Making Smart Contracts Predict and Scale

Syed Badruddoja, Ram Dantu, Yanyan He, Mark Thompson, Abiola Salau, Kritagya Upadhyay
Dept. of Computer Science & Engineering
University of North Texas
Denton, TX, 76207, USA
E-mail: syedbadruddoja@my.unt.edu, ram.dantu@unt.edu, yanyan.he@unt.edu,
mark.thompson2@unt.edu, abiolasalau@my.unt.edu, kritagyaupadhyay@my.unt.edu

2022 Fourth International Conference on Blockchain Computing and Applications (BCCA) | 978-1-6654-9958-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/BCCA55292.2022.9922480

***Abstract*—The machine learning algorithms can predict the events based on the trained models and datasets. However, a reliable prediction requires the model to be trusted and tamper-resistant. Blockchain technology provides trusted output with consensus-based transactions and an immutable distributed ledger. The machine learning algorithms can be trained on blockchain smart contracts to produce trusted models for reliable prediction. But most smart contracts in the blockchain do not support floating-point data type, limiting computations for classification, which can affect the prediction accuracy. In this work, we propose a novel method to produce floating-point equivalent probability estimation to classify labels on-chain with a Naive Bayes algorithm. We derive a mathematical model with Taylor series expansion to compute the ratio of the posterior probability of classes to classify labels using integers. Moreover, we implemented our solution in Ethereum blockchain smart-contract with the Solidity programming language, where we achieved a prediction accuracy comparable to the scikit-learn library in Python. Our derived method is platform-agnostic and can be supported in any blockchain network. Furthermore, machine learning and deep-learning algorithms can borrow the derived method.**

***Index Terms*—Blockchain, DApp, Smart Contract, Machine Learning, Artificial Intelligence, Naive Bayes**

I. Introduction & Motivation

Trustable AI Model: The machine learning training models are prepared with well-known algorithms proven to yield high accuracy. One of the crucial problems in recent development involves the trust of data and model [1], [2]. If the data and model of the machine learning process are altered, then the artificially intelligent applications fail to produce good results. A machine learning model and training data can be trusted by securing the training process by ensuring integrity. A trusted model that has not been tampered with can produce a trusted prediction.

Smart Contract Limitations: Most of the supervised machine learning algorithms produce floating-point output for classifying labels. For example, k-nearest neighbor [3] distance computation has square root values, naive Bayes yields probability values [4], and decision-trees [5] use entropy with information gain where decimal numbers are inevitable. However, the lack of standardized libraries, signed exponent computation and floating-point data type in blockchain smart contracts has made it hard to develop intelligent applications with such complex computational requirements [6], [7]. Hence, the decentralized application(DApp) development to learn and predict is limited.

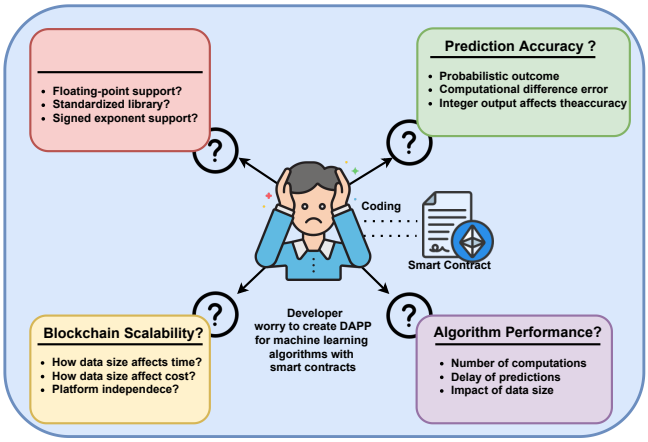

Fig. 1. DApp developers face challenges in developing smart contracts for machine learning algorithms with language limitation, scalability, prediction accuracy, and algorithm performance

Prediction Accuracy Of Learning Smart Contracts: Smart contracts [8] of blockchain technology execute functions with static rules to secure transactions that are recorded in the distributed ledger as a result of mining a block [6], [9]–[11]. Financial trades, reputation systems, legal contracts , and ownership validation are static transactions that do not require to learn updates or possess cognitive intelligence [12]–[18]. Machine learning algorithms require a dynamic update of learning models that can sustain the needs of training models at different times. Smart contracts must support such dynamic updates to learning model parameters and predict with reasonable accuracy. With the limitations of smart contracts discussed earlier, it is difficult to guarantee good prediction accuracy. Due to the lack of fixed-point computation support, the training model may not produce the correct prediction output due to computational differences.

Scalability of Blockchain: One of the limitations that restrict the use of blockchain is the scalability factor [19], [20]. With large computational requirements, the blockchain platform will not be able to withstand the number of computations, and single transactions may become very expensive. Furthermore, the block creation delay may also limit the support of the application that requires faster transaction output which is not guaranteed. For example, figure 1 shows some issues with smart contracts limiting a reliable AI prediction with smart contracts.

Blockchain	Consensus	Language	Float Support	Number of DApps
Ethereum	PoW (Expected Upgrade to PoS)	Solidity [26]	No	2970 [23]
Algorand	PoS	TEAL [27]	No	7 [25]
Cardano	PoS	Marlowe [28]	No	72 [22]
Polkadot (Ethereum Compatible)	PoS	Solidity [29]	No	50 [21]
NEO	DBFT-POS	C [30]	No	176 [24]
Binance	T-BFT	Solidity [26]	No	3494 [25]

TABLE I
FLOAT SUPPORT IN DIFFERENT SMART CONTRACTS AND BLOCKCHAIN PLATFORMS FOR POPULAR BLOCKCHAIN DAPPS

Algorithm Performance: Machine learning algorithms have different computational complexity per their training and prediction structure. The computational complexity of training a model can range from $O(nd)$ in the Naive Bayes algorithm to $O(n^2)$ in support vector machine models, where n is the number of samples and d is the dimension [31], [32]. With known limitations of blockchain smart contracts, it will be interesting to see how the machine learning algorithm performance will be affected when we develop smart contracts to learn and predict.

II. PROBLEM DEFINITION

Machine learning algorithms require a trusted platform to secure the training and prediction of classification tasks. Ethereum blockchain smart contracts can learn and make predictions, providing security with decentralized consensus. However, machine learning algorithms use floating-point computations to learn and predict. Due to a lack of support for floating-point calculations, signed integer exponents, and standardized libraries, the Ethereum blockchain cannot accurately provide computational output. Moreover, due to these limitations, the training accuracy of a model and prediction accuracy of new data may not be reliable. One such algorithm is Naive Bayes which requires probability computation with floating-point output. Naive Bayes uses Gaussian probability that has exponent computation and floating-point requirements. Furthermore, the performance of blockchain networks to produce such complex calculations can affect the scalability, computational costs, and prediction delays.

III. OUR CONTRIBUTION

- We have proposed a novel method to secure a machine learning algorithm using the Ethereum blockchain. Our solution performs training and prediction with smart contracts. **See figure 3**.
- We have derived a platform-agnostic numerical method to compute Gaussian probability based on Taylor series expansion inside the smart contract. **See section VI.**
- Our solution can help other AI algorithms compute relevant output using smart contracts. **See table III.**
- The prediction accuracy of the smart contracts with probability estimation is comparable to that of the scikit-learn python library prediction. **See figure 4**.
- We have shown that the cost of training and prediction of our model forms a linear relationship with the rising number of samples and features. **See figure 5.**
- The prediction time of the test dataset is given by a gamma probability distribution function, with most of the predictions occurring between 15-25 seconds. **See figure 7.**

IV. RELEVANT LITERATURE

Existing Applications: DeepBrainChain aimed to reduce the cost of computing involved in research and development [33] to share the computing load by decentralizing the work while protecting the privacy of data shared for training purposes and trained the model outside the smart contract. Artificial intelligence(AI) algorithm developers can participate openly in building AI-based blockchain applications. Cortex [34] Labs prepared a Cortex virtual machine ensuring capabilities of Ethereum virtual machines with a separate GPU processing engine for training data in deep neural networks. Cortex Labs also has its public chain and transaction wallets with Cortex coins for transactions. The platform takes the input of the image and feeds the parameters to choose an algorithm already stored off-chain. It allows the developers to be incentivized based on model performance. Danku-Algorithmia developed a Danku project [35] that allows anyone to post a dataset and ML model that will be evaluated and rewarded, ensuring ownership of models. MATRIX project [36] develops AI-based blockchain applications promising growth of intelligent blockchain applications. MATRIX has made auto-coding smart contracts one of the key aspects for developers to create applications without knowing the language of smart contracts.

Blockchain-AI Fusion: In decentralized and collaborative AI on blockchain [37], the perceptron algorithm is proposed to train the model and test for prediction outside the blockchain network and store the model for inference. This work aimed to find evil and good input in the machine learning model with some incentivization benefits. Blockchain provides automation features lacking in the ML algorithm, eventually improving performance [38]. Machine learning can take the help of blockchain to build a privacy-preserving model for its prediction technique [39]. Liu et al. [43] bring to light the advantage of collaboration between machine learning and blockchain technology that can benefit network and communication systems. The proposed idea that blockchain can facilitate training data and a sharing model for decentralized intelligence is highly feasible but hardly implemented. Machine learning applications can use blockchain in communication and networking systems to provide security, scalability, and privacy in intelligent smart contracts.

Solidity Restrictions: Table I provides a list of the popular blockchain platforms that lack floating-point data type support, and most use Solidity programming language for smart contract development. The current version of Solidity does not support *fixed-point numbers* that impede blockchain

Contribution	Cortex [34]	Danku [35]	Deepbrainchain [33]	Miscellaneous Works [36]–[39]	Our Solution
On-chain prediction	No	No	No	No	Yes
On-chain training	No	No	No	No	Yes
Platform-agnostic	No	No	No	No	Yes
Secure prediction	No	No	No	No	Yes
Update AI learning model	Yes	Yes	Yes	Yes	Yes

TABLE II
COMPARISON OF OUR PROPOSED SOLUTION WITH EXISTING BLOCKCHAIN APPLICATION INTEGRATING ARTIFICIAL INTELLIGENCE. CORTEX, DANKU, DEEPBRAINCHAIN, AND MISCELLANEOUS AI-BLOCKCHAIN PROJECTS DO NOT PERFORM ON-CHAIN TRAINING AND PREDICTION USING SMART CONTRACT

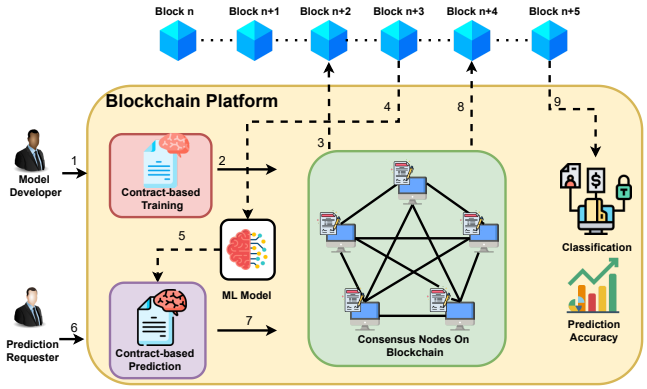

Fig. 2. The trusted prediction platform with AI and Blockchain integration with smart contracts to train and predict using a machine learning algorithm.

from developing any application that requires decimal number computations [40]. We can currently declare the decimal values in the Solidity language, but operations like division, square root, and logarithmic outputs are impossible. There is a lack of *standardized libraries* that application developers can efficiently utilize to create applications without spending much time on small computations or functions. Fixidity [41] and ABDK [42] are two libraries trying to implement fixed-point equivalent outcomes but pose challenges on how the converted values will impact the probability results. The ABDK library performs 64x64 fixed-point numbers that define the numerator as a 128-bit integer type with the denominator representing 2^{64} [42]. These libraries are helpful for simple mathematical calculations with low complexity but not for iterative computations that make the computations more convoluted and expensive for blockchain smart contracts.

Novelty Of Our Work: To the best of our knowledge, existing solutions do not address the problems with Solidity smart contracts, so they cannot train and predict inside a smart contract. Table II shows the comparison of contributions from different existing literature and the novelty of our solution.

V. METHODOLOGY

Research Approach: Designing a blockchain application to train a model and predict requires a machine-learning algorithm to be defined in smart contracts. Figure 2 shows a high-level design of smart contract-based training and prediction in a blockchain network. The smart contract will train the model on a blockchain network to record model parameters with a Naive Bayes algorithm. The prediction formula of Gaussian probability will classify objects based on the trained model. Our proposed solution is platform-agnostic and can be used in any blockchain smart contract.

Naive Bayes Algorithm: In supervised machine learning techniques, the Naive Bayes algorithm is one of the least complex algorithms for classification tasks. The training involves computations of means, variance and prior probability concerning each class and prediction involves simple Gaussian probability computations. Moreover, the training time complexity of the Naive Bayes algorithm is *O(n*d)* and prediction complexity is *O(c*d)* where n is the number of samples, d is the number of features, and c is the number of classes [44], [45]. Due to its low complexity, the cost of training and prediction is expected to be low, enabling more applications to use this algorithm with blockchain smart contracts.

$$p(C|x) = \frac{p(x|C)p(C)}{p(x)}. \tag{1}$$

The posterior probability term $p(C|x)$ will be calculated in our case for all the classes to obtain the highest probability for the prediction. The C stands for class and x stands for features in the test dataset. The term $p(C)$ is the probability of class in the training dataset. This term will take the number of the same targets in the training class.

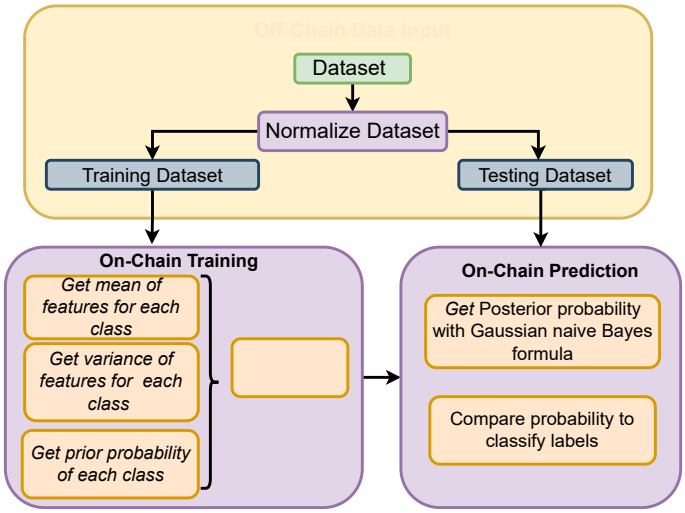

Fig. 3. Flow diagram of blockchain application to train and predict with smart contract function with Gaussian probability estimation

Naive Bayes likelihood is $p(x|C) = p(x_1|C)p(x_2|C)\dots p(x_N|C)$ under the assumption of independence among features. As per Bayes' theorem, we can also calculate prior probability and have a product of all the likelihood and priors to find the probability of that class.

After all the iterations, the class with the highest probability will be the predicted class.

Ethereum Blockchain: Ethereum [11] is a blockchain platform with a proof of work consensus protocol providing a distributed ledger transaction with security, planned to merge with a proof of stake(PoS) network chain soon [11]. As of the Ethereum white paper source from [11], Ethereum decentralized applications will not be affected actively by the merge. Moreover, the Ethereum virtual machine (EVM) used by Ethereum is the most stable platform and is borrowed by other blockchain platforms such as Polygon, Binance, Fantom, Polkadot, Avalanche and many more [46]–[48]. Due to this reason, our proposed solution can run on many blockchain platforms that use EVM.

On-chain Training & Prediction: Figure 3 shows the diagram of our proposed solution to train and predict on-chain with a naive Bayes algorithm. The input dataset is divided into training and testing datasets. The training dataset is normalized with a built-in python function. The smart contract function inside on-chain training computes means, variances, and the prior probability of each class with respective features. The on-chain prediction part calculates the estimated probability of class with the Gaussian probability distribution function in fractions for testing the dataset.

Probability Ratio Derivation: To overcome the limitation of floating-point support that hinders the accuracy of prediction, we have derived a probability estimation (discussed in section 6) to estimate the probable values of each class by comparing ratios of estimated values to perform the classification of labels using Gaussian naive Bayes.

Reusability of Derivation: The derivation used in this work can be adapted to different AI algorithms such as decision tree, logistic regression, random-forest and neural network, which can classify objects by comparing probability or likelihood. Decision-tree and random-forest must compute entropy(randomness) with logarithmic computations involving exponents. Logistic regression can use the Taylor series expansion for binary classification with sigmoid computation. Neural networks use sigmoid and softmax activation functions that involve e^x, which can utilize our solution. Table III shows a reusability matrix.

Performance Baseline: Scikit-learn [49] is a Python library providing simple and efficient tools for predictive data analysis. The library is open to everyone and can be reused in many contexts. We have generated a baseline prediction performance with this library to compare the accuracy of the smart contract with the respective datasets.

Expected Analysis: Our earlier work from [54] shows preliminary results on prediction. We have improved our derivations which is expected to provide higher prediction accuracy. Furthermore, we planned to record the blockchain transaction fee concerning the number of features and data samples that can provide us trend line for training and prediction. Moreover, we also considered registering the delays per prediction to give valuable insight into application choice.

Algorithm	Algorithm Function	Mathematical Function	Re-usability
Decision Tree	Entropy	Logarithm	Yes
Random Forest	Entropy	Logarithm	Yes
Logistic Regression	Sigmoid	Exponent	Yes
Neural Network	Sigmoid /Softmax	Exponent	Yes
Reinforcement Learning	Softmax	Exponent	Yes

TABLE III
RE-USABILITY OF THE DERIVATION IN OTHER AI ALGORITHM FUNCTIONS

VI. DERIVATION OF ALGORITHM FOR COMPUTING GAUSSIAN PROBABILITY PARAMETERS

As mentioned earlier, the comparison of posterior probabilities of a specific case x belonging to the classes C (i.e., $p(C|x)$) is required to predict the classification of case x. In this section, we provide the details of the calculation of the posterior probability, the techniques to reduce error in integer arithmetic, and the procedure of probability comparison using integers.

Posterior Probability: Let C_k denote the k-th ($k = 1, \ldots, N_C$) class and x_i denote i-th ($i = 1, \ldots, N$) feature. Under the assumption of Gaussian Naive Bayes with normal distributions $\mathcal{N}(\mu_{k,i}, \sigma^2_{k,i})$ for i-th feature of class C_k and conditional independence for the features of a given class, the numerator of the posterior probability can be rewritten as

$$p(x|C_k)p(C_k) = p(x_1|C_k)\ldots p(x_N|C_k)p(C_k)$$
$$= \frac{p(C_k) * 1}{\sqrt{((2\pi)^N \Pi_i \sigma^2_{k,i})}} e^{-\frac{(x_1-\mu_{k,1})^2}{2\sigma^2_{k,1}} - \ldots - \frac{(x_N-\mu_{k,N})^2}{2\sigma^2_{k,N}}}. \quad (2)$$

Techniques To Reduce Error In Integer Arithmetic: Some operations in the calculation of posterior probability, such as square root, divisions, and exponential function evaluations, may produce errors in integer arithmetic. We have implemented the following techniques in the current work to reduce the error.

1) *Reduce the number of divisions*.
We combine multiple divisions in the exponent of Eqn. (2) together to have a single division as
$$p(x|C_k)p(C_k) = \frac{p(C_k)}{\sqrt{((2\pi)^N \Pi_i \sigma^2_{k,i})}} e^{-\frac{\sum_i (x_i-\mu_{k,i})^2 \Pi_{j\neq i}\sigma^2_{k,j}}{2\Pi_i \sigma^2_{k,i}}}. \quad (3)$$

2) *Eliminate the operation square root*.
We take the square from both sides of Eqn. 3 to eliminate the operation of the square root as
$$(p(x|C_k)p(C_k))^2 = \frac{(p(C_k))^2}{((2\pi)^N \Pi_i \sigma^2_{k,i})} e^{-\frac{\sum_i (x_i-\mu_{k,i})^2 \Pi_{j\neq i}\sigma^2_{k,j}}{\Pi_i \sigma^2_{k,i}}}. \quad (4)$$

3) *Approximation of exponential function.*
Consider the exponential function $e^{a/b}$ where a, $b > 0$ are integers. With the quotient remainder theorem, we have
$$\frac{a}{b} = q + \frac{r}{b}, \quad (5)$$
where q, r, b are integers and $|r| < |b|$. Then the exponential function becomes
$$e^{a/b} = e^{q+r/b} = e^q e^{r/b}, \quad (6)$$
To deal with the term e^q, we approximate $e \approx 19/7$ and

consequently approximate $e^q \approx 19^q/7^q$.

The Taylor series deals with the remaining term $e^{r/b}$. The Taylor series of exponential function e^y is

$$e^y = \sum_{n=0}^{\infty} \frac{y^n}{n!} = 1 + y + \frac{y^2}{2!} + \frac{y^3}{3!} + \frac{y^4}{4!} + \dots. \quad (7)$$

Since $|y| = |r/b| < 1$ in our current work, we can use the first five terms to approximate the full exponential function. Similarly, to reduce the number of operations of division, we combine the divisions into a single one as

$$e^{r/b} \approx \frac{24b^4 + 24rb^3 + 12r^2b^2 + 4r^3b + r^4}{24b^4}. \quad (8)$$

Probability Comparison: Regardless of the number of classes, pairwise comparison is implemented each time. Since the comparison of $p(C_k|x)$ and $p(C_l|x)$ and the comparison of $p(C_k|x)^2$ and $p(C_l|x)^2$ indicate the same preference over the classes, we compare $p(C_k|x)^2$ and $p(C_l|x)^2$ to avoid the operation of square root. Specifically, the ratio of the two probabilities $p(C_k|x)^2/p(C_l|x)^2$ is calculated, which indicates that x is more likely to be in C_k if the ratio is greater than 1 and C_l if it is less than 1.

Denote $A0 = A1 = 1$, $B0 = D0 = (\Pi_i \sigma_{k,i}^2)$, $C_0 = \sum_i (x_i - \mu_{k,i})^2 \Pi_{j \neq i} \sigma_{k,j}^2$, $B1 = D1 = (\Pi_i \sigma_{l,i}^2)$, $C_1 = \sum_i (x_i - \mu_{l,i})^2 \Pi_{j \neq i} \sigma_{l,j}^2$. With the cancellation of the constant, the ratio is

$$\begin{aligned}
\frac{p(C_k|x)^2}{p(C_l|x)^2} &= \frac{\frac{A0}{B0} e^{-\frac{C0}{D0}}}{\frac{A1}{B1} e^{-\frac{C1}{D1}}} \frac{(p(C_k))^2}{(p(C_l))^2}, \\
&= \frac{A0 * B1}{B0 * A1} e^{-\frac{C0*D1 - C1*D0}{D0*D1}} \frac{(p(C_k))^2}{(p(C_l))^2}, \\
&= \frac{A}{B} e^{-\frac{C}{D}} \frac{(p(C_k))^2}{(p(C_l))^2}, \\
&= \frac{A}{B} e^{-Q} e^{-\frac{R}{D}} \frac{(p(C_k))^2}{(p(C_l))^2},
\end{aligned}$$

Note that $C/D = Q + R/D$ so we can simplify the notation using $A = A0 * B1$, $B = B0 * A1$, $C = C0 * D1 - C1 * D0$ and $D = D0 * D1$. With the aforementioned technique for the exponential function, e^{-Q} can be approximated by a ratio of integers $M1/N1$, and $e^{-\frac{C}{D}}$ can be approximated by another ratio of integers $M2/N2$. Denote $M = A*M1*M2*(p(C_k))^2$ and $N = B * N1 * N2 * (p(C_l))^2$, then the ratio $\frac{(p(C_k|x))^2}{(p(C_l|x))^2}$ is approximated by a ratio of integers M/N. We can conclude that x is more likely to be in C_k if $M > N$.

Note that all the calculations are integer operations if the inputs x_i, mean $\mu_{k,i}$, variance $\sigma_{k,i}^2$, and ratio $(p(C_k))^2/(p(C_l))^2$ are integers. Algorithm 1 shows functions relating to posterior probability estimation and prediction to classify objects.

Algorithm 1 Posterior Probability Comparison

function MULTIPLY_VARIANCE($variance[], position$) ▷ function to multiply variances
  $x \leftarrow 1$
  for (each i values in $variance[]$) **do**
    if $i == position$ **then** continue
    else
      $x \leftarrow x * v[i]$
    end if
  end for
  return x
end function
function GET_PARAMETERS($features[]$, $variance[]$, $mean[]$) ▷ Returns Gaussian probability parameters for each class in the form of $\frac{A}{B} e^{\frac{-C}{D}}$)
  $A0, B0, C0, D0 \leftarrow 1$
  for (each feature i in $feature[]$) **do**
    $B0 \leftarrow B0 * variance[i]$
    $C0 \leftarrow C0 + (feature[i] - mean[i])^2 * Multiply\_variance(variance, i)$
    $D0 \leftarrow B0$
  end for
  return A0,B0,C0,D0
end function
function COMPARE_PROBABILITY(A0,B0,C0,D0,Prior0, A1,B1,C1,D1,Prior1) ▷ Compare Gaussian probability of two class
  $A \leftarrow A0 * B1$
  $B \leftarrow B0 * A1$
  $C \leftarrow ((C0 * D1) - (D1 * C0))$
  $D \leftarrow D0 * D1$
  $Q \leftarrow C/D$
  $R \leftarrow C\%D$
  $eden \leftarrow 24D^4$
  $enum \leftarrow 24D^4 + 24D^3R + 12D^2R^2 + 4R^3D + R^4$
  $Y0 \leftarrow A * 7^Q * eden * Prior0^2$
  $Y1 \leftarrow B * 19^Q * enum * Prior1^2$
  return Y0,Y1
end function

VII. EXPERIMENTAL SETUP

Dataset: The datasets for machine learning algorithms are usually smaller than deep learning algorithms, which helps in low computation and reliable prediction accuracy. We gathered popular datasets from the UCI repository for our experiment. The three datasets are "Pima diabetes" [50] (diabetes detection), "Banknote authentication" [51] (banknote detection) and "Page block detection" [52] (Memory page block), are shown in Table 1. All of the datasets perform binary classification. The training and testing dataset is created from the main dataset with 80% weight on training and 20% weight on testing. We have also generated a simulated dataset with 64 random features to stress test the performance of the smart contract for scalability for prediction.

Data Pre-processing: The division of the dataset, preprocessing, normalization and conversion of floating points to integers are all executed with the Python framework. The input of features (if found to be decimal) is transformed into an integer instead of a floating-point variable. We have multiplied the features with a constant number, preferably with a power of 10, depending on the number of decimal values. This step makes sure that we would only feed integer input to the smart

Dataset	Features	Samples	Training Samples	Testing Samples
Pima Diabetes	8	768	614	154
Bank Note Authentication	4	1372	1097	278
Page Block Detection	10	5473	4378	1095

TABLE IV
POPULAR DATASETS WITH NUMBER OF FEATURES, NUMBER OF SAMPLES, TRAINING SAMPLES AND TESTING SAMPLES

contracts.

Blockchain Platform: We have tested our hypothesis on the local blockchain by deploying the smart contracts on ganache-cli for verifying prediction accuracy. We have deployed our smart contracts in a public test network for a real system test. Ropsten is a public test network for Ethereum widely used for testing smart contract deployments. We deployed our smart contract on the Ropsten network to record final performance metrics. The GitHub access to our project implementation, is provided in [53].

VIII. Performance Evaluation

Prediction Accuracy: The Pima Indian Diabetes dataset prediction provided an accuracy of 75 percent with smart contracts, similar to scikit-learn library function prediction accuracy. The banknote authentication provided an accuracy of 71 percent and page block detection datasets produced an accuracy of 85 percent. All of these prediction accuracies are comparable to a baseline accuracy of Python-based prediction using the same dataset. It is expected that the larger datasets will also produce similar accuracy.

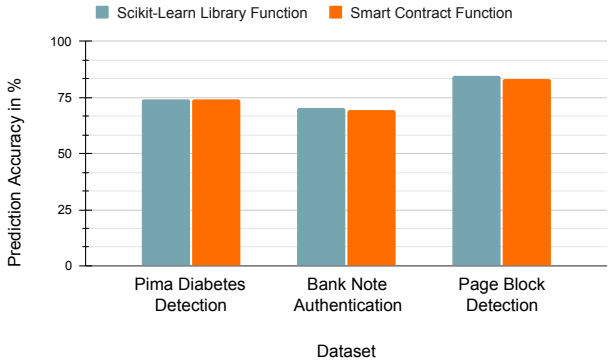

Fig. 4. Prediction accuracy comparison scikit-learn Python library versus smart contract deployment for "Pima diabetes detection", "Banknote authentication" and "Page block detection" dataset with our proposed derivation of Gaussian naive Bayes method inside smart contract

Training & Prediction Cost: Fig. 5 show the rise of cost in Ethers for training the number of sample inputs. The training cost involves the computation of the mean, variance, and prior probability of features concerning each class. Training costs rise linearly, providing a trendline for estimating the cost. The sample size can be expanded to increase training costs but will fall on the same linear line. Moreover, the prediction cost is given in figure 6 which also reveals a linear relationship between the number of features and the price of prediction, assuring scalability of our proposed model. Gas consumption is one of the units of measurement for the difficulty of function. We can convert the gas consumption to ethers by the formula of $transactionfee = (gas_consumed * (gas_price + basefee)/10^9$ as per the Ethereum white paper [11]. We can calculate the prediction cost of ethers using the formula in figure 6.

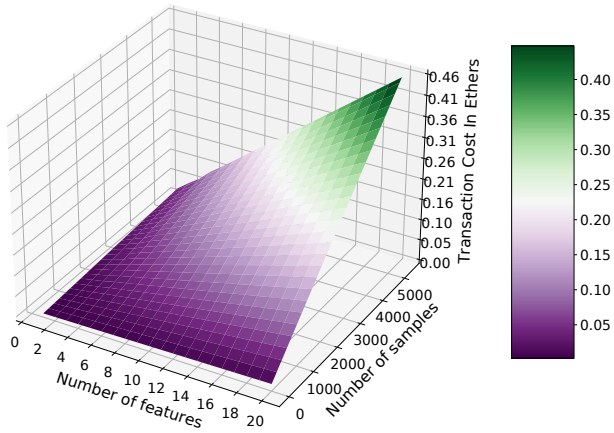

Fig. 5. Ethereum Ropsten Transaction cost (Ethers) for training naive Bayes algorithm, computing mean, variance, and prior probability. The relationship between the number of samples, number of features, and transaction cost are linear.

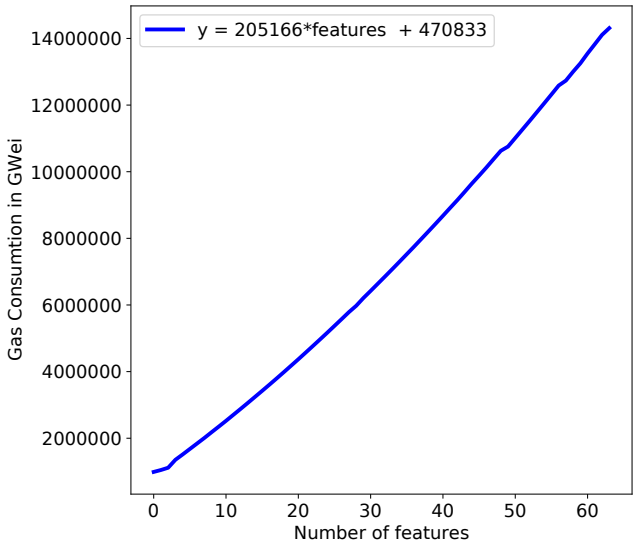

Fig. 6. Transaction fee for computing posterior probability with respect to the rising number of features reveals a linear relationship assuring scalability.

Prediction Time: Fig. 7 shows the distribution of time taken for each prediction per class, revealing a gamma distribution of seconds by Ethereum miners. The y-axis shows the probability density value of the particular event's duration. The x-axis represents the transaction duration of the events. Gamma distribution has a shape parameter that defines the shape of the curve, while the rate parameter specifies the spread of the curve. The shape parameter does not vary much, but the rate parameter varies between 0.07 to 0.12, with 0.12 making the spread narrow and 0.07 making the spread wider. The distribution is found to be independent of several features of the datasets. We have concluded that the distribution of time for the transaction of prediction forms a gamma curve where

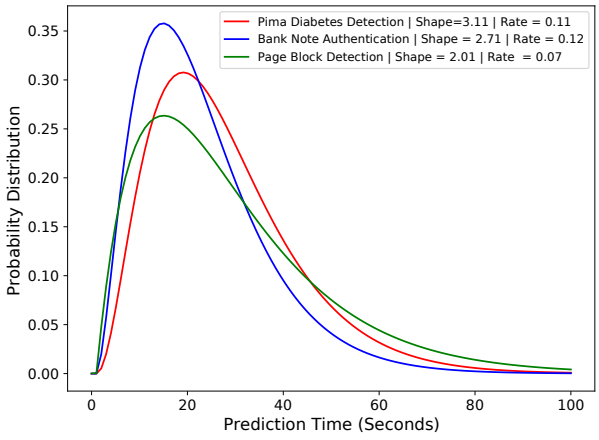

Fig. 7. Prediction time of test dataset forms a gamma distribution showing a high probability of prediction events within the range of 15-25 seconds

most of the number of transactions varies between 15 - 25 seconds and is independent of the number of features.

Algorithm Performance: The Naive Bayes algorithms have a time complexity of $O(nd)$ for training and $O(dc)$ for prediction, where n is the number of samples, d is the number of features and c is the number of classes with built-in Python scikit-learn functions. With our derived method of the algorithm for smart contracts, we have also achieved the time complexity of $O(nd)$ for training and $O(dc)$ for prediction.

IX. Limitation And Challenges

Block Gas Limit: One of the challenges faced in this project is related to the limitation of the blockchain network to support more computation within the given block gas limit of the Ethereum network. Currently, the Ethereum main chain has a gas limit of 30,000,000 Gwei and the Ropsten test network with 30,000,000. For this reason, sending more than 6000 sets of input data points for training creates a block gas limit error and requires a separate transaction. Ethereum blockchain gas limit has been growing with rising demands of applications. The gas limit is expected to expand more to support more data inputs in the future.

Integer Value Overflow: Solidity smart contracts in Ethereum blockchain support signed integer values ranging from negative $2^{256}/2$ to positive $2^{256}/2$. The probability computations of multiplying integers for Gaussian probability with more than ten features emitted integer overflow errors that provided outputs out of the specified range. This created a low prediction accuracy. This problem was solved using a reduction method of dividing the large number with a value close to 2^{30} that minimized the individual probability estimated values and made predictions more accurate.

Algorithm Accuracy Machine learning algorithms do not perform well with the rising number of samples as the algorithms cannot perform feature engineering on their own. Naive Bayes is one of the algorithms that consider the features naive and independent, which cannot produce high accuracy predictions. The prediction accuracy of our proposed solution also suffers from this limitation.

X. Conclusion

We have proposed a novel solution to develop a trusted machine learning model with a smart contract in the blockchain framework that can train the model and produce predictions with high accuracy. We proposed an on-chain training and prediction model of blockchain deployment to meet our hypothesis. The training of the naive Bayes algorithm included computing means, variances, and the prior probability of features concerning each class performed inside a smart contract. The smart contract slashes any floating-point output due to a lack of compatibility. The derivation of the posterior probability with Gaussian naive Bayes rule using Taylor series expansion provides a novel method to obtain probability estimations between different classes. Our derivation can be used in machine learning algorithms such as decision-tree, random-forest, and logistic-regression. With this derivation, our prediction experiment produced accuracy similar to scikit-learn Python library functions. The cost of training forms a linear relationship with the rising number of samples and features. The prediction cost graph also shows a linear rise assuring the scalability of our proposed method.

XI. Future Work

More AI Algorithms : From table III, We have already implemented neural network-based prediction on smart contract and submitted a conference paper to a different conference (pending review). We plan to deploy other mentioned algorithms in a similar way to prove our hypothesis with convincing results. The implementation of decision-tree and random-forest will involve logarithmic computations for building trees with splits. Logistic regression will again use the sigmoid function for binary classification. These projects are under development and will soon be ready for submission to conferences.

Other Blockchain Platforms : Different blockchain platforms emerged with more scalability and low-cost computation targets. Different consensus protocols are also used in those platforms. We plan to implement our solution on other platforms such as Polkadot, Binance, Cardano, and Algorand to study the performance statistics to help researchers and users choose the right platform considering scalability, delay, and efficiency.

XII. Acknowledgement

We thank National Security Agency for partially supporting our research work through grants H98230-20-1-0329, H98230-20-1-0403, H98230-20-1-0414, and H98230-21-1-0262.

References

[1] Brundage, M., Avin, S., Wang, J., Belfield, H., Krueger, G., Hadfield, G., ... & Anderljung, M. (2020). Toward trustworthy AI development: mechanisms for supporting verifiable claims. arXiv preprint arXiv:2004.07213.

[2] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, G. Loukas, "A Taxonomy and Survey of attacks against machine learning",Volume 34,2019,100199,ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2019.100199.

[3] Peterson, L. E. (2009). K-nearest neighbor. Scholarpedia, 4(2), 1883.

[4] Rish, I. (2001, August). An empirical study of the naive Bayes classifier. In IJCAI 2001 workshop on empirical methods in artificial intelligence (Vol. 3, No. 22, pp. 41-46).

[5] Safavian, S. R., Landgrebe, D. (1991). A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3), 660-674.

[6] Stuart D. Levi and Alex B. Lipton, Skadden, Arps, Slate, Meagher & Flom LLP, "An introduction to Smart Contracts and their Potential and Inherent Limitations" https://corpgov.law.harvard.edu/2018/05/26/an-introduction-to-smart-contracts-and-their-potential-and-inherent-limitations/, Accessed September 2021

[7] "Solidity Programming guide", https://docs.Soliditylang.org/en/v0.8.9/, Accessed September 2021

[8] "Introduction to Smart Contract", https://ethereum.org/en/developers/docs/smart-contracts/, Accessed September 2021

[9] S. Wang, Y. Yuan, X. Wang, J. Li, R. Qin and F. -Y. Wang, "An Overview of Smart Contract: Architecture, Applications, and Future Trends," 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 108-113, DOI: 10.1109/IVS.2018.8500488.

[10] Szabo, Nick. "Smart contracts: building blocks for digital markets." EXTROPY: The Journal of Transhumanist Thought,(16) 18, no. 2 (1996): 28

[11] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." white paper 3, no. 37 (2014).

[12] K. Upadhyay, R. Dantu, Z. Zaccagni and S. Badruddoja, "Is Your Legal Contract Ambiguous? Convert to a Smart Legal Contract," 2020 IEEE International Conference on Blockchain (Blockchain), 2020, pp. 273-280, doi: 10.1109/Blockchain50366.2020.00041.

[13] K. Upadhyay, R. Dantu, Y. He, A. Salau and S. Badruddoja, "Make Consumers Happy by Defuzzifying the Service Level Agreements," 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2021, pp. 98-105, doi: 10.1109/TPSISA52974.2021.00011.

[14] K. Upadhyay, R. Dantu, Y. He, A. Salau and S. Badruddoja, "Paradigm Shift from Paper Contracts to Smart Contracts," 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2021, pp. 261-268, doi: 10.1109/TPSISA52974.2021.00029.

[15] K. Upadhyay, R. Dantu, Y. He, S. Badruddoja and A. Salau, "Can't Understand SLAs? Use the Smart Contract," 2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA), 2021, pp. 129-136, doi: 10.1109/TPSISA52974.2021.00015.

[16] A. Salau, R. Dantu, K. Morozov, K. Upadhyay, S. Badruddoja, "Multi-Tier Reputation for Data Cooperatives", The 3rd International Conference on Mathematical Research for Blockchain Economy, 2022

[17] A. Salau, R. Dantu, K. Morozov, K. Upadhyay, and S. Badruddoja (2022). Towards a Threat Model and Security Analysis for Data Cooperatives. In Proceedings of the 19th International Conference on Security and Cryptography - SECRYPT, ISBN 978-989-758-590-6; ISSN 2184-7711, pages 707-713. DOI: 10.5220/0011328700003283

[18] A. Salau, R. Dantu and K. Upadhyay, "Data Cooperatives for Neighborhood Watch," 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2021, pp. 1-9, doi: 10.1109/ICBC51069.2021.9461056.

[19] Yang, D., Long, C., Xu, H., & Peng, S. (2020, March). A review on scalability of blockchain. In Proceedings of the 2020 The 2nd International Conference on Blockchain Technology (pp. 1-6).

[20] A. Chauhan, O. P. Malviya, M. Verma and T. S. Mor, "Blockchain and Scalability," 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C), 2018, pp. 122-128, DOI: 10.1109/QRS-C.2018.00034.

[21] Online article, "Polkadot Dapps", https://www.dapp.com/topics/polkadot banner.

[22] Online article, "Cardano Crowd", https://cardanocrowd.com/dapps.

[23] Online article, "State of the Dapps", https://www.stateofthedapps.com/stats/platform/ethereumnew

[24] Online article, "Neo Dapps" , https://ndapp.org/overview

[25] Online article, "Dapp Radar", https://dappradar.com/rankings

[26] "Solidity Smart Contract language Documentation", https://docs.Soliditylang.org/en/v0.8.14/

[27] "TEAL Smart Contract Language Documentation", https://developer.algorand.org/docs/get-details/dapps/avm/teal /specification/

[28] "Marlowe Smart Contract Language Documentation" https://plutus-apps.readthedocs.io/en/latest/marlowe/tutorials/marlowe-data.htmlmarlowe

[29] "Polkadot Smart Contract Language Documentation", https://wiki.polkadot.network/docs/build-smart-contracts

[30] "Neo Smart Contract Language Documentation", https://docs.neo.org/docs/en-us/develop/write/basics.html

[31] P. Kumar, "Computational Complexity Of ML Models" , https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770, Retrieved March 2022

[32] Prashant, " Computational Complexity Of ML Algorithms", https://medium.com/analytics-vidhya/computational-complexity-of-ml-algorithms-1bdc88af1c7a, Retrieved March 2022

[33] D. Wang, C. Chang, J. Pai, B. Xu, H. Gu, S. Liu, K. Ye, "Artificial Intelligence Computing Platform Driven By BlockChain", https://www.deepbrainchain.org/assets/pdf/DeepBrainChainWhitepaper_en.pdf

[34] Cortex labs.2018. "AI Smart Contracts — The Past, Present, and Future",December 6, 2018, Retrieved September 3, 2020 from https://medium.com/cortexlabs/ai-smart-contract-5018dc56e2d8

[35] A. B. Kurtulmus and K. Daniel, Trustless machine learning contracts; evaluating and exchanging machine learning models on the Ethereum Blockchain, 2018, [online] Available: https://arXiv:1802.10185. https://github.com/algorithmiaio/danku

[36] Matrix Technical Whitepaper, Sep. 2018, [online] Available: https://www.matrix.io/html/MATRIXTechnicalWhitePaper.pdf.

[37] J. D. Harris and B. Waggoner, "Decentralized and Collaborative AI on Blockchain," 2019 IEEE International Conference on Blockchain (Blockchain), Atlanta, GA, USA, 2019, pp. 368-375, DOI: 10.1109/Blockchain.2019.00057.

[38] Tao Wang and Xinmin Wu and Taiping He.2019." Trustable and Automated Machine Learning Running with Blockchain and Its Applications", KDD 2019 Auto ML

[39] H. Kim. S. Kim. J. Y. Hwang and C. Seo.2019. "Efficient Privacy-Preserving Machine Learning for Blockchain Network",in IEEE Access, vol. 7, pp. 136481-136495, 2019, DOI: 10.1109/ACCESS.2019.2940052.

[40] Solidity v0.7.4- Documentation, https://docs.Soliditylang.org/en/v0.7.4 /abi-spec.html#types

[41] B. Vieira, Fixidity Library, https://github.com/CementDAO/Fixidity

[42] M. Vladimirov, ABDK Fixed Point Libraries, https://github.com/abdk-consulting/abdk-libraries-Solidity

[43] Y. Liu, F. R. Yu, X. Li, H. Ji and V. C. M. Leung, "Blockchain and Machine Learning for Communications and Networking Systems," in IEEE Communications Surveys & Tutorials, vol. 22, no. 2, pp. 1392-1431, Second quarter 2020, DOI: 10.1109/COMST.2020.2975911.

[44] P. Kumar, "Computational Complexity of ML Models", https://medium.com/analytics-vidhya/time-complexity-of-ml-models-4ec39fad2770, Retrieved March 2022

[45] Sunil, "6 Easy Steps to Learn Naive Bayes Algorithm With Codes in Python and R", https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#: :text=Pros%3A,you%20need%20less%20training%20data., Retrieved March 2022

[46] "Ethereum State of the dapps", https://www.stateofthedapps.com/stats, Retrieved March 2022

[47] Master ventures, "The Explosion of EVM Blockchains", https://masterventures.medium.com/the-explosion-of-evm-blockchains-7dd10537aaba, February 2022

[48] Coin guides, "List of all EVM blockchains and how to add any EVM network to Metamask", https://coinguides.org/evm-blockchains-add-evm-network/, November 2021

[49] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of Machine Learning Research, 12, 2825-2830.

[50] PIMA Indian diabetes dataset https://www.kaggle.com/uciml/pima-indians-diabetes-database

[51] Bank Note Authentication Dataset https://archive.ics.uci.edu/ml/datasets/banknote+authentication

[52] Page Block Detection Dataset https://archive.ics.uci.edu/ml /datasets/Page+Blocks+Classification

[53] "Project Implementation Github Link", https://github.com/syber2020/Blockchain_NB.git

[54] Badruddoja, S., Dantu, R., He, Y., Upadhayay, K., Thompson, M. (2021, May). Making smart contracts smarter. In 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC) (pp. 1-3). IEEE.