Final Report: Forecasting Bitcoin Transaction Fees

Partner: Trilemma Foundation

Jenny (Yuci) Zhang, Tengwei Wang, Ximin Xu, Yajing Liu

2025-06-25

Table of contents

1 Executive Summary

1.1 Subsection (Use Two Hashes)

2 Introduction

2.1 Subsection (Use Two Hashes)

2.2 Subsection (Use Two Hashes)

3 Data Science Techniques

3.1 Subsection (Use Two Hashes)

3.2 Subsection (Use Two Hashes)

To understand and anticipate Bitcoin transaction fee rate dynamics, we implemented a sequence of models. Each of the models were selected for its ability to address specific limitations observed in the previous ones. Below, we walk through these choices, results, and where each model fell short.

3.3 Dummy Model (Global Median)

We began with a simple dummy model that always predicted the global median transaction fee rate. Although it had no predictive power, it served as a baseline to measure improvements from more sophisticated approaches. This model completely ignored any temporal or external structure in the data, but offered a useful starting point to quantify how difficult the prediction task really was.

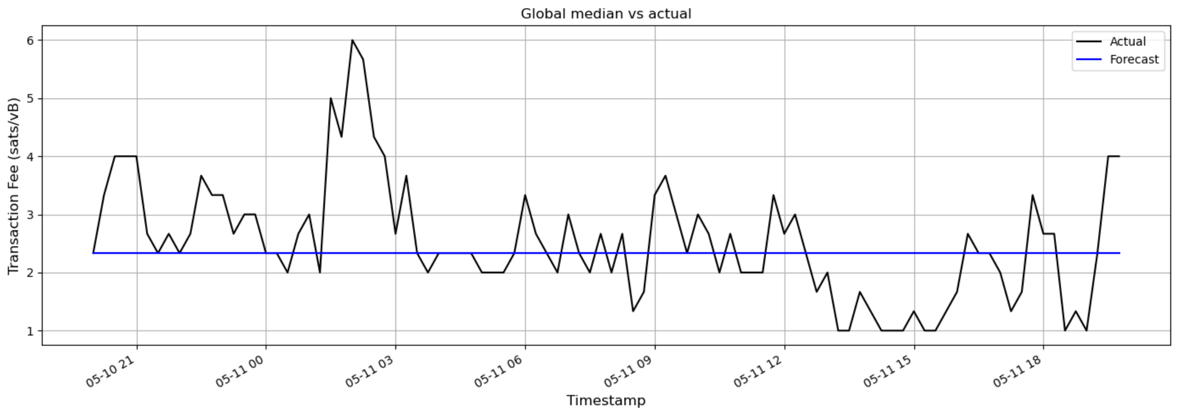

Figure 1: Global median vs actual values

3.4 Holt-Winters Exponential Smoothing (HWES)

Given the presence of regular daily cycles in the data, Holt-Winters Exponential Smoothing was a natural next step. It captured seasonality fairly well and improved upon the dummy model. However, analysis of the residuals revealed persistent autocorrelation, suggesting that the model failed to account for important lag effects or hidden patterns beyond periodic behavior.

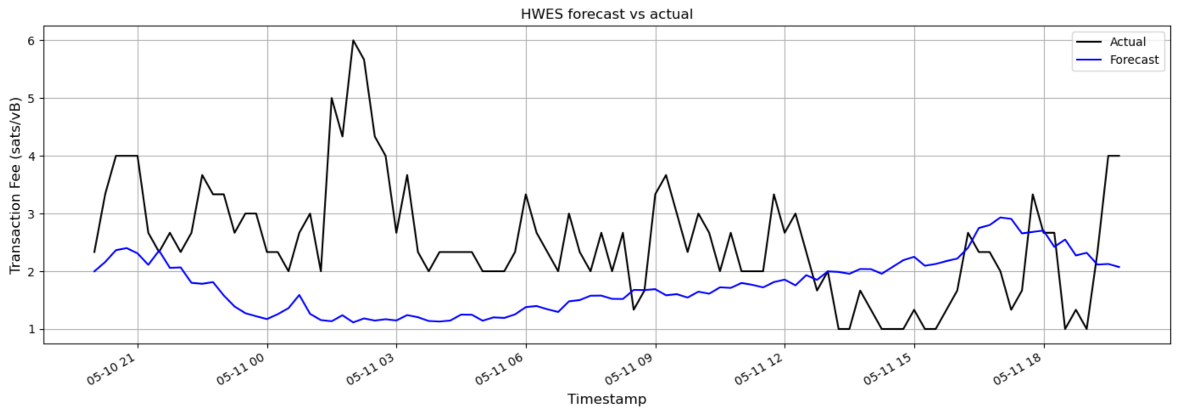

Figure 2: Forecasting results of HWES vs actual values

3.5 SARIMA

To address the temporal dependencies missed by HWES, we adopted SARIMA, which supported autoregressive and seasonal differencing components. Because of its capacity to learn from prior values, SARIMA produced a better short-term fit. However, its univariate nature prevented us from including important exogenous features like transaction count, mempool congestion, or size distributions. That limited its practical usefulness in a multi-variable environment.

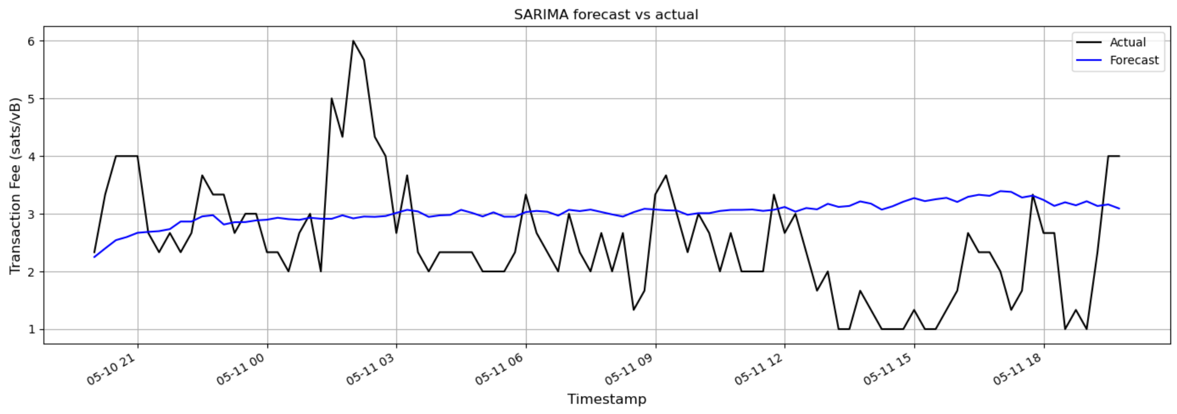

Figure 3: Forecasting results of SARIMA vs actual values

3.6 XGBoost

We then turned to XGBoost, a powerful tree-based model capable of incorporating a broad array of features. This model significantly expanded our input space and enabled non-linear interactions among variables. While its numerical performance improved, especially on average error metrics like MAPE, the model still struggled with volatility. It produced smooth and flat outputs that failed to capture sudden fee spikes. However, those spikes were precisely the events most critical for users.

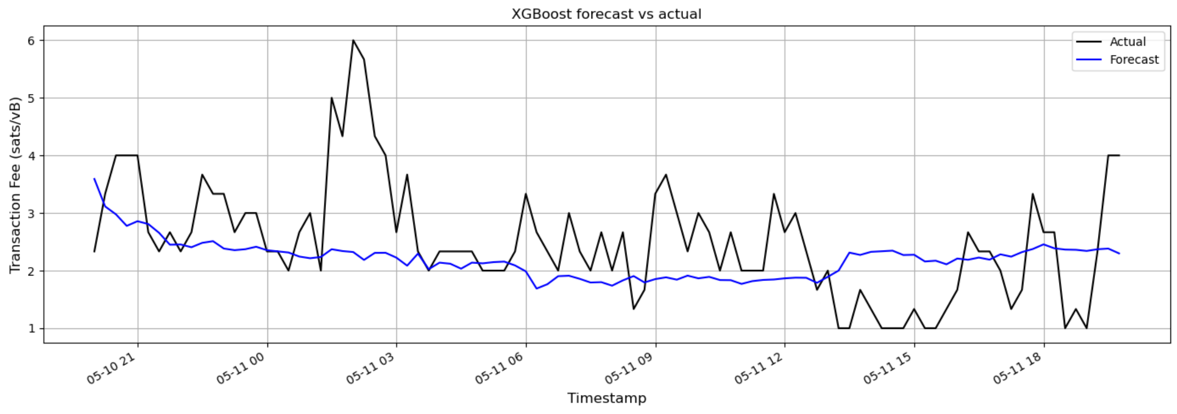

Figure 4: Forecasting results of XGBoost vs actual values

3.7 Prophet

We explored Facebook's Prophet model to take advantage of its flexibility in modeling seasonality, changepoints, and custom events. Prophet brought in useful priors for time series with irregular behavior and offered a simple interface for integrating domain knowledge. Unfortunately, it still smoothed over the spikes and underperformed in capturing real-time fee volatility. Its strength in trend estimation did not translate well to our highly reactive use case.

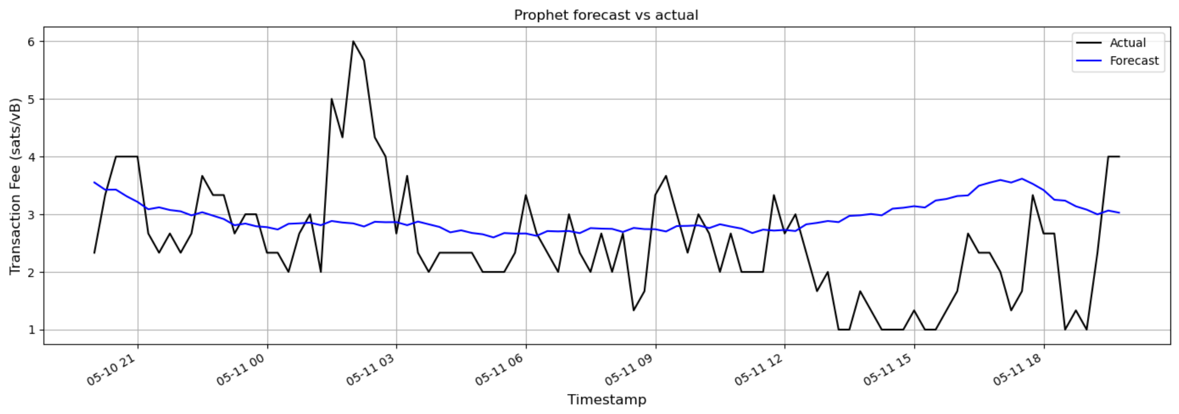

Figure 5: Forecasting results of Prophet vs actual values

3.8 DeepAR

To try more dynamic modeling, we implemented DeepAR, which was an LSTM-based autoregressive forecasting model. In theory, DeepAR should have leveraged temporal context more effectively and handled sequential data better. However, the outputs were unstable and noisy. The results often failed to align with real-world fee movements. The model demonstrated limited generalizability, and its probabilistic forecasts often appeared more random than informative.

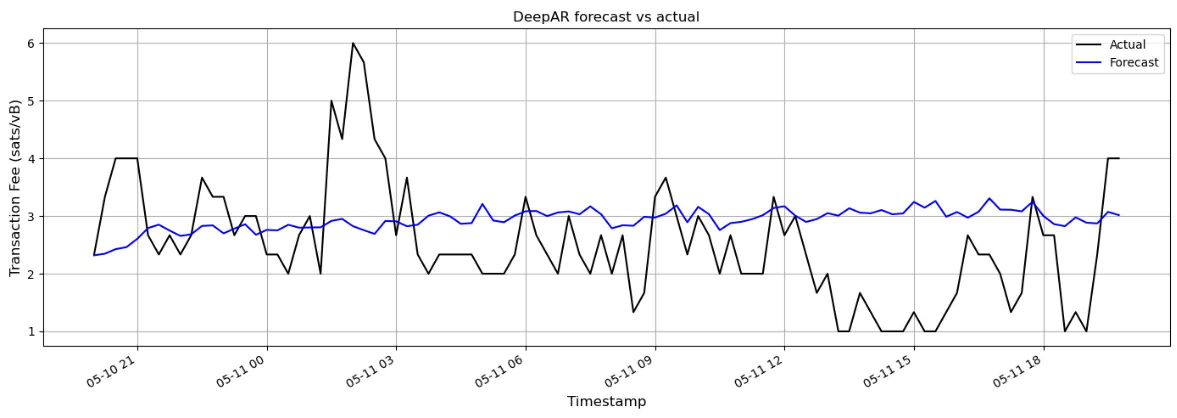

Figure 6: Forecasting results of DeepAR vs actual values

3.9 Temporal Fusion Transformer (TFT)

Our final and most advanced model was the Temporal Fusion Transformer(TFT). TFT was designed to integrate static covariates, time-varying features, attention mechanisms, and variable selection into a unified deep learning architecture. Among all models, TFT came closest to capturing both overall volatility and individual spike events. It successfully learned temporal dependencies, responded to feature relevance dynamically, and produced the most realistic forecasts. While computationally expensive and complex to tune, its performance and interpretability made it the strongest candidate for this forecasting task.

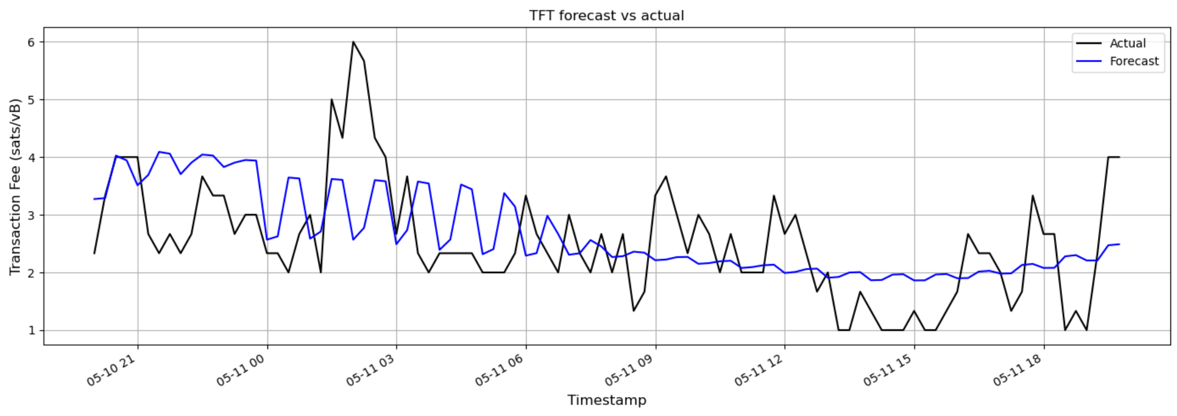

Figure 7: Forecasting results of TFT vs actual values

3.10 What Might Have Worked Better

3.10.1 Current Limitation

Most models react to spikes after they happen due to:

- Lagged exogenous features.
- Absence of true leading indicators.
- Volatile, abrupt, and irregular nature of spikes.

3.10.2 Potential Improvement

A more forward-looking architecture could be created via multi-stage forecasting, e.g.:

- Train models to predict key exogenous signals (e.g., mempool congestion, pending transaction counts).
- Use their predicted values as inputs into the main fee prediction model.

3.10.3 Why Not Implemented

- Partner's feedback during mid-project phase discouraged premature inclusion of predicted external signals.
- Additional models would introduce forecast compounding errors.
- Partner had tried social sentiment modeling in previous work (e.g., scraping tweets, news) but results were unsatisfactory.
- Team chose to focus on maximizing what could be done within the current data window and feature scope.

3.11 Subsection (Use Two Hashes)

4 Data Product and Results

4.1 Overview of the Data Product

This data product directly supports Trilemma Capital's mission of serving industry talent and advancing Bitcoin infrastructure through data science, both educational and technical. Its design intentionally tailors to three core audiences. General users and institutions can rely on the

24-hour forecasts to plan transactions and reduce fee costs. Learners and educators receive a transparent, step-by-step walkthrough of Bitcoin-fee forecasting and time-series methodology. Industry experts and partners see infrastructure-grade modeling practice embodied in a modular pipeline and well-documented repository. The product is purposefully modular. Jupyter notebooks guide users through EDA, modeling decisions, and final TFT results. Python scripts implement a structured pipeline for reproducible experiments and easy re-training on new data. Finally, the open-source GitHub repository—with clear documentation—enables collaboration, scalability, and long-term extensibility.

4.2 Results

5 Conclusion and Recommendations

5.1 Subsection (Use Two Hashes)

5.2 Subsection (Use Two Hashes)

6 Appendix

6.1 Terminology

Term	Definition
Bitcoin	Unit of currency is called ”bitcoin” with a small b, and system is called ”Bitcoin,” with a capital B. ”bitcoin” is a virtual currency (cryptocurrency) designed to act as money and a form of payment outside the control of any one person, group, or entity (i.e. decentralized).
Bitcoin Address	“1DSrfJdB2AnWaFNgSbv3MZC2m74996JafV” An encoded base58-check version of a public key 160-bit hash consists of a string of letters and numbers. Think of it analogous to an email address when sending someone an email.
Blockchain	A decentralized digital ledger that records transactions across a network of computers, making it transparent, immutable, and resistant to tampering. Technology used by Bitcoin.
Fees	The sender of a transaction often includes a fee to the network for processing the requested transaction. Most transactions require a minimum fee of 0.5 mBTC (millibitcoin) = 0.0005 BTC. Typical unit measurement in satoshi/bytes.
Hash	A function that converts an input of letters and numbers into an encrypted output of a fixed length. The hash is irreversible, meaning it cannot be decrypted back to the original input. Hashes are used in Bitcoin to create blocks and verify transactions.
Mempool	The bitcoin Mempool (memory pool) is a collection of all transaction data in a block that have been verified by Bitcoin nodes, but are not yet confirmed.
Mining / Miner	A process/network node that finds valid proof of work for new blocks, by repeated hashing.
Node	Refers to blockchain stakeholders and their devices that keep a copy of the distributed ledger and serve as communication points within the network. Major purpose is to verify the validity of the transactions within a particular blockchain.
Proof-of-Work	A piece of data that requires significant computation to find; In bitcoin, miners must find a numeric solution to the SHA256 algorithm that meets a network-wide target, the difficulty target.

Satoshi	The smallest denomination of bitcoin that can be recorded on the blockchain. 1 Bitcoin is equivalent to 100 million satoshis, named after the creator of Bitcoin, Satoshi Nakamoto.

Table 1: Key Terms and Definitions in Bitcoin and Blockchain (Alphabetically Ordered)

7 References