Proposal Report: Forecasting Bitcoin Transaction Fees

Partner: Trilemma Foundation

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

2025-05-09

Table of contents

Introduction **1**

Data Overview **1**
Data Description 2

Data Science Techniques **3**
Data Preprocessing 3
Forecasting Methodology 4

Workflow & Timeline **5**
Workflow 5
Timeline 6

References **6**

Introduction

Data Overview

---

title: Data Overview jupyter: jupytext: text_representation: extension: .qmd format_name: quarto format_version: '1.0' jupytext_version: 1.16.4 kernelspec: display_name: Python 3 (ipykernel) language: python name: python3

We use Bitcoin Mempool data, which records hourly time series information about unconfirmed transactions waiting to be confirmed in the blockchain. The categories of key features include projected data, fee rates, mempool state, and difficulty adjustment. This rich dataset captures the real-time dynamics of network demand and transaction prioritization. It provides the necessary historical context for building accurate fee rate forecasting models.

Figure 1: Real-time data of Bitcoin Mempool.

Data Description

Our dataset contains 11,902 hourly entries and 67 features across several categories relevant to fee prediction, including recommended fee rates, mempool statistics, detailed fee histograms, mining difficulty adjustments, and price data in various fiat currencies. The projected data contains projected mempool blocks, transaction sizes, and their associated fee ranges. The fee rate features provide various types of recommended fees for different confirmation targets, including fastest, half-hour, hour, minute, and economy rates, along with 37 binned fee categories based on transaction counts. The mempool state captures aggregate statistics such as the total transaction count, virtual size (vsize), and total fees observed at each time step. Lastly, the difficulty adjustment data includes information about the current difficulty epoch,

such as its progress percentage and the estimated time to the next retarget event. Below is the table including the key categories of features.

Table 1: Data descriptions by categories.

Category	Description	Columns	Range
Projected Data	Projected mempool blocks, transaction sizes and their fee ranges.	5	Varies by feature and type.
Fee Rates	Different types of recommended fee rates and confirmation targets. Fees are also binned by counts.	5 fee rates and 37 binned	Min: 1 sat / vbyte, Max: 41 sats / vbyte, Median: 2-3 sats / vbyte
Mempool State	Aggregate mempool statistics (e.g., total vsize, count).	3 (count, vsize and total_fee)	Count: 24 - 169, vsize: 1.2e4 - 5.6e7 vbytes, Fee: 1.2e4 - 2.3e8 sats
Difficulty Adjustment	Data related to the current difficulty epoch (progress%, estimated retarget)	10	Varies by feature and type.

Data Science Techniques

Data Preprocessing

Understanding and preparing the data is critical for building reliable models of Bitcoin transaction fees. Several challenges arise during this process. First, we detected outliers that may represent data entry errors or anomalies, such as BTC prices recorded as -1. Second, documentation from the data source is limited. Third, transaction behavior is expected to vary over time, with differences across weekdays and weekends, as well as possible seasonal patterns. Finally, relationships between variables like transaction volume and fee rates may hold important signals that require further exploration.

To address the issues, we begin by identifying and correcting outliers and abnormal values through data cleaning. We also inspect feature definitions and cross-validate them to understand how certain features are computed. Temporal patterns are explored by introducing time-based features such as hour-of-day and day-of-week. We further analyze the data for signs of seasonality, subject to the period covered. To capture relevant relationships, we compute correlations, for example, between transaction volumes and fee rates or between median and total fee rates, to uncover meaningful relationships. We also consider engineering new features, such as binary indicators for positive or negative network events, to capture additional signals relevant to fee dynamics.

We select the "fastest fee" rate as the primary response variable, as it serves as a clear benchmark for users prioritizing transaction speed. Given the right-skewed nature of the fee distribution, we apply a logarithmic transformation to improve model performance. We address missing and abnormal values through data cleaning procedures. Finally, we engineer time-based features—such as hour of day and day of week—as well as indicators of mempool congestion to enrich the feature space for downstream modeling.

Forecasting Methodology

The overall feature structure reflects both temporal dependencies and external influences, such as time-of-day effects, network congestion, and market conditions. Since our data contains both time-series patterns and contextual signals from network and market activity, we deliberately combine autoregressive, regression-based, and deep learning models to address different aspects of this complexity.

To tackle the challenges of predicting Bitcoin fee rates—like changing patterns over time, external factors, and non-linear behavior—we start with a few simple but solid baseline models. To understand how things like mempool congestion or the day of the week affect fees, we use **Linear Regression**. It's not meant for forecasting, but it gives us a clearer idea of which external factors are actually influencing the fee rates. Then, to capture how fee rates depend on their own past values over time, we use **ARIMA**, which is well-suited for modeling time-based patterns and serves as a good starting point for temporal forecasting. Since fee rates often show short-term trends and recurring daily patterns, we use **Holt-Winters Exponential Smoothing (HWES)** to capture these seasonal behaviors without needing additional input features. Finally, to deal with more complex nonlinear interactions, we bring in **XGBoost**, a powerful tree-based model that works well with structured data and can handle both historical and external features.

As the project moves forward, we need models that can better deal with the complexity and unpredictability of the data. For example, we want to understand broader patterns like seasonal cycles with holiday effects in fee rates. That's where **Prophet** comes in—it breaks the time series into interpretable parts like trend and seasonality, giving us a clearer picture of the overall structure. While it's more suited for daily data and doesn't respond well to sudden changes, it's still helpful for spotting big-picture patterns. We also need a way to model multiple related time series—like our detailed fee histogram bins—while accounting for uncertainty. **DeepAR** is a good fit for this, since it learns from sequences and produces probabilistic forecasts, which is especially useful in such a volatile environment. Lastly, to make sense of all the moving parts in our high-dimensional dataset, we use the **Temporal Fusion Transformer (TFT)**. It can focus on the most important features at each point in time using attention and variable selection, making it powerful for capturing complex relationships. That said, it's a heavy model that takes more computation and needs careful tuning to avoid overfitting.

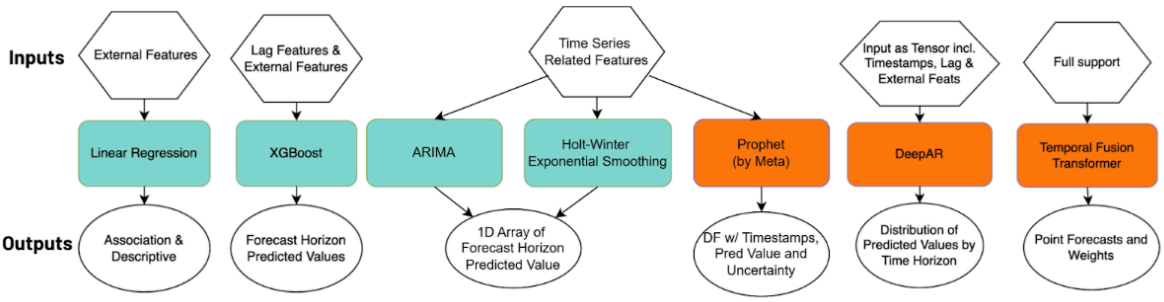

Figure 2: Diagram of model inputs and outputs for baseline and advanced forecasting models.

To assess model performance, we use **Mean Absolute Error (MAE)** and **Root Mean Squared Error (RMSE)** as our primary evaluation metrics. MAE provides an interpretable average of prediction errors in the same unit as the target variable, while RMSE penalizes larger deviations more strongly, making it useful for identifying models that are sensitive to extreme fluctuations. For probabilistic models like DeepAR and TFT, we also evaluate the **calibration of predicted confidence intervals** to ensure that they reliably reflect observed variability in fee rates. This is critical for making informed, risk-sensitive decisions in high-variance network environments.

Workflow & Timeline

Workflow

The workflow diagram below outlines the end-to-end structure of our project. We begin by collecting raw time series data from the Bitcoin mempool, which is then processed and explored during the EDA phase. After feature engineering, we build baseline models to establish initial benchmarks. We then transition to advanced models for improved performance and uncertainty estimation. Once the best-performing models are selected, we evaluate them based on success criteria including prediction accuracy and interval reliability. We follow this with timeframe optimization to determine which forecast intervals are most dependable. Finally, we deploy the selected model to AWS for real-time fee prediction.

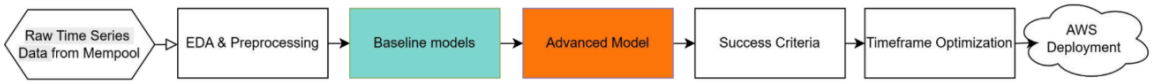

Figure 3: Workflow of the modeling pipeline from data preprocessing to deployment.

Timeline

Week	Milestone
Week 1	Write proposal report, conduct initial EDA, and create a full data specification notebook to document and visualize all available features and the target.
Week 2	Train and evaluate baseline models, including ARIMA, Holt-Winters, and XGBoost; use Linear Regression for exploratory analysis of external features.
Week 3	Build advanced models such as Prophet, DeepAR, and Temporal Fusion Transformer.
Week 4	Continue building advanced models if needed; test and compare their performance against baselines using standard metrics.
Week 5	Perform timeframe optimization by analyzing model performance across the 24-hour forecast horizon.
Week 6	Integrate modeling pipeline, finalize deliverables, and prepare for final presentation and AWS deployment.

References