# Predict Chain

Matthew Pisano, Connor Patterson, William Hawkins

Spring 2023

## 1 Project Description

PredictChain is a marketplace for predictive AI models. Users are able to upload datasets for training predictive models, request that train models be trained on any previously uploaded datasets, or submit queries to those trained models. These various models will be operated by a central node or nodes with computing resources available. A variety of models will be made available, ranging from cheap, fast, and simple to more expensive, slower, and more powerful. This will allow for a large variety of predictive abilities for both simple and complex patterns. All the past predictions form these models will be stored on the blockchain for public viewing.

### 1.1 Problem Solved

*Clearly state the problem your project solves*

PredictChain helps to solve one of the main issues that involve AI models today: accessibility. Our project fulfils this need in two ways. Oftentimes, individuals or groups poses data that they would wish to be used in predictive analysis. However, these people may not have access to the compute capacity to train predictive models on this data. Additionally, yet other people have neither access to predictive training data, nor do they have access to computational resources. PredictChain solved both of these problems simultaneously.

When users upload their datasets, they allow model to be trained on those datasets. Higher quality datasets will produce higher quality models. When users submit parameters for training, they allow the model that their parameters produce to be used publicly. Both of these users are rewarded for their work when a model is queried, and it produces a correct prediction. This encourages users to participate in contributing the resources needed for good predictions, while leaving a public record for other users to view.

### 1.2 Background

*Why is this an important and relevant problem within the context of AI and Blockchain*

### 1.3 Use Case/ Motivating User Story

*Use case or motivating user story detailing how someone would use the system you have built*

## 2 Implementation Details

*Provide details of how you implemented your solution. Please add as many diagrams as necessary and explain the diagrams you added in the text. Any external libraries used and rationale for using them Links to all the resources produced (GitHub source code, demo link, link to the recorded video of the demo, etc.)*

The structure of PredictChain is primarily broken up into two parts: the client and the oracle. Both of these parts interact with each other through the blockchain.

## 2.1 The Client

The client serves as a middleman between the front end user interface and the blockchain. It is run as a server, serving UI content to the user, taking in requests from the UI, and parsing those requests into a form suitable for both the blockchain and for the oracle. Additionally, the client constantly polls for updates coming from the oracle, through the blockchain. These updates are queued and sent to the front end upon request. This allows the user to both interact with the blockchain and to see important updates that come from it.

## 2.2 The Oracle

The oracle accomplishes the majority of the other tasks that this project requires. It constantly polls for updates coming from the client, through the blockchain. Upon receiving these updates, it begins the execution of one of its main operations. These are:

- Downloading a user-specified dataset and saving it
- Training one of the raw models based on user inputted parameters
- Querying one of the trained models on user inputted data and comparing it to the real-world result

After each of these operations, the oracle sends out several transactions. These can be either rewards to contributors of a model or confirmations/results of the operation that has been performed.

When working with user-submitted datasets, the oracle uses a handler to manage the operations performed on that dataset. The handler can save datasets to a specified environment, load datasets from a specified environment, parse that dataset as a pandas dataframe, and split the dataset by the values of one of its attributes. The environments that the handler's recognize are *local* and *ipfs*. When using either of these environments, the handler abstracts away the complexities of working with either of them into a unified interface.

When working with user-trained models, the oracle uses a similar, common interface. This interface can create the model architecture, train the model on a selected dataset, query the trained model, evaluate its performance, save the model, and load it back from a specified environment. When creating and training a model, the interface chooses among a group of archetype or template models. These models can be a:

- Multi-layered perceptron neural network
- Recurrent neural network
- Long short-term memory neural networks
- Gated recurrent unit neural networks

Each of these models has a *model_complexity* attribute. This is a simple float value, designed to give users a general idea of how performant a model can be once trained and serves as a method of calculating the cost of using or training that given model. The attribute itself is calculated using the size of the network and a linter multiplier to account for more complex model architectures. For models like GRUs or LSTMs, the complexity is higher as they are mode complex, and often better performing models. For models like MLPs, the complexity is lower. This gives the desired effect of faster, simpler models being cheaper than the slower, more complex models without any heavy calculations. The interface abstracts most of the complexities of training, querying, and evaluating these models. The only difference between them is the inclusion of several optional parameters.

## 2.3 The Blockchain

In PredictChain, the blockchain serves as both a records keeper and a messenger between the client and the oracle. This is accomplished by using transactions as a form of direct communication. With every transaction sent is a note. This note is a json-encoded string (encoded in base64) that communicate information about

the operation that the transaction is requesting be performed and arguments relevant to that operation. The operations in the note are represented by a series of op codes. These codes are abbreviations of the operation name enclosed in angle brackets, for example *¡QUERY_MODEL¿*. The arguments to these operations are represented as a named dictionary, with each key being the name of the argument and each value being the argument itself. This named strategy allows the program to be very flexible without worrying about the exact ordering of the arguments. Blockchain is quite useful in its role due to its immutability and its transparency. Using a blockchain means that all requests are permanently stored and public, so other users can see what type of models are useful for specific datasets and what results those models have produced.

# 3 Evaluation

*How do you know your solution works? What tests have you performed, and what results have you obtained? If you have acquired any users to try out your system, summarize their reactions and feedback*

# 4 Conclusion

## 4.1 Summary

*A short summary of the project*

## 4.2 Limitations

*Limitations of the work*

## 4.3 Future Work

*Potential future work*
[1]

# References

[1] author. *title*.