Impact of Liquidity Pool Size on Trading Volume in BTC-ETH Pools

Team #111
Matias Vizcaino (avizcaino3) — Walter Jack Simmons (wsimmons35) — MingShen Wang (mwang709) — Vítor de Matos Castilho (vcastilho3)

09 July 2023

Problem Statement and Objective

Decentralized Finance (DeFi), hosting about 48.78 billion USD in total value (as of April 23, 2023[2]), relies heavily on liquidity pools in decentralized exchanges like Uniswap for efficient crypto trading. These pools, using an automated market maker model, earn from trading fees, participation rewards, and interest, distributed among liquidity providers based on their pool share[9, 5, 12].

This study primarily **aims to examine the influence of pool size on trading volume, specifically in BTC-ETH liquidity pools in exchanges like Uniswap**. It hypothesizes a direct correlation between pool size and trading volume, supported by existing studies[9, 7]. This understanding could optimize liquidity provision, enhance DeFi market efficiency, and benefit stakeholders. For instance, liquidity providers could improve fee returns and risk management, traders could refine strategies, and DeFi platforms could develop more effective liquidity pools[8, 10].

Uniswap Liquidity Pools: An Overview

Various DeFi platforms cater to different needs. Uniswap, a popular choice, provides a user-friendly interface and a broad spectrum of token pairs. Other platforms like Curve Finance and Balancer offer low-slippage trades and customizable pools, respectively[12].

Uniswap, a decentralized exchange (DEX) on Ethereum, supports trading of Ethereum-native and wrapped non-native assets. It uses a Constant Product Market Maker (CPMM) model in which liquidity providers deposit two tokens, keeping a constant product of reserves[10, 7, 10].

Uniswap v3 introduced the concept of "virtual" versus "real" reserves, where virtual reserves behave as if all liquidity is concentrated at the current price range [6]. The unique mechanism allows more efficient capital usage and offers liquidity providers the ability to customize their price exposure. Precise calculations of position holdings and pool's parameters are possible through the equations provided by the Uniswap v3 liquidity math paper [6, 10]. Uniswap V3's liquidity pools and the constant product market maker (CPMM) model are at the heart of the study of BTC-ETH pools. Research conducted in this area provides key insights and methodologies for predicting trading volume based on various activities within these pools [10]. A detailed understanding of decentralized finance (DeFi) and how it differs from traditional financial structures is crucial in order to fully comprehend the mechanics of automated market makers (AMMs) and the operation of liquidity pools [8]

The CPMM formula is given by:

$$x \cdot y = k \quad \text{and} \quad price = \frac{y}{x}$$

Here, x and y denote the quantities of Token X and Y in the pool, and k is the constant product[8, 10].

Uniswap operates without a central authority, ensuring privacy, fund control, and a diverse range of tokens[10]. Centralized exchanges (CEX), though offering faster transaction speeds and better customer support, may pose security risks as they hold user funds[8]. Liquidity pools in DEXs enable direct interaction with smart contracts, ensuring continuous liquidity but exposing liquidity providers to impermanent loss risks[5, 7]. Prices are influenced by trade size, market conditions, supply-demand dynamics, and external price changes[9, 7, 10].

Analytical Methodology and Achievements

A thorough analysis of the interconnectedness between Uniswap v3 pools and a systematic selection of subsets of pools for analysis is required. This will be done over different time periods, using similar methodologies as found in recent studies [9]. The relationship between liquidity, trading volume, and price stability in pools will be examined, along with an investigation of how liquidity pool size impacts trading volume.

Feature Engineering: Our models' predictive power was enhanced by creating new features from existing data[10]. This process effectively captured crucial information about liquidity pool dynamics, spillover effects, and price divergences.
Modelling and Optimization: Through the application of Ordinary Least Squares (OLS) regression, we examined the relationship between liquidity pool size and trading volume, which offered profound insights into their interplay[10]. Model performance was assessed using metrics such as R-squared, while we optimized feature selection, corrected multicollinearity, and explored alternative functional forms for improved model accuracy and interpretability[10].
Innovations: Building on the reference paper, we implemented the underlying code in Python, thus creating a robust data model to explore interactions between two liquidity pools more effectively[10]. Furthermore, we developed a mathematical language to facilitate the data engineering aspects of the data model[10].

Table 1: Analytical Techniques and Main Contributions

Substantial accomplishments were achieved at each of our research stages, leading to a robust analysis and valuable results[10].

Data Collection: We sourced data from multiple APIs (Uniswap, Binance) successfully, and associated this data with Ethereum block numbers for consistent time measurement.
Data Preprocessing: We maintained data consistency, addressed discrepancies from different sources, and efficiently cleaned, formatted, and handled missing values in the dataset.
Exploratory Data Analysis (EDA): Utilizing EDA, we generated descriptive statistics, visualizations, and correlation analyses. This approach facilitated the identification of trends, outliers, and the relationships between variables.
Final Analysis and Conclusions: We articulated our findings and conclusions, providing quantitatively-driven insights with the potential to enhance liquidity provision, optimize trading strategies, and improve the efficiency of DeFi marketplaces.
Confirmation of Results: Our results largely coincide with those in the reference paper, with minor discrepancies likely due to the techniques used and features available. Notably, we utilized fewer features and feature categories than the reference paper, which might explain the minor divergences observed in our results[10].

Table 2: Supporting Work and Confirmation of Results

Our work provides a significant contribution by offering an open framework that allows other researchers and professionals to conduct further research in this field. This outcome not only answers our problem statement but also opens avenues for future exploration[9].

Challenges and Insights

Our main challenges were data collection and processing due to intricate engineering and the rate limits of the Etherscan free-tier API[3]. Nonetheless, we overcame these hurdles and made significant progress in understanding liquidity pool dynamics[9, 5, 10].

We embraced two time concepts—"trading clock" and "time horizon"—which enabled us to thoroughly explore both short and long-term trends. This detailed analysis offered invaluable insights for decision-making and risk management in the DeFi ecosystem[9].

We successfully addressed complexities in understanding entry-level liquidity pool mathematics and integrated the associated formulas into calculations.py functions. This step significantly enriched our feature derivation and bolstered our overall analysis[6, 5]. For instance, we tackled the complexity of tick math, which involves mapping discrete ticks to continuous prices. This step, guided by the Uniswap v3 liquidity math paper, was essential for interpreting Uniswap data and indexes [6]. In addition, we adjusted raw price ratios for token decimals to get human-readable prices, a process critical for presenting our analyses [6].

[Place this into our context: It is also important to acknowledge the economic and security risks inherent to these protocols. Economic risks such as yield dilution, conversion risk, exchange risk, counterparty risk, and liquidation risk all pose challenges to the sustainability and effectiveness of yield farming [12]. Moreover, security

risks, including flash loan attacks, rug pulls, reentrancy, and key exploits are also prevalent in the yield farming landscape [12]]

Key Note:

- Our main focus during the development process was programming and data structure, leading to our code's modular design. Consequently, the mathematical formulas used in the feature engineering stage were not rigorously tested or based on an in-depth understanding of the underlying liquidity mathematics. There is room for improvement, and we encourage peer-review of the calculations.py module. The modular nature of our model presents a prime opportunity for further research[6, 10].
- Our models, although providing useful insights, might benefit from a more detailed understanding of the underlying liquidity mathematics, as provided by the Uniswap v3 liquidity math paper [6]. We encourage rigorous testing and peer review of our calculations.py module to ensure accuracy and completeness of our results[6, 10].

Dataset

In line with the *"DeFi modeling and forecasting trading volume" (2023)* study[10], we gathered and structured trade data spanning a minimum of 6 months. Our GitHub repository provides more details and the code used to gather this information[11]. Our dataset incorporates information from the following sources:

Source	**Description**	**Data**
Uniswap's The Graph API[4]	This API offers transaction details, trading volumes, and block data from Uniswap v3 liquidity pools.	Includes transaction IDs, timestamps, amounts, USD equivalents, and other related data.
Etherscan API[3]	Utilized to extract corresponding transaction data from Etherscan based on transaction hashes.	Block hashes, block numbers, sender addresses, gas details, transaction hashes, and other relevant information.
Binance[1]	Provides Centralized Exchange (CEX) data for ETHBTC trades. Data was obtained via scripts from Binance's GitHub repository.	Contains detailed data about each trade performed on the Binance platform, including trade prices, quantities, timestamps, and buyer/seller characteristics.

Table 3: Dataset Source Description

Key Variables

Target Variable: The key dependent variable in our study is the trading volume of liquidity pools (amountUSD) over specific blocks. We propose several models for different time horizons to investigate how the relationships evolve over time.

Independent Variables: The independent variables are constructed from various features and are grouped as follows:

1. **Direct Pool Features (41):** These include measures such as volatility, rate, number of trades, and average trade size.
2. **CEX Spillover Effects (6):** This category includes variables like actual coin trade volume on Centralized Exchanges (CEX), which can influence liquidity pools on decentralized platforms.

Engineering Ecosystem

Data engineering and feature extraction are crucial elements for building robust predictive models. In the case of analyzing BTC-ETH trading volume on Uniswap, several aspects of data must be considered.

First, let's consider the nature of the data we are dealing with. Uniswap is a decentralized exchange that operates on an automated market-making (AMM) model, and its Version 3 (v3) utilizes a novel mechanism of concentrated liquidity pools. The structure and dynamics of these liquidity pools play a crucial role in trading volumes [10]. We must extract relevant data from these pools, including their sizes, trading volumes, and related metrics, for accurate analysis. Understanding DeFi's goals and mechanisms from a broad perspective is also valuable [8].

Moreover, it's important to incorporate data from different sources, considering not just the pools of interest but also neighboring pools and other prominent exchanges such as Binance. This kind of broad-spectrum data helps in capturing spillover effects, which are significant drivers of trading volume [10]. For example, we can study the effects of trading volume and the activities from interconnected DeFi protocols and stablecoins [9].

Feature selection is a crucial part of the process. Stepwise regression can be an effective tool for optimizing the model and refining the set of independent variables [10]. We should also account for factors like network effects, economies of scale, and concentration issues that can influence liquidity provision [8, 9].

From a risk perspective, we should account for external risks to liquidity providers like arbitrage opportunities, fraudulent DEXs, regulatory risks, censorship risks, etc. [5]. Systemic risks from interconnected DeFi protocols and stablecoins could also impact trading activity [8]. Additionally, economic risks like yield dilution, conversion risk, exchange risk, counterparty risk, and liquidation risk pose challenges and should be considered [12].

For the implementation part, we can take advantage of mathematical equations and formulations. For example, we can use equations to calculate the amounts of assets based on liquidity, price range, and the current price [6]. Formulas for calculating impermanent loss and changes in holdings after a price change could also be useful in our analysis [6, 5, 7].

In-Depth: Feature Engineering

Feature engineering plays a crucial role in capturing the dynamics and relationships between variables in our analysis. The focus lies on engineering features that capture historical patterns of the DEX pools and spillover effects from CEX activity.

To start with, we can first define all the blocks in scope, then move to define the Reference blocks, and finally, the interval between reference blocks (which will be all the blocks in the interval).

Let's assume B is the entire sequence of blocks within the scope of analysis and K is the total number of blocks under study. We can represent this in the following way:

$$B = \{b_k\}_{k=1}^{K} \text{ where each } b_k \text{ is a block within the scope of analysis} \tag{1}$$

In the construction of a time-series data model, the "Reference Block" acts as a chronological reference point. These blocks are key markers within the blockchain, identified based on Mint Operations when users provide liquidity to different pools on the decentralized exchange (DEX), such as Uniswap. The Reference Blocks are categorized into "same" and "other" pools. We position the most recent at the top and with an index of 0, so the sequence of blocks is ordered in decreasing chronological order.

$$R_s = \{b_s\}_{s=1}^{S} \text{ where each } b_s \in B \text{ and is a reference block in the "same" pool} \tag{2}$$

$$R_o = \{b_o\}_{o=1}^{O} \text{ where each } b_o \in B \text{ and is a reference block in the "other" pool} \tag{3}$$

Here, S and O are the total number of reference blocks in the "same" and "other" pools respectively.

Finally, the intervals between these reference blocks, which contain all blocks between b_i and b_j (inclusive of b_i), can be defined as the sequence of block numbers between b_i and b_j. This can be expressed as:

$$I = \{b_{ij}\}_{i=0, j=i+1 \ i,j \leq N}^{N} \tag{4}$$

In this formula, b_{ij} represents the sequence of blocks from b_i to b_{j-1}, inclusive of b_i and exclusive of b_j. This correctly captures the notion of an interval being all blocks "between" b_i and b_j in the blockchain sequence.

Block Reference Interval chains for the "same" and "other" pools' mint operations generate sequences of block numbers. These chains include the last N reference blocks and every block in between, where N signifies the number of lags or intervals analyzed for the direct features. Let C_s and C_o be the chains of "same" and "other" pools, which can be mathematically defined as:

$$C = \{b_i\}_{i=0}^{N} \text{ where each } b_i \text{ is a block} \in \{R_s, R_o\} \text{ and } b_i > b_{i+1} \text{ for } i \in \{0, 1, ..., N-1\} \tag{5}$$

Time horizons, representing future periods or timeframes for predictions or analyses of target variables, are constructed by starting from the first Reference Block and increasing the block count by M until the next reference block is reached. M represents the number of blocks or periods between reference points in constructing time horizons, influencing the construction of the dependent variable. Choosing an appropriate value for M is essential in determining the length and granularity of the time horizons and impacts the level of detail and prediction accuracy in the analysis or predictions.

$$H_M = \{b_0 + kM\}_{k=0}^{K} \text{ with } K = \min\left(K_{\max}, \left\lfloor \frac{\text{distance to next reference block}}{M} \right\rfloor\right) \tag{6}$$

The Horizon Block Table captures the dynamics and relationships between the same/other/both pools by evaluating their respective target variables and features at different horizons and lags. The granularity level varies for each component within the Horizon Block Table, as illustrated in the following table:

$$HBT = \{(H_M, F_{\text{DEX}}, F_{\text{DP}}, F_{\text{CEX}})\} \text{ for each horizon } H_M \tag{7}$$

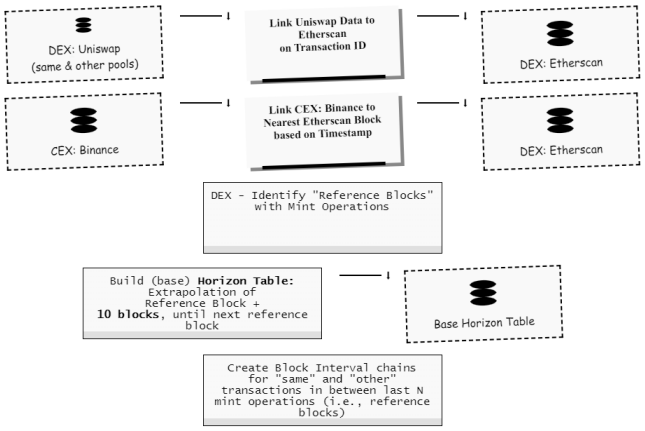

Figure 1: Data Engineering

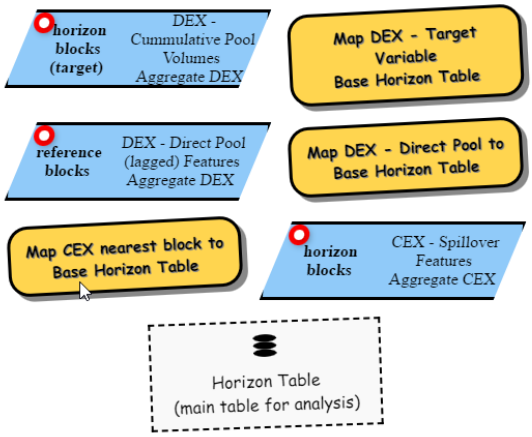

Figure 2: Feature Engineering

Component	Granularity Level
DEX target variable	Horizon block level
DEX direct pool features	Reference block level
CEX spillover features	Horizon block level

Table 4: Granularity Level of Components in Horizon Block Table

This granularity distinction allows for comprehensive examination of the behavior and interactions of the "same" and "other" pools at different levels within the time-series data model. The Horizon Block Table provides insights into the dynamics of the analyzed data, revealing the collaborative or competitive dynamics between the "same" and "other" pools throughout the time series.

TODO -¿ Table of the values we use on our project. Mention that the combination provides a rich set of models for analysis

Data Profiling Report

This report provides an analysis of our data profiling exercise, focusing on block intervals, transaction types, reference blocks, and target variables. We aim to develop a comprehensive understanding of our dataset, paving the way for more robust model fitting and further analysis.

The data merging process between Uniswap transactions and Etherscan demonstrated a high match rate of transactions for the six-month analysis period, affirming the integrity of our dataset. Further cleaning and integrity checks were performed on each of the sources for facilitating subsequent processing.

0.1 Block Intervals and Transactions

The dataset encompasses block intervals ranging from 0 to 3, where block time refers to the number of blocks between reference blocks that carry out a mint operation, in both the same and other pools. A pattern of increasing block time and transaction count is observed as we transition from the most recent to preceding intervals. The maximum block time recorded is 8,497 for the 500 pool and 8,460 for the 3000 pool, both at intervals 2 and 3. The maximum row count approximates 1,115, indicating substantial data captured within these intervals.

Transaction data, classified as 'burns', 'mints', and 'swaps', reveals 'swaps' to be the prevalent transaction type across all dates and both the 500 and 3000 pools. However, the volume of each transaction type exhibits variations over time, illustrating the dynamic nature of the market. The 500 pool consistently records higher transaction volumes across all types and dates, implying a greater liquidity or popularity of the 500 pool. The number of burns and mints transactions generally balance, signifying an equilibrium between supply inflow (mints) and outflow (burns), with occasional deviations.

0.2 Reference Blocks

Reference blocks form a critical component in our analysis. Our dataset accommodates different numbers of reference blocks for the 500 (3,242) and 3000 (4,986) pools, with each block serving as a reference point for recording block

time and transaction count. An unusual surge in mint activity recorded in July 2022 merits further investigation to uncover potential market influences or events during that period.

0.3 Target Variables

The target variables in our dataset comprise 'cum_volume_500', 'cum_volume_3000', and 'cum_volume_base' for the 500, 3000, and both reference pools, respectively. These variables encapsulate cumulative volumes across different pools and underpin our forthcoming modeling and analysis.

0.4 Data Cleaning and Preprocessing

Prior to the modeling phase, it was essential to address records with missing values in the independent variables. Specifically, we identified null values in the volatility (same pool) and avg-USD/rate-USD (other pool) metrics. We deployed Python utility functions customized for Pandas DataFrame to conduct the initial data cleaning and preprocessing. These tasks comprised identifying and addressing missing values, eliminating correlated columns, and aggregating data at specific intervals.

- **Handling Missing Values:** [CHECK THIS SECTION] Certain columns such as 'rate-USD-iother_01', 'avg-USD-iother_01', 'vol_0_1', 'vol_0_2', 'vol_0_3' contained null values, which we replaced with zeros to render the dataset suitable for model fitting. However, a few columns like 'slother_3', 'rate-count-iother_23', 'wlother_3' continued to accommodate a significant number of null values. Further decisions regarding these null values remain unindicated.
- **Correlated Features:** [Insert Matrix] We identified columns such as 'blsame_2', 'blsame_3', 'blother_2', 'blother_3', 'vol_0_1', 'vol_0_2' as highly correlated, leading to their removal from the dataset to prevent multicollinearity during model fitting.
- **Column Aggregation:** We aggregated certain columns that shared prefixes like 'rate-count-isame_', 'rate-count-iother_', 'binance-btc-', summing up their data in new columns appended with '03'. We subsequently dropped the original columns.
- **Data Loss Assessment:** We meticulously tracked the change in row count during the data cleaning process, which led to an insignificant data loss of 0.09% observations, denoting a successful data cleaning operation with the majority of the data preserved.

0.5 Data Preparation for Model Fitting

We prepared three Horizon Tables (refbase, ref500, ref3000) for each mint of reference, leading to a total of 118,785 horizons. Before proceeding to fit the Ordinary Least Squares (OLS) models, we limited the horizons to a maximum of 30, which with K=10 it corresponds to about 300 blocks and $\sim$2.3 minutes (given a mean block time of 14 seconds). This step allows us to focus on the most immediate blocks, thereby reducing noise and enhancing the robustness of our modeling process.

In conclusion, our detailed data profiling has enabled us to unearth critical insights into block intervals, transaction types, reference blocks, and target variables. Despite certain challenges such as missing values and high correlation among variables, strategic management of these issues ensures minimal data loss. This thorough comprehension of our data lays a robust foundation for the subsequent stages of our data exploration and analysis pipeline.

In-Depth: Modelling Approach

Our study aims to investigate the relationship between liquidity pool size and trading volume using regression models. Specifically, we employ an Ordinary Least Squares (OLS) regression approach.

The OLS regression model is suitable for this study due to its simplicity, interpretability, and wide usage in econometric analysis. While other models like Poisson regression, negative binomial regression, or logistic regression could be considered, these are more suited for count data or binary outcomes, rather than our continuous outcome of interest. In addition, through OLS we can compare our analysis with other academic work (reference), which is imporant given the lack of research on this area and the novelity of approaches.

The application of OLS to time-series models requires additional considerations due to the sequential nature of time-series data. Traditional regression techniques can fail to account for this temporal element. Hence, we have used methods that incorporate time trends and seasonal patterns. However, we can use OLS regression if we have reason to believe our time-series data follows a linear trend, bearing in mind issues such as autocorrelation and non-stationarity.

0.6 Ordinary Least Squares (OLS) Regression Method

Ordinary Least Squares (OLS) is a technique used in linear regression to estimate unknown parameters. The method works by minimizing the sum of the squared residuals. In this context, residuals are the differences between the observed and predicted values. The objective function to minimize is expressed as follows:

$$\min_{\beta} ||Y - X\beta||^2 \tag{8}$$

where: - Y is the dependent variable - X is the matrix of independent variables - β is the vector of parameters to estimate.

0.7 Model Structure and Variable Definitions

We define a set of pools, denoted as P, where $p \in P = \{500, 3000, \text{both}\}$. This accommodates individual, interaction and aggregate metrics within and across two pools, enabling comprehensive study and insights into spillover effects.

A reference pool P_r is introduced, with $P_r \in P$ and $P_r \neq$ both, denoting a pool where a reference mint operation occurs. There are thus $len(P) * len(P_r)$ possible combinations (i.e., 3*2=6).

The response variable in our model, $Y_{p_{\text{horizon}}}$, denotes the cumulative volume for a pool at a given horizon. The explanatory variables are structured to discriminate "same" and "other" pool features using Block Interval Chains.

The OLS regression model used in our study is expressed as follows:

$$Y_{p_{\text{horizon}}} = \beta_0 + \beta_1 \cdot \widetilde{01}X + \beta_2 \cdot \widetilde{12}X + \beta_3 \cdot \widetilde{23}X + \ldots + \epsilon \tag{9}$$

In this equation, β_0, β_1, β_2, β_3, etc., represent the coefficients associated with each spot lagged variable, while ϵ is the error term or residual.

0.8 Model Testing and Performance Metrics

We constructed and evaluated OLS regression models for different target variables and horizons. Lagged variables were incorporated into the models to predict trading volumes at different time horizons. The models' performance was measured using metrics like R-squared, adjusted R-squared, mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE). We limited our horizon analysis to 30, equivalent to 300 blocks.

0.9 Train/Test Split and Cross-Validation

The Train/Test Split methodology assessed the model's ability to generalize to unseen data. Our data set was partitioned into a training set for model training and a test set for evaluating predictive performance.

Cross-validation was applied to assess the OLS model for each horizon. We employed GroupKFold, a form of k-fold cross-validation, to prevent data leakage between training and testing sets. This ensured groups were not split between these sets. We defined groups according to the block reference number. For each fold, the data was split, features standardized, target variables transformed, and an OLS regression model fitted. R-squared and adjusted R-squared were computed for both training and test sets. This process was repeated for each horizon, resulting in trained models and performance metrics. Cross-validation aids in estimating model performance, identifying overfitting, and ensuring reliable generalization across different data subsets.

1 Model Optimization and Selection

The coefficient of determination, also known as R2 score, quantifies the amount of variance in the dependent variable predictable from the independent variable(s). It ranges from 0 to 1, where 0 means the model explains none of the response data's variability around its mean, and 1 means it explains all the variability. However, R2 can be negative when the model fits poorly and performs worse than a horizontal line (the mean of the dependent variable). This may occur when the model's predictions are consistently far off, causing large errors, or when overfitting occurs on the training data, resulting in poor performance on the test data.

Our iterative "All Horizon Runs" and "Best Horizon Run" approaches allowed us to explore different modeling techniques and optimize the models based on performance. The results informed the selection of the most suitable models for predicting the target variables.

1.1 Model Training for All Horizon Runs

The initial phase of the model training, termed as "All Horizon Runs", was directed towards training Ordinary Least Squares (OLS) models across all horizons. The models utilised target variables with data preprocessing being performed beforehand. The model performance was examined through a 5-fold cross-validation strategy, deploying

metrics such as R-squared, adjusted R-squared, and Mean Squared Error (MSE). A variety of visualization tools facilitated the comparison of models' performance over different horizons and target variables.

1.2 Optimising the Best Horizon Run

The subsequent stage, referred to as the "Best Horizon Run", aimed to determine the optimum model for specific combinations of horizon and target variables. The 'step-wise' model with parameters `pool = 3000` and `target_variable = 'cum_volume_500'` demonstrates superior performance in both the train and test environments. We focus on `horizon = 150`, which exhibits higher mean R^2 values of 0.203 in both the training and testing data, indicating better generalization and predictive accuracy. In comparison, other models yield lower mean R^2 values, particularly a negative value in the testing data, suggesting suboptimal performance and poorer generalization to unseen data.

The relatively small differences between R^2 and R^2_{adj} for both the overall data and the specific combination suggest that the 'step-wise' model is not overfitting due to the inclusion of too many predictors. This is a positive indication that the model is well-specified and likely to generalize well to new data. This support the selection of the 'step-wise' model for the specific combination of parameters, as the model appears to have a good balance between complexity (number of predictors) and performance (as measured by R^2_{adj}).

The subsequent process involved an extensive data analysis phase, incorporating correlation analysis and Variance Inflation Factor (VIF) calculations. Strategies to mitigate multicollinearity and enhance model performance were deployed, including the removal of highly correlated variables and data aggregation.

1.3 Addressing Multicollinearity and Refinement of Models

A key aspect of model refinement was the management of multicollinearity, a circumstance where high correlation is exhibited between two or more variables. This was tackled by discarding certain columns and aggregating feature intervals. To evaluate the impact of multicollinearity on the variance of the estimated regression coefficients, a correlation matrix and Variance Inflation Factor (VIF) analysis were performed pre and post-adjustment. High VIF values suggest high multicollinearity, calculated using the following formula:

$$VIF(k) = \frac{1}{1 - R_k^2} \tag{10}$$

where R_k^2 is the coefficient of determination of a predictor in a model that includes all other predictors.

Before Multicollinearity Reduction: VIF counts for thresholds: [1, 5, inf] 1: 0, 5: 21, inf: 22

After Multicollinearity Reduction: VIF counts for thresholds: [1, 5, inf] 1: 0, 5: 24, inf: 2

(2 (rate-count-iother, blother_1) variables with high VI)

By analysing feature correlations and VIF, multicollinearity was identified. Features demonstrating high multicollinearity were removed (cols_drop_correlated = ['blsame_2', 'blsame_3', 'blother_2', 'blother_3', 'vol_0_1', 'vol_0_2', 'binance-count-01', 'binance-count-12', 'binance-count-23', 'rate-USD-isame_01', 'rate-USD-isame_12', 'rate-USD-isame_23', 'rate-USD-iother_01', 'rate-USD-iother_12', 'rate-USD-iother_23')

Following the methodology outlined by [10], we further reduced multicollinearity by aggregating certain intervals (cols_aggregate_intervals_range = ['rate-count-isame_', 'rate-count-iother_', 'binance-btc-')

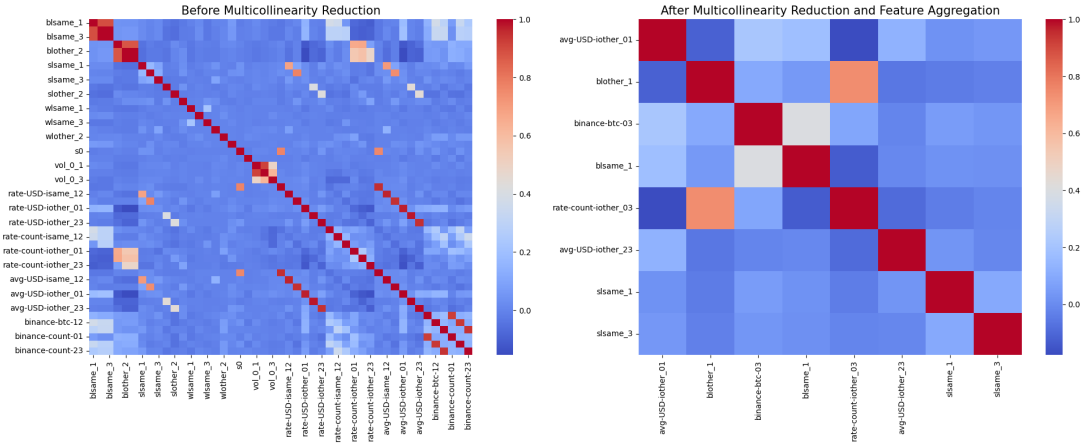

Figure 3: Correlation Matrix

1.4 Feature Selection via Step-wise Regression

The step-wise feature selection process is a robust method for fitting regression models. This iterative process involved adding or removing variables based on the t-statistics of their estimated coefficients. The least significant features were systematically eliminated and the model retrained, until all remaining features were deemed significant. This approach aids in preventing overfitting and improves model interpretability. Hello

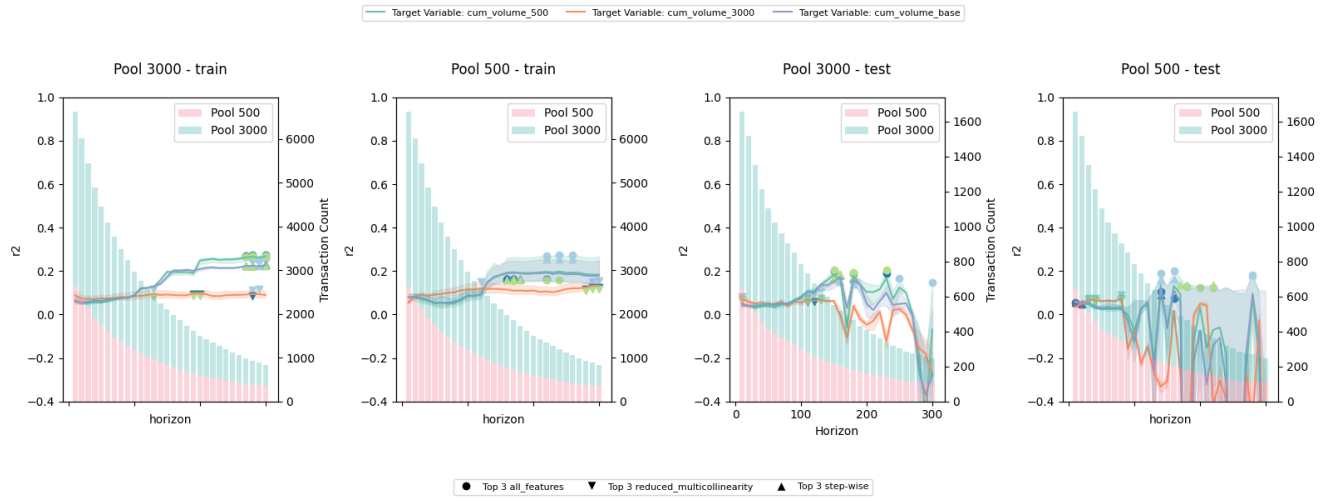

Figure 4: OLS Horizon Analysis

2 Analysis and Discussion

This section discusses the significance of our model refinement process, its findings, and provides an interpretation of the predictor coefficients. The effectiveness of the multicollinearity reduction process is examined first, followed by a residual analysis. A step-by-step description of each predictor is also provided.

2.1 Mitigation of Multicollinearity

Our process for reducing multicollinearity significantly improved the model. Initially, several variables with perfect multicollinearity (infinite VIF) were observed, along with a considerable number of variables in the high VIF category. This indicated a high degree of multicollinearity. However, following the reduction process, no variables exhibited high VIF, and most variables fell into the low VIF category, signaling the effective mitigation of multicollinearity in the data.

2.2 Residual Analysis

Following the mitigation of multicollinearity, we re-ran the models and conducted a residual analysis to inspect the distribution of residuals for any patterns or anomalies. This process is critical to validate the model's assumptions: a zero mean error term, constant variance, a normal error distribution, and no correlation between consecutive residuals (autocorrelation).

2.3 OLS Regression Model

Our OLS regression model indicates a significant overall relationship (F-statistic = 57.38, $p < 0.001$), with an R-squared value of 0.238. This suggests that the predictors collectively explain 23.8% of the variance in the dependent variable.

2.4 Predictor Coefficients

Predictor coefficients, as seen in Table 5, represent the mean change in the response for one unit of change in the predictor, while other predictors in the model are kept constant. For instance, for every unit increase in 'avg-USD-iother_01', the log of 'cum_volume_500' increases by 0.1750, assuming all other predictors are constant.

A Durbin-Watson statistic of 1.517, close to 2, indicates that there is no significant autocorrelation in the residuals. However, the Prob (Omnibus) and Prob (JB) probabilities being 0 suggests that the residuals are not normally distributed.

Predictor	Coef	Std Err	P-value	Significance
Constant	6.1377	0.026	0.000	True
avg-USD-iother_01	0.1750	0.013	0.000	True
blother_1	-0.1919	0.018	0.000	True
binance-btc-03	0.0976	0.013	0.000	True
blsame_1	-0.0612	0.014	0.000	True
rate-count-iother_03	0.1115	0.019	0.000	True
avg-USD-iother_23	0.0294	0.012	0.016	True
slsame_1	0.0181	0.012	0.135	False
slsame_3	0.0201	0.012	0.096	False

Table 5: Predictor Coefficients

2.5 Feature Importance

[TODO] Understanding the variables contributing significantly to the predictive power of our model can guide feature selection and model refinement. This can be accomplished through permutation importance or coefficient estimates, subject to the model and the specific context.

2.6 Residual Dispersion Analysis

Further analysis of residuals allows for understanding their dispersion. Residuals randomly dispersed around zero signify an appropriate model for the data. If not, it could be a sign that the model's assumptions have been violated.

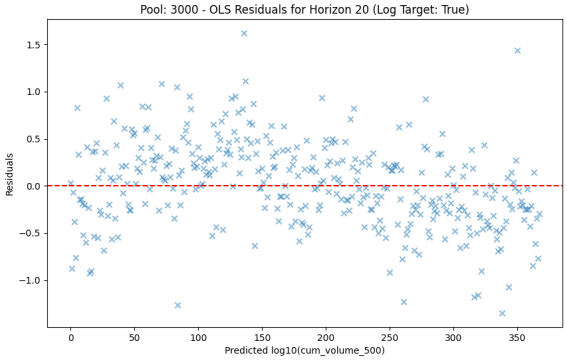

Figure 5: Residuals: One Horizon

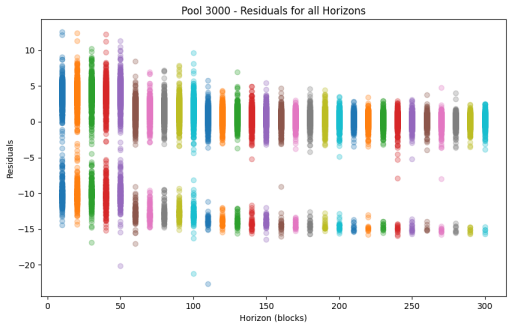

Figure 6: Residuals: All Horizons

2.7 Residual Analysis Insights

From the one horizon plot, residuals appear scattered around the center line of 0, without an obvious pattern, suggesting our model's assumptions (linearity, independence, and homoscedasticity) are likely met. However, a slight trend is observed: residuals are mostly positive for lower predicted values and mostly negative for higher predicted values, indicating potential underprediction and overprediction respectively.

The all horizons plot shows that residuals are distributed around zero across all horizons. However, a larger spread of residuals is noted for smaller horizons, suggesting less accurate model predictions. It also shows a trend where residuals for smaller horizons are skewed towards positive values, and residuals for larger horizons are skewed towards negative values.

2.8 Refinement Based on Residual Analysis

These insights from the residual analysis can guide further refinement of the models. For example, exploring different models or additional features for smaller horizons could improve prediction accuracy.

3 Performance Analysis and Discussion

Our models have evolved through three stages: 'all_features', 'multicollinearity', and 'stepwise'. Initially, all available features were used then multicollinearity was addressed, and finally, a stepwise regression was applied to select the most statistically significant features. Each stage aimed to improve the model's interpretability and predictive power.

We analyze the performance of our models across different aspects: horizon sizes, R-squared values, data pools, and target variables, and identify the high-performing categories.

3.1 Performance Across Horizons

As the horizon expands, the predictability of target variables (indicated by R^2 values) generally decreases. However, some combinations of run types and target variables maintain high R^2 values across larger horizons. For instance, 'all_features' and 'step-wise' run type with 'cum_volume_300' target variable in the pool 3000 data set consistently exhibits high R^2 values across various horizons.

3.2 Negative R-squared Values

Negative R^2 values, mostly in the test set, suggest certain models perform worse than a simplistic average-based model, indicating a failure to generalize to unseen data.

3.3 Comparing Pools

Comparing pool 500 and pool 3000, pool 3000 generally yields higher R^2 values. This suggests that reference mints in pool 3000 may lead to better predictions.

3.4 Comparison of Target Variables

For both pools, the 'cum_volume_500' target variable tends to yield higher R^2 values than 'cum_volume_3000', possibly suggesting a stronger correlation with the selected features.

3.5 High Performing Categories

Combinations with 'cum_volume_500' target variable frequently demonstrate good R^2 values, suggesting stronger predictability or a stronger linear relationship with the chosen features. The 'all_features' run type often appears among these high-performing combinations, indicating that including all features often yields better results. However, generalization to the testing data is a challenge, indicating a need for model refinement.

Going Forward

To summarize, the learnings from this project, particularly regarding data engineering and feature generation, are substantial. This largely unexplored field of financial technology presents vast opportunities for fresh findings and contributions to the existing body of knowledge. Our initial results are promising, and we're eager to delve into the forthcoming phases of our project, emphasizing model optimization and refining our analysis and conclusions.

The model could potentially be improved further by investigating the variables that still have mid/high VIF after the reduction. They might still have some level of correlation with each other, and additional feature engineering might be helpful. For instance, these variables could be further investigated to understand their correlations, and if they are conveying similar information, one of them could potentially be removed.

It would be valuable to investigate if there are any non-linear relationships between the predictors and the target variable. If such relationships exist, they might not be adequately captured by the linear model. In such a case, using a non-linear model might be beneficial.

Possible next steps could involve investigating the residual errors pattern further and possibly applying some transformation to the target variable or the features.

Our future steps in the modeling phase include:

- Dataset Splitting: Partitioning the dataset into training and test sets, performing model metrics analysis, and interpretation.
- Analysis Scope: Examining both Pool 500 and Pool 3000, with a focus on target variables related to volume on the same pool as the reference mint, the other pool, or both.
- Experimentation: Identifying the optimal combinations of features, horizons, and target variables for accurate predictions.

- Feature Selection: Using techniques like step-wise or Principal Component Analysis to reduce feature complexity, limit overfitting, and improve interpretability.

Our planned experiments include testing different lag lengths, incorporating quadratic features or interaction terms, and accounting for structural breaks, such as the Terra-Luna collapse (May 2022) and the Merge (Sep 2022).

We aspire here or on further work to drive a comprehensive analysis and quantitative discussion on:

1. The relationship between liquidity pool size and trading volume in BTC-ETH liquidity pools.
2. The influence of the liquidity pool size on the slippage in BTC-ETH trading pairs, considering CEX spillover effects.
3. Impact of BTC-ETH price volatility on trading volume relative to liquidity pool size, focusing on price divergence.
4. Specific periods/events that significantly affect the relationship between BTC-ETH liquidity pool size and trading volume.

While Ordinary Least Squares (OLS) regression is a good starting point, it might be worth exploring other types of models that could better capture the relationships in the data. For example, models that can handle non-linear relationships (e.g., decision trees, neural networks) or models that can account for time-dependence in the data (e.g., ARIMA, state space models) could potentially improve performance.

Future Extensions

Future extension:

Future extensions may include price divergences in BTC-ETH exchanges and pools, as arbitrage opportunities may also drive activity.

1. Direct pool features (2): total value locked (TVL)
2. Network spillover effects (8): trade flow imbalance.
3. Price divergences between the 500 and 3000 pools (2).

[A key concept to understand in this analysis is the notion of impermanent loss. The impermanent loss function for Uniswap v2 liquidity providers is a notable concept to be included in the discussion [5]. The role of yield farming protocols, their classification, and their sources of yield are also significant considerations in this analysis [12]]

Appendix

From the transaction data, we created 8,228 reference blocks that correspond to each mint operation. These blocks were then used to engineer features to calculate metrics for the same pool and other pools. These features capture data about liquidity pools and calculate metrics such as volatility, traded volume rate, trades count, and average volume. To expand the analysis, we have also included lagged features for the previous three mint operations. Table 6 serves as our block reference table.

pool	reference_blockNumber	same_blockNumberChain	other_blockNumberChain
500	14498564	[14498564, nan, nan, nan]	[14498564, nan, nan, nan]
500	14498699	[14498699, 14498564, nan, nan]	[14498699, nan, nan, nan]
500	14499597	[14499597, 14498699, 14498564, nan]	[14499597, 14499560, 14499457, 14499198]
500	14499836	[14499836, 14499597, 14498699, 14498564]	[14499836, 14499560, 14499457, 14499198]
500	14500355	[14500355, 14499836, 14499597, 14498699]	[14500355, 14500043, 14499560, 14499457]
...	...	...	...
3000	15648981	[15648981, 15648330, 15648305, 15648187]	[15648981, 15648887, 15648536, 15646933]
3000	15649246	[15649246, 15648981, 15648330, 15648305]	[15649246, 15649243, 15648887, 15648536]
3000	15649545	[15649545, 15649246, 15648981, 15648330]	[15649545, 15649522, 15649347, 15649269]
3000	15649565	[15649565, 15649545, 15649246, 15648981]	[15649565, 15649522, 15649347, 15649269]
3000	15649578	[15649578, 15649565, 15649545, 15649246]	[15649578, 15649522, 15649347, 15649269]

Table 6: Block Interval Chains

Finally, we've constructed a base table 7 with the `start_blockNumber` of each horizon and the `reference_blockNumber`, defined by mint operations in the block. This base table is joined with the mint aggregated transaction data to generate our target variables and independent variables that are lagged.

The effects of BTC-ETH price volatility on trading volume compared to liquidity pool size: The binance-btc-03 predictor in your OLS model, which you've stated represents spillover effects from Binance, could potentially

horizon_blockNumber	min_flag	reference_blockNumber	horizon_label	cum_volume_500
108757	0	15552674	9	423,485.34
108758	0	15552674	10	423,485.34
108759	1	15552772	1	328,338.73
108760	0	15552772	2	406,084.78
108761	0	15552772	3	536,640.71
...	...	...	...	...
109047	0	15555464	12	122,730.73
109048	0	15555464	13	123,650.59

Table 7: Example: Horizon Table (reference mint on pool=3000)

encapsulate aspects of price volatility. Its positive coefficient suggests that as these spillover effects (which may include price volatility) increase, the dependent variable also increases.

Time-Series Specific Analysis: If your data has a temporal component, consider analyzing autocorrelation and partial autocorrelation plots of the residuals. This will help you check for any leftover temporal structure in your errors which, if present, suggests there may be room to improve your model.

References

[1] binance-public-data. https://github.com/binance/binance-public-data/blob/master/python/README.md.

[2] defillama. https://defillama.com/.

Note: DefiLlama is a popular platform that provides real-time data and analytics for decentralized finance (DeFi) protocols and liquidity pools.

[3] etherscanapi. https://api.etherscan.io/api.

[4] Uniswapapi. https://api.thegraph.com/subgraphs/name/uniswap/uniswap-v3.

[5] Andreas A Aigner and Gurvinder Dhaliwal. Uniswap: Impermanent loss and risk profile of a liquidity provider, 2021. https://github.com/atiselsts/uniswap-v3-liquidity-math/blob/.

[6] Atis Elsts. Liquidity math in uniswap v3 technical note, 2021. https://github.com/atiselsts/uniswap-v3-liquidity-math/blob/.

[7] Lioba Heimbach, Eric Schertenleib, and Roger Wattenhofer. Risks and returns of uniswap v3 liquidity providers. 5 2022. http://arxiv.org/abs/2205.08904http://dx.doi.org/10.1145/3558535.3559772.

[8] Igor Makarov. Cryptocurrencies and decentralized finance (defi), 2022.

[9] Deborah Miori and Mihai Cucuringu. Defi: data-driven characterisation of uniswap v3 ecosystem an ideal crypto law for liquidity pools. 12 2022. http://arxiv.org/abs/2301.13009.

[10] Deborah Miori and Mihai Cucuringu. Defi: modeling and forecasting trading volume on uniswap v3 liquidity pools, 2023. https://ssrn.com/abstract=4445351

Note: This project is inspired by the above paper, from which the main methodology is borrowed, replicated, and expanded.

[11] Matias Vizcaino. Team111repo. https://github.gatech.edu/MGT-6203-Summer-2023-Canvas/Team-111.

Note: The GitHub repository associated with this project contains the code and additional details.

[12] Jiahua Xu and Yebo Feng. Reap the harvest on blockchain: A survey of yield farming protocols. *IEEE Transactions on Network and Service Management*, 20:858–869, 3 2023. http://dx.doi.org/10.1109/TNSM.2022.3222815.