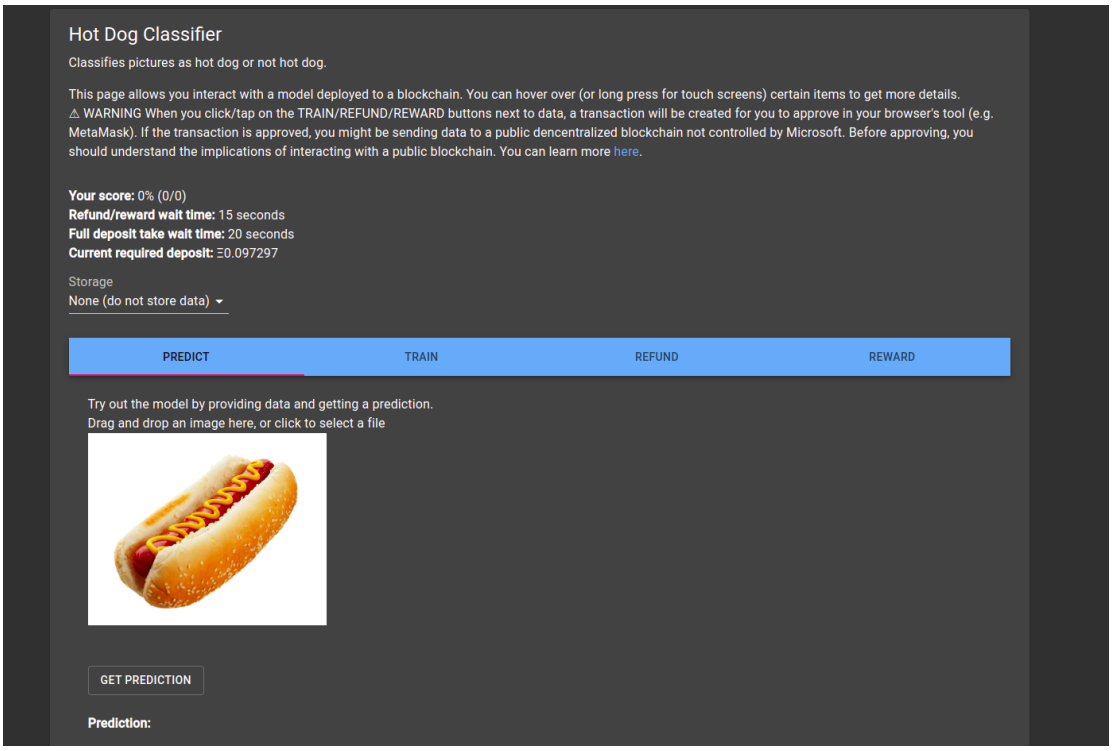


Figure 6.3: Train models by adding data(either "good" or "bad") by staking a deposit and get rewarded for every good data and get a refund if the data increases the accuracy of the model. At the same time get penalized for every "bad" data added.

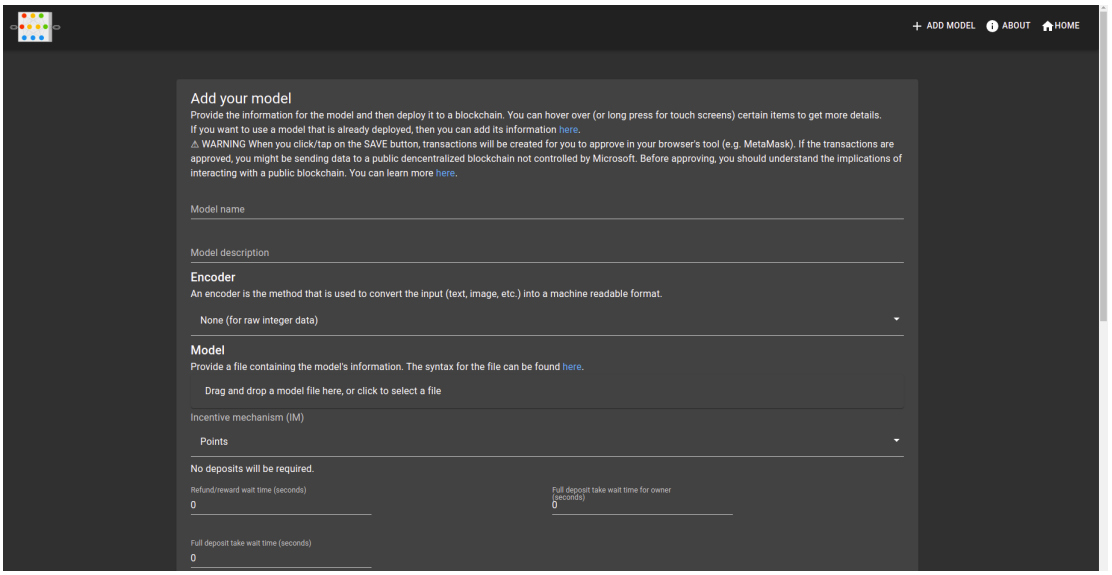


Figure 6.4: Interface to train models as per the user's requirements. All the information regarding the model will be stored in the blockchain in the form of smart contracts.

REFERENCES

[1] A. Marathe, K. Narayanan, A. Gupta, and M. Pr, "DInEMMo: Decentralized incentivization for enterprise marketplace models," 2018 IEEE 25th International Conference on High Performance Computing Workshops (HiPCW), pp. 95–100, 2018.

[2] A. B. Kurtulmus and K. Daniel, "Trustless machine learning contracts; evaluating and exchanging machine learning models on the ethereum blockchain," 2018. [Online]. Available: https: //algorithmia.com/research/ml-models-on-blockchain

[3] Li, M., Weng, J., Yang, A., Lu, W., Zhang, Y., Hou, L., Liu, J., Xiang, Y., Deng, R.H.: Crowdbc: A blockchain-based decentralized framework for crowdsourcing. IEEE Transactions on Parallel and Distributed Systems 30(6) (June 2019) 1251–1266

[4] Harris, Justin Waggoner, Bo. (2019). Decentralized and Collaborative AI on Blockchain. 368-375. 10.1109/Blockchain.2019.00057.

[5] UTK Machine Learning Club: Fake News — Kaggle. https://www.kaggle.com/c/fake-news/overview (2020) [Online; accessed 07-January-2020].

[6] J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in Proceedings of the Fifth AAAI National Conference on Artificial Intelligence, ser. AAAI'86. AAAI Press, 1986, pp. 496–501. [Online]. Available: http://dl.acm.org/citation.cfm?id=2887770.2887853

[7] V. Buterin, "A next generation smart contract decentralized application platform," 2015.

[8] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review, vol. 65, no. 6, p. 386, 1958.

[9] Wikipedia contributors, "Inversion of control — Wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Inversion of controloldid=885334776, 2019, [Online; accessed 27-March-2019].

[10] J.S. Weng, J. Weng, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," IACR Cryptology ePrint Archive, vol. 2018, p. 679, 2018.

[11] J. D. Abernethy and R. M. Frongillo, "A collaborative mechanism for crowdsourcing prediction problems," in Advances in Neural Information Processing Systems 25, ser. NeurIPS '11, 2011, pp. 2600–2608.

[12] D. Cer, Y. Yang, S. yi Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y.-H. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," 2018.

[13] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[14] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," White Paper, [Online]. Available: https://bitcoin.org/bitcoin.pdf, 2008.

[15] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in Proceedings of ICML Workshop on Unsupervised and Transfer Learning, 2012, pp. 17–36.

[16] B. Waggoner, R. Frongillo, and J. D. Abernethy, "A market framework for eliciting private data," in Advances in Neural Information Processing Systems 28, ser. NeurIPS '15, 2015, pp. 3492–3500.

[17] F. Wessling, C. Ehmke, M. Hesenius, and V. Gruhn, "How Much Blockchain Do You Need? Towards a Concept for Building Hybrid DApp Architectures", 2018 ACM/IEEE 1st International Workshop on Emerging Trends in Software Engineering for Blockchain, Sweden, 2018