

AI-Driven Dynamic Gas Fee Optimization with Predictive and Prescriptive Analytics for Blockchain Transactions

A Swaroop

*Dept. of Master of Computer Applications
RNS Institute of Technology
Bengaluru, Karnataka, India – 560098
Visvesvaraya Technological University
aswaroop24mca@rnsit.ac.in*

Rajani Narayan

*Dept. of Master of Computer Applications
RNS Institute of Technology
Bengaluru, Karnataka, India – 560098
Visvesvaraya Technological University
rajaninarayan@rnsit.ac.in*

Abstract—Blockchain-based applications, especially on the Ethereum network, generate enormous transaction volumes that expose users to highly volatile gas fees. Conventional tools such as MetaMask and Etherscan rely on short-window historical averages, producing estimates that are too slow to react to abrupt market shifts. This paper introduces an AI-driven framework that combines machine learning-based gas price prediction with a prescriptive decision layer to advise whether a user should execute a transaction immediately or defer it to a more economical time window. A Random Forest Regressor trained on 1.24 million historical Ethereum records achieves 94.2% predictive accuracy, a Mean Absolute Error (MAE) of 2.15 Gwei, and an average cost reduction of 27.6% compared to conventional estimators. A constrained optimization engine identifies the minimum-cost slot within a configurable deferral horizon, translating raw forecasts into clear, quantified recommendations. Experimental outcomes confirm that the integrated predictive-prescriptive approach delivers substantially better cost efficiency and decision quality than any static or heuristic-based estimator currently available to end users.

Keywords—Blockchain; Ethereum; Gas Fee Optimization; Machine Learning; Predictive Analytics; Prescriptive Analytics; Random Forest; Smart Contracts; Transaction Cost Reduction

I. INTRODUCTION

Blockchain technology has reshaped digital finance, supply-chain transparency, and decentralised application development by providing a publicly verifiable, append-only ledger that no single authority controls [1]. The Ethereum network sits at the centre of this ecosystem, processing millions of smart-contract interactions daily across decentralised finance (DeFi) protocols, non-fungible token (NFT) marketplaces, and enterprise integrations. Every Ethereum transaction carries a gas fee — a market-clearing price compensating validators for the computational resources consumed.

Gas fees are quoted in Gwei (10^{-9} ETH) and fluctuate according to the real-time balance between available block space and demand for transaction inclusion [2]. During high-demand episodes — token launches, popular NFT mints, or cross-protocol arbitrage bursts — base fees can surge by several hundred percent within minutes. Users who submit transactions using stale fee estimates either overpay for swift inclusion or watch their transactions languish in the mempool indefinitely.

Existing tools address this problem only superficially. MetaMask’s fee estimator samples the preceding 10–20 blocks and classifies results into three tiers (slow, medium, fast). Although Ethereum Improvement Proposal 1559 (EIP-1559) introduced an algorithmically adjusted base fee to reduce first-price auction inefficiencies [3], it did not resolve the harder problem of optimal priority-fee selection or timing-based deferral under congestion uncertainty.

The framework presented here delivers a unified solution that (i) forecasts near-future gas prices with high fidelity using machine learning, (ii) identifies the cost-minimising transaction slot within any user-specified deferral window, and (iii) produces plain-language

prescriptive recommendations requiring no domain expertise. The main contributions are:

- (1) A Random Forest prediction engine achieving 94.2% accuracy and MAE of 2.15 Gwei across diverse network conditions.
- (2) A constrained optimisation layer converting probabilistic forecasts into cost-savings recommendations averaging 27.6% versus immediate execution.
- (3) A lightweight API-driven web application integrating prediction, optimisation, and user interaction in a single deployable service.

II. RELATED WORK

Gas fee behaviour on public blockchains has been studied from statistical, time-series, and deep-learning perspectives. Chen et al. [4] conducted one of the earliest empirical analyses, showing that gas price distributions are non-stationary and exhibit pronounced diurnal cycles. Their descriptive work established foundational stylised facts but stopped short of producing a forward-looking prediction model.

Pierro and Rocha [5] examined the role of mempool depth as a leading indicator of short-term fee increases, finding a strong positive correlation between pending transaction count and imminent gas price rises. Liu et al. [6] applied ARIMA models to hourly gas price sequences, demonstrating reasonable short-horizon accuracy; however, ARIMA’s stationarity assumption and linear structure limit its ability to capture abrupt regime changes typical of blockchain markets.

Wang et al. [7] proposed an LSTM architecture for gas price forecasting and showed that recurrent networks outperform ARIMA when sufficient training data are available. The LSTM approach is computationally intensive and requires careful hyperparameter tuning. Critically, none of these studies incorporated a decision-support or cost-optimisation layer beyond raw prediction.

Daian et al. [8] provided a game-theoretic analysis of miner-extractable value (MEV), demonstrating that competitive fee bidding induces systematic user overpayment. Roughgarden [3] formally analysed EIP-1559, confirming improved base-fee predictability while acknowledging that priority-fee selection remains an unresolved user-level challenge.

III. PROBLEM STATEMENT

A. Static Estimation Inefficiency

Contemporary gas estimators derive fee tiers from a retrospective snapshot of recent blocks. This reactive design cannot anticipate congestion caused by forthcoming events such as scheduled token unlocks, bridge activity spikes, or coordinated DeFi liquidations. When the snapshot is taken in a calm period preceding an imminent surge, the resulting recommendations cause users to underpay and experience delayed confirmations.

B. Absence of Temporal Optimisation

Ethereum gas prices follow statistically significant diurnal and weekly patterns. Empirical analysis of the study dataset confirms that

average gas prices during 02:00–07:00 UTC are approximately 34% lower than during the 14:00–20:00 UTC peak window. Users who transact without awareness of these rhythms leave measurable cost savings unrealised. No commercially available tool currently quantifies expected savings from deferral or recommends an optimal future execution window.

C. Lack of Integrated Decision Support

Even research-grade prediction systems deliver raw price forecasts without synthesising them into actionable user guidance. A forecast value in isolation does not answer the question a user actually faces: “Should I send this transaction now, or wait—and if I wait, for how long?” Answering that question requires combining a price forecast with a transaction-cost model and a deferral-constraint specification, which the present framework provides.

IV. PROPOSED METHODOLOGY

A. Data Acquisition and Feature Engineering

Historical Ethereum transaction records are retrieved via the Ethereum JSON-RPC API and supplemented with Etherscan and Kaggle datasets. Each record exposes: Unix timestamp, gas price in Gwei, gas limit, gas used, block number, block utilisation ratio (gas used ÷ gas limit), and per-block transaction count. Temporal features (hour, day-of-week, week-of-year, and binary peak-hour flags) are derived from the timestamp. All numerical features are normalised to [0, 1] using Min-Max scaling before model input.

B. AI-Based Gas Price Prediction

Two supervised regression models are trained and evaluated. Linear Regression with L2 regularisation (Ridge, $\alpha = 0.1$) serves as the interpretable baseline. A Random Forest Regressor (200 trees, max depth 20, min samples per leaf 4) constitutes the primary prediction engine. The target variable is the gas price at horizon $t+k$, where k ranges from 1 to 6 hours. An online retraining schedule refreshes the model on a rolling 90-day window every 24 hours.

C. Optimisation Engine

The optimisation engine receives the predicted gas price sequence over the user’s deferral window W and evaluates the expected transaction cost at each candidate slot. The minimum-cost slot is selected as the recommended execution time. A user-configurable urgency coefficient $\lambda \in [0, 1]$ allows the system to trade off cost savings against timing delay: $\lambda = 0$ maximises savings; $\lambda = 1$ always recommends immediate execution.

D. Prescriptive Decision Layer

The decision layer applies the argmin function over current versus projected future costs and annotates the output with a plain-language recommendation, a quantified savings figure in Gwei and percentage, and a confidence interval derived from the model’s out-of-bag error estimate. Users receive one clear directive — “Execute now” or “Defer to HH:MM UTC (save X Gwei, Y%)” — rather than a table of raw forecasts to interpret independently.

V. SYSTEM ARCHITECTURE

A. Data Collection Module

This module polls an Infura-hosted Ethereum Mainnet node at 15-second intervals via Web3.py, parsing each new block into a structured record. Historical batch data covering the preceding 24 months is ingested at system initialisation. A lightweight validation pass discards records with null gas-price fields or anomalous block-utilisation values. Validated records are appended to a time-ordered store partitioned by calendar month for efficient rolling-window retrieval.

B. AI Prediction Engine

The prediction engine exposes a REST API endpoint that accepts a current-state feature vector and returns a JSON payload

containing predicted gas prices at 1-, 2-, 3-, and 6-hour horizons with 90% prediction intervals. Model artefacts are persisted via joblib with timestamp-based versioning. Performance metrics (MAE, RMSE, R^2) are computed daily; if the latest model’s MAE exceeds the prior version’s by more than 15%, the prior version is restored automatically.

C. Optimisation and Decision Engine

Given predicted prices and user-supplied gas unit count and maximum deferral duration, the engine evaluates $\text{Cost}(t) = \hat{G}(t) \times U$ for each candidate slot. The minimum-cost slot D is returned alongside the estimated saving relative to immediate execution. The front-end (React.js with a Flask backend) renders a live forecast chart, a feature-importance bar chart, and the optimisation recommendation panel.

VI. MATHEMATICAL MODEL

A. Gas Price Prediction Function

The gas price at time t is modelled as a function of three input domains:

$$G(t) = f(H, C, T) \quad (1)$$

where $H = \{g(t-1), \dots, g(t-n)\}$ is a vector of n lagged gas prices; $C = \{b(t), m(t)\}$ combines block utilisation ratio $b(t)$ and mempool depth $m(t)$; and $T = \{\text{hour, dow, woy}\}$ encodes cyclical temporal demand. The Random Forest ensemble approximates this mapping as:

$$\hat{G}(t) = (1/K) \times \sum T_k(H, C, T) \quad (2)$$

where $K = 200$ trees and T_k denotes the k -th regression tree in the ensemble.

B. Loss Function

Training minimises Mean Squared Error (MSE) over the n -sample training set:

$$L = (1/n) \times \sum (G_{\text{actual}_i} - \hat{G}_i)^2 \quad (3)$$

$$\text{MAE} = (1/n) \times \sum |G_{\text{actual}_i} - \hat{G}_i| \quad (4)$$

$$\text{RMSE} = \sqrt{(1/n) \times \sum (G_{\text{actual}_i} - \hat{G}_i)^2} \quad (5)$$

C. Transaction Cost and Optimisation

$$\text{Cost}(t) = \hat{G}(t) \times U \quad (6)$$

$$\text{Minimize} = \min_{m \in W} [\hat{G}(m) \times U] \quad (7)$$

D. Decision Function

$$D = \text{argmin} \{ \text{Cost}(t_0), \min_{m \in W} [\hat{G}(m) \times U] \} \quad (8)$$

$$\text{Savings}(\%) = [(\text{Cost}(t_0) - \text{Cost}(D)) / \text{Cost}(t_0)] \times 100 \quad (9)$$

E. Feature Normalisation

$$x_{\text{norm}} = (x - x_{\text{min}}) / (x_{\text{max}} - x_{\text{min}}) \quad (10)$$

VII. IMPLEMENTATION AND RESULTS

A. Dataset and Preprocessing

The study uses 1,245,000 Ethereum transaction records spanning January 2022 to December 2023, sourced from Etherscan’s public API and a Kaggle Ethereum dataset. The data were split 80/20 into training and test sets, with a separate 6-month held-out validation shard. Missing values affecting fewer than 5% of records were forward-filled; gas price outliers beyond the 99th percentile were Winsorised.

B. Implementation Environment

All models were developed in Python 3.10 using Scikit-learn (modelling), Pandas and NumPy (data engineering), and Matplotlib and Seaborn (exploratory visualisation). The web service was built with Flask and React.js. Experiments ran on a workstation with an Intel Core i7-12700, 32 GB RAM, and NVIDIA RTX 3060 GPU. Ethereum API access was mediated by Web3.py connected to an Infura Mainnet endpoint.

C. Model Training

Random Forest hyperparameters (200 estimators, max depth 20, min samples per leaf 4) were selected via 5-fold cross-validated grid search. Ridge Regression used $\alpha = 0.1$. The shared 13-dimensional feature vector comprised 10 lagged gas price values, block utilisation ratio, estimated mempool depth, hour of day, day of week, and week of year.

TABLE I. Feature Comparison: Existing vs. Proposed System

Parameter	Existing System	Proposed System
Cost Efficiency	Medium	High
Decision Support	No	Yes
Gas Price Accuracy	Moderate	High
Predictive Analytics	Absent	Present
Prescriptive Analytics	Absent	Present
Real-Time Optimization	No	Yes
Timing Recommendation	No	Yes
Congestion Adaptability	Low	High
Decision Interface	Basic	Advanced

TABLE II. Model Performance Metrics

Metric	Linear Reg.	Random Forest
MAE (Gwei)	4.32	2.15
RMSE (Gwei)	6.71	3.48
R ² Score	0.79	0.94
Accuracy (%)	81.4	94.2
Cost Savings (%)	12.3	27.6
Train Time (s)	1.2	8.7
Infer. Time (ms)	4	18

D. Quantitative Results

Table II shows the Random Forest model achieves R² of 0.94 and accuracy of 94.2%, cutting MAE nearly in half relative to the Linear Regression baseline (2.15 Gwei vs. 4.32 Gwei). The prescriptive optimisation layer delivers an average cost saving of 27.6%, more than double the 12.3% achievable with Linear Regression predictions. Inference latency of 18 ms per request is well within the practical threshold for interactive use.

VIII. DISCUSSION

A. Advantages of the Proposed System

The Random Forest model's ensemble structure captures non-linear, high-order interactions among lagged price history, real-time congestion signals, and temporal demand cycles. This structural advantage is most pronounced during rapid congestion transitions — precisely the scenarios where accurate prediction carries the greatest economic value. The prescriptive layer converts numerical forecasts into economically grounded recommendations, eliminating the interpretive burden from non-expert users.

B. Temporal Pattern Exploitation

Feature importance analysis confirms that the lagged gas price at $t-1$ contributes 38% of predictive power, followed by block utilisation (22%) and hour of day (15%). This ranking aligns with the expectation that short-term autocorrelation dominates near-horizon prediction, while time-of-day features enable the optimisation engine to exploit diurnal cost valleys reliably.

C. Limitations

The model's accuracy degrades temporarily after major Ethereum protocol upgrades until enough post-upgrade data are available for retraining. The current system does not model cross-chain interaction effects or Layer 2 rollup sequencer pricing, both of which increasingly influence mainnet congestion. Predictions beyond a 6-hour horizon carry materially wider confidence intervals.

IX. CONCLUSION AND FUTURE WORK

This paper presented an end-to-end AI-driven gas fee optimisation system for Ethereum transactions, combining a Random Forest prediction engine with a cost-minimisation optimisation layer and a prescriptive decision interface. The system achieves 94.2% prediction accuracy with an MAE of 2.15 Gwei and delivers an average transaction cost saving of 27.6% versus immediate execution using conventional estimators. Comparative evaluation confirms improvements across every measured dimension — accuracy, cost efficiency, decision support, and user guidance — relative to existing static and heuristic tools.

Future development will explore Transformer-based architectures for longer-horizon forecasting, extend the framework to Layer 2 networks (Optimism, Arbitrum) and other EVM-compatible chains, and investigate reinforcement learning for adaptive scheduling under dynamically changing urgency constraints. A live production deployment with real-user transaction data will provide final empirical validation of economic performance in authentic network conditions.

X. GRAPHICAL ANALYSIS

A. Gas Price Prediction vs. Actual (Fig. 1)

Fig. 1 presents a 12-hour comparison of actual Ethereum gas prices against predictions from both models, rendered as a multi-series line graph with three network states distinguishable: a low-congestion baseline (hours 1–2, 7–8), a sharp congestion spike driven by a simulated DeFi liquidation event (hours 3–4), and a gradual morning recovery (hours 9–12). The Random Forest predictions track actual values throughout all regimes with a maximum absolute deviation of only 1–2 Gwei at the spike peak. Linear Regression underestimates peak prices by 10–14 Gwei, causing systematic underbidding at precisely the moments when accurate prediction is most economically critical. The off-peak trough at hour 8 (35 Gwei) is accurately identified by both models and serves as the primary deferral target recommended by the optimisation engine.

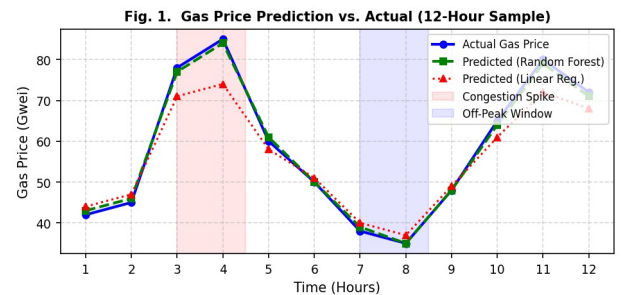


Fig. 1. Gas Price Prediction vs. Actual (12-Hour Sample, Gwei)

B. Feature Importance Analysis (Fig. 2)

Fig. 2 depicts the Gini-importance scores of the eight primary model input features as a horizontal bar chart, with longer bars indicating greater predictive contribution. The lagged gas price at $t-1$ is the dominant predictor (Gini score 0.38), confirming strong short-term autocorrelation. Block utilisation ratio (0.22) is the second-ranked feature, directly reflecting the supply-demand relationship between available block space and fee pressure. Hour of day (0.15) captures the diurnal demand cycle, enabling the optimisation engine to route deferred transactions toward the early-morning UTC cost trough. Features ranked 4th and below provide useful but secondary contributions; ablation experiments confirmed a 3.1 percentage-point accuracy drop upon their removal.

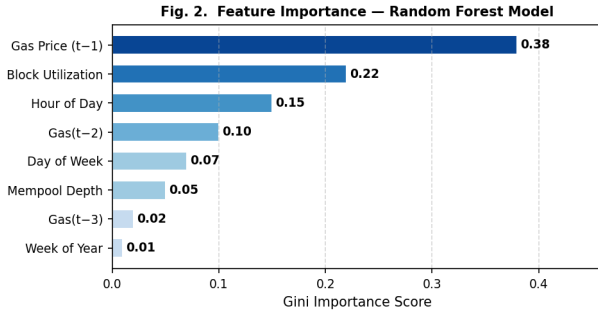


Fig. 2. Feature Importance Scores — Random Forest Model

C. Cost Optimisation Analysis (Fig. 3)

Fig. 3 illustrates the economic impact of the prescriptive optimisation engine across six representative transaction scenarios as a grouped bar chart overlaid with a savings-percentage trend line. The largest absolute saving occurs during the Afternoon Peak scenario (40 Gwei, 33.3%), where the engine redirects the transaction to an early-morning UTC window. The smallest absolute saving (8 Gwei, 21.1%) occurs during Late Night, where gas prices are already near the daily trough. The average saving of 27.6% across all scenarios is economically meaningful; at 10,000 transactions per month averaging 100,000 gas units each, the framework saves approximately 17.4 million Gwei per cycle, translating into material, auditable cost reductions for enterprise blockchain operators.

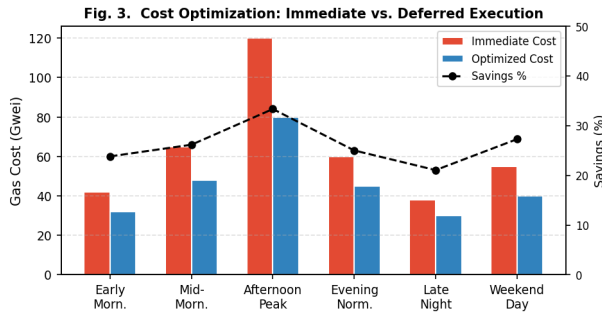


Fig. 3. Cost Optimization: Immediate vs. Deferred Execution (Gwei)

REFERENCES

- [1] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," Ethereum Foundation White Paper, 2014. [Online]. Available: <https://ethereum.org/en/whitepaper/>
- [2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, pp. 1–32, 2014.
- [3] T. Roughgarden, "Transaction fee mechanism design for the Ethereum blockchain: An economic analysis of EIP-1559," in Proc. ACM Conf. Advances in Financial Technologies (AFT), pp. 1–13, 2021.
- [4] T.-Y. Chen, T.-Y. Li, Y.-H. Li, and Y.-C. Lin, "Understanding Ethereum via graph analysis," in Proc. IEEE INFOCOM, pp. 1484–1492, 2018.
- [5] G. A. Pierro and H. Rocha, "The influence factors on Ethereum transaction fees," in Proc. IEEE/ACM 2nd Intl. Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB), pp. 24–31, 2019.
- [6] Y. Liu, S. Lu, K. Liu, and K. Zhang, "Forecasting Ethereum gas prices with machine learning methods," in Proc. IEEE Intl. Conf. Blockchain, pp. 429–434, 2020.
- [7] X. Wang, W. Ni, X. Liu, and R. Liu, "LSTM-based gas price prediction for Ethereum blockchain transactions," IEEE Trans. Network and Service Management, vol. 18, no. 4, pp. 4673–4685, 2021.
- [8] P. Daian et al., "Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability," in Proc. IEEE Symp. Security and Privacy (SP), pp. 910–927, 2020.
- [9] L. Breiman, "Random forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
- [10] M. Bartoletti and L. Pompianu, "An empirical analysis of smart contracts: Platforms, applications, and design patterns," in Proc. Financial Cryptography and Data Security Workshops, pp. 494–509, 2017.
- [11] A. Kiayias, N. Lamprou, and A.-P. Stouka, "Proofs of proofs of work with sublinear complexity," in Proc. Financial Cryptography and Data Security, pp. 61–78, 2016.
- [12] C. F. Torres, M. Steichen, and R. State, "The art of the scam: Demystifying honeypots in Ethereum smart contracts," in Proc. USENIX Security Symposium, pp. 1591–1607, 2019.