Final Report

Group 1: Yuan Wang, Yanmin Chen, Tianye Yin, Alexander Tan, Jessie Xue

Introduction

Ethereum's role as a foundational blockchain technology is increasingly critical in the rising field of Web3 gaming. The mechanism of gas, which powers every transaction and smart contract on the Ethereum network, requires users to pay a fee, priced in gwei. This fee fluctuates based on network demand and transaction complexity. For game developers, who aim to integrate Ethereum transactions into their gaming platforms, managing these costs effectively is essential to maintain gameplay fluidity and user engagement

Predicting Ethereum gas prices is notoriously challenging due to their volatile nature, influenced by factors like network congestion and transaction volume. For game developers in the Web3 space, unpredictability in gas prices complicates budgeting and resource allocation, potentially leading to inflated operational costs and degraded user experiences. Our project seeks to develop a predictive model specifically tailored to anticipate gas usage, and thereby the base fee, for future blocks. This model will empower game developers to set optimal transaction fees (Max Fee and Max Priority Fee) to achieve a balance between cost-efficiency and transaction throughput. Ultimately, we aim to optimize transaction costs and enhance the scalability of their gaming applications on the Ethereum network.

The project will focus on the application of advanced statistical and machine learning techniques to model Ethereum gas usage. We will leverage both on-chain and off-chain data, ranging from historical Ethereum transaction data to prices of other cryptocurrencies, to identify predictors of gas costs and build our predictive model.

Commented [1]: need to add on

The report is structured to first detail the data pre-processing and preliminary exploration steps, including data gathering, cleaning, and baseline modeling findings. This is followed by a discussion of more advanced models, including those using transformers. We then conclude with a summary of our project's findings, recommendations, and limitations.

- Data Sources/Collection/online/offline data
- EDA - from presentation #1
- Feature Engineering
- Baseline Model (&Data Processing) - from presentation #2
- Advanced Models (&Data Processing)
- Discussion of Results & Business Implications
- Summary

Data Sources/Collection: Online and Offline Data

On-Chain Data Collection

Our project primarily utilizes on-chain data extracted from BigQuery's public datasets, specifically focusing on Ethereum's blockchain data. We access two main datasets:

Crypto_ethereum.blocks: This dataset includes comprehensive details about individual blocks on the Ethereum blockchain. Key fields used in our analysis include:

- timestamp - Time at which the block was mined.
- gas_used - Total gas used by all transactions in the block.
- number - The block number, indicating the block's position in the blockchain.
- gas_limit - The maximum amount of gas that could be used in the block.
- base_fee_per_gas - The minimum per-gas fee for inclusion in the block post-London upgrade.
- transaction_count - Number of transactions in the block.
- size - The size of the block in bytes.
- Additional fields like miner, difficulty, and total_difficulty provide insights into the mining process and the network's status at the time of block creation.

Crypto_ethereum.transactions: Contains transaction-level data within the Ethereum network. Important fields include:

- hash - Unique identifier of the transaction.
- value - Value of Ether transferred in the transaction.
- gas_price - Price per unit of gas specified in the transaction.
- receipt_gas_used - Amount of gas used by the transaction after execution.

Data retrieval through Kaggle, which offers an efficient interface for querying and downloading these large datasets.

Off-Chain Data Collection

To augment the predictive capabilities of our models and enrich our analyses, we integrate a multitude of off-chain data sources:

- Etherscan: Provides additional Ethereum network metrics and token information.

- Financial Platforms: We leverage data from Yahoo Finance and Bloomberg Crypto to include broader financial indicators such as:
    - Bitcoin prices: Helps in understanding correlations and potential spillovers between major cryptocurrencies.
    - Stock indices and prices: Includes S&P 500, Nvidia, and ASML. These are used to gauge the influence of broader market trends on the cryptocurrency market.
    - Relative Strength Index (RSI): A technical indicator used to analyze market conditions.
    - Crypto Volume from Solana: Comparative analysis with other significant blockchains, like Solana, provides insights into the relative performance and investor sentiment across different networks.

This combination of on-chain and off-chain data allows us to conduct more possible analysis, providing an internal and external comprehensive view for analyzing ethereum features in blockchain.

Exploratory Data Analysis (EDA)

During the preliminary exploratory data analysis, we investigate the impact of on-chain and off-chain data on the dynamics of Ethereum's base fee and gas used, created from transaction-level dataset,with a sanity check to ensure its reliability before diving into deeper analysis.

Key Findings:

Autocorrelation of Gas Usage: We examined the autocorrelation of gas usage by aggregating data on an hourly basis. Our findings showed a significant correlation within the 24 hour period, this finding suggesting that past gas usage can influence future usage patterns. This pattern could be important for forecasting models at predicting short-term fluctuations in gas demand.

Impact of External Events: During the Terra-Luna crash, contrary to expectations of a dramatic decrease, gas usage exhibited a gradual decline. This highlights the resilience of the network during market turmoils. Another case of Ronin Network Crypto Heist did not show significant impact on daily gas usage, despite the substantial financial implications. This indicates that such security breaches might not directly translate to changes in ethereum network activity levels.

Transactional Analysis: Our transaction-level analysis spotlighted how network load and user behavior patterns could impact Ethereum's total transactions and price. Notable findings included predictable spikes in gas fees associated with algorithmic trading operations, NFT drops, and updates in decentralized applications.

EIP-1559 Impact: Our post Implementation analysis for EIP-1559 showed it smoothed out fee market volatility to some extent, this is demonstrated by our control chart of the ratio of average base cost to average priority fee. However, this trend was skewed during periods of high network congestion, with users paying higher priority fees, understanding this trend may require empirical explanation.

Off-Chain Influences: Our analysis extended to off-chain factors where we explored correlations between Ethereum's base fee and external financial indices like the Nasdaq100 and cryptocurrency prices from Gemini exchange. The results showed moderate correlations, indicating that while these factors have some relationship with Ethereum's gas usage and fees, they are not the sole drivers.

Cross-sectional and Lagged Correlations: We also explored the lagged effects of gas usage and transaction counts on current levels by constructing a new correlation matrix. Interestingly, we observed a negative correlation between the average units of gas used in previous hours and those used subsequently, suggesting potential for a predictive model at an hourly resolution.

Methodological Insights and Challenges:

Our exploratory analysis utilized a variety of statistical tests and visualizations to draw meaningful insights from the data. We employed two-sided Z-tests, correlation matrices, and control charts among other techniques to rigorously test our hypotheses and uncover patterns within the data. These preliminary insights ultimately informed our next steps in feature engineering as well as baseline and further model building.

Also, several anomalies and outliers identified during the analysis warrant further investigation. Additionally, integrating more granular data and expanding the scope of off-chain data sources could enhance the predictive accuracy and depth of our future analyses.

Feature Engineering

Next, we conducted featuring engineering to help capture additional patterns or insights the original columns might miss. We created features based on 3 buckets for more than 40 new variables to discover new insights concerning time, interactions, or other off-chain data.

Trend Features: Our first set of features were created to capture the various effects of time over minute, hourly, daily, and weekly intervals. This involved creating lag variables to find trends in gas usage over time. Rolling window calculation variables for mean and standard deviation to remove short term fluctuations and capture trends and patterns over selected time intervals.

Expanding mean features provide cumulative gas used calculations up to the current time point to capture the overall trend. Difference features to find the difference between the current and previous observations to capture the rate of change. Features that measure percentage change from the previous rows and exponentially weighted percentage change from the previous rows were also included. In addition, seasonal decomposition features using statsmodel package to find seasonal trends. Sine and cosine function for hour of day and day of week to analyze the frequencies to determine any cyclical trends.

Off-Chain Features: External features we used to enrich our analysis include Bitcoin and Solano price as a proxy for the crypto market for similar blockchain assets compared to Ethereum. S&P 500 price to find any relationship between Ethereum and traditional financial markets. Nvidia and ASML as a proxy for Bitcoin miners and the technology sector. In the past, the demand for semiconductors has had a strong positive correlation with increases in crypto currency mining activity. Recently however, the demand for semiconductors have also skyrocketed due to artificial intelligence which could be noise for this feature.

Interaction Variables: Interactions between features were transformed to see if one feature effect on the target variable depends on another feature. Some examples are transaction count and gas used per block, base limit and gas fee, day of week and base fee, and gas limit and transaction. These example interaction variables can hopefully capture how the number of transactions in a block collectively impacts the total gas use, how the cost of gas and the maximum gas allowed in a block together affect blockchain dynamic, time of day could influence the value and volume of transaction, and network's capacity limit (gas limit) interacts with transaction demand in influencing gas usage or fee.

Technical Analysis Indicators: With the assumption that the fluctuation in Ethereum price can lead to investment and thus transactions, which can affect gas price, technical analysis indicators like Relative Strength Index (RSI), which measures the speed and magnitude of the recent price changes and Moving (MACD), the difference between 12-period exponential moving average and 26-period exponential moving average, were also included.

Sentimental Factors: It is reasonable to assume that when users have a positive attitude towards Ethereum, they tend to trust them and make transactions on the platform, and vice versa. Thus, the Total Value Locked (TVL) on-chain of Ethereum and its market competitor Solana were included, which can reflect investors’ confidence in these products.

When applying these features to the base model, we noticed the model performed exceptionally well which was unrealistic. Data leakage was the likely cause for overfitting and in feature engineering poses significant risks to predictive modeling. The suspect variables were interaction variables involving the dependent variable and rolling window functions. When features are created using future information, either directly through the dependent variable or indirectly via

correlated predictors, it leads to models that overfit training data and fail to generalize to unseen data. Misaligned rolling windows might inadvertently include future data points, resulting in misleadingly high performance metrics during model validation. To mitigate these risks, we removed these variables from further analysis.

Baseline Models

Three baseline ARIMA models were developed to assess the effectiveness of different data granularities and to explore the impact of seasonality. The models were constructed using a sequential 80/20 train-test split.

The first model used day-level gas usage data, modeled non-seasonally with an ARIMA(0,1,0) determined by the auto_arima function from the pmdarima package. This model yielded an RMSE of 26,266.13, corresponding to 0.17% of the average daily usage, and a very low MAPE of 0.13%. The negative R-squared value of -0.01 suggests that the model does not capture underlying patterns in the data effectively.

The second non-seasonal model addressed hour-level data, identifying an ARIMA(1,1,1) as the best fit. This approach resulted in an RMSE of 118,434.46, or 0.78% of the average usage, with a MAPE of 0.55%. The increase in error metrics compared to the daily model illustrate the complexities and potential volatility inherent in hourly data, which could be influenced by numerous external factors not accounted for in models using only time-series data.

For the last ARIMA model, we incorporated weekly seasonality into the day-level data with an ARIMA(5,1,5)(0,0,2)[7] model. This model had an RMSE slightly higher than the non-seasonal daily model at 26,738.83, which is 0.18% of the mean, but maintained a consistent MAPE of 0.13%. Given that the predictive accuracy did not improve, the introduction of weekly seasonality could point to the fact that weekly fluctuations do not play a significant role in the time series. Thus, these ARIMA models signal the need to explore additional variables and more advanced modeling techniques to better capture gas use dynamics.

Next, we modified the target variable, gas used per block, through a temporal shift and allocated the dataset into training and testing sets using an 80/20 split to ensure robust evaluation. The models tested, after tuning their parameters, yielded the following R-squared values: 0.000361 for linear regression, 0.031 for XGBoost, and 0.0257 for CatBoost. However, all models have high Mean Squared Errors (MSE), indicating significant average discrepancies between the predicted and actual values. This suggests that further model refinement or alternative strategies may be necessary to improve predictive accuracy.

We aggregated the dataset to a daily granularity and calculated rolling averages and standard deviations for windows of 7, 15, 30, and 60 days for gas used per block. We also created

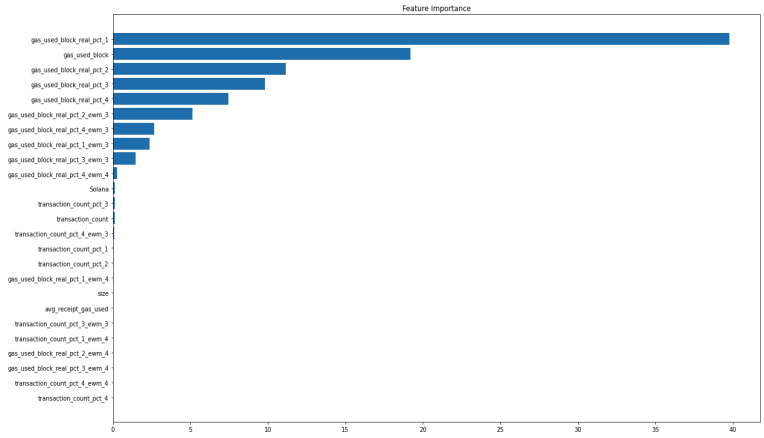
interaction features to capture potential interdependencies among variables. After standardizing and scaling these features, we conducted feature selection to finalize the most relevant predictors. We then split the dataset using an 80/20 train-test ratio. After conducting modeling, the R-squared values for the XGBoost and CatBoost rolling predictions were -0.2116 and -0.2488, respectively, while the simple linear model showed an R-squared of -2.8792. These negative values suggest a significant lack of fit across the models, although the nonlinear models (XGBoost and CatBoost) performed relatively better compared to the linear model. It appears that our feature engineering might have inadvertently introduced some noise into the models, indicating potential areas for refinement and optimization.

Advanced Models

Catboost:

In addition to the models mentioned above, we tried one of the boosting algorithms, catboost, and got a decent result. Due to the limitation of our data collection strategy, there were no records for volume, the closing price for ETH-USD, RSI, and MACD before April 8th 2022 13:00:00, and thus data before that time were dropped and the end datetime was set to be January 31st 2024 23:59:59. The target variable was set to be the total gas used in the next block.

We chose a total number of of 41 representative features from each kind in the feature pool and computed the feature importance using a default catboost regression model. The result plot is attached below.



Based on the feature importance score, top 25 features were chosen to build the model. We tried to tune the hyperparameters and thus conducted sliding window validation. 5 windows were used

to test 28 different combinations of the number of iterations and the depth. Each window size was 12 months, where the first 9 months were used as the training set, the following 2 months as the testing set and the last month as the out of time set for further validation. In addition, within each window, we used the training set's statistics to preprocess the data, including winsorization, normalization, scaling so that the numbers are in the range from -1 to 1, and impute the missing values with the training set's mean.

cbr_iter	d	trn_rmse	tst_rmse	oot_rmse	trn_r2
100	3	3,381,275.05	2,889,230.15	2,919,744.88	0.6518
100	4	3,184,024.34	2,827,788.87	2,847,468.20	0.6911
100	5	3,031,182.39	2,850,125.35	2,864,531.92	0.7200
100	6	2,922,256.69	2,881,156.56	2,897,277.43	0.7399
150	3	2,796,020.95	2,607,712.33	2,618,529.28	0.7617
150	4	2,562,950.74	2,822,451.53	2,824,373.48	0.7998
150	5	2,383,910.40	3,080,988.70	3,081,321.82	0.8268
150	6	2,242,199.09	3,382,149.27	3,383,836.59	0.8469
200	3	2,396,019.62	2,716,575.15	2,714,105.91	0.8250
200	4	2,154,137.47	3,232,248.01	3,222,399.71	0.8585
200	5	1,953,617.43	3,703,876.89	3,694,955.96	0.8836
200	6	1,810,320.73	4,124,197.06	4,118,224.83	0.9001
250	3	2,116,587.32	3,001,929.74	2,989,152.37	0.8634
250	4	1,881,971.23	3,776,186.10	3,759,384.77	0.8919
250	5	1,668,977.89	4,395,656.54	4,383,053.42	0.9151
250	6	1,525,379.61	4,861,131.92	4,851,335.89	0.9291
300	3	1,931,872.24	3,321,702.91	3,301,762.84	0.8862
300	4	1,697,446.22	4,300,425.94	4,279,275.77	0.9121
300	5	1,493,400.52	4,936,963.94	4,922,124.82	0.9320
300	6	1,352,614.27	5,395,062.32	5,384,702.06	0.9443
350	3	1,797,829.17	3,699,726.67	3,675,652.36	0.9015
350	4	1,569,828.71	4,754,720.38	4,731,650.54	0.9248
350	5	1,378,661.14	5,337,872.09	5,322,850.93	0.9421
350	6	1,246,232.78	5,773,784.26	5,764,296.16	0.9527
400	3	1,690,587.72	4,094,464.96	4,067,477.44	0.9128
400	4	1,470,797.31	5,172,023.00	5,148,233.61	0.9340
400	5	1,298,278.59	5,653,475.64	5,639,964.03	0.9486
400	6	1,174,056.17	6,049,930.13	6,042,599.75	0.9580

The optimal combination for the hyperparameter in our case was the number of iterations equal to 150 and depth equal to 3. In the 5 window validation, the average RMSE for the training set, the testing set and the out of time set were 2,796,020.95, 2,607,712.33 and 2,618,529.28 respectively. The training set average R^2 was 0.76.

Multivariate RNN:

We aggregated our dataset to a daily granularity, employing summarization techniques such as taking averages or sums of related variables as appropriate. We also applied a log transformation to normalize the distribution of the variables, enhancing the model's ability to capture exponential trends and reduce skewness and scaled the variables to a [0,1] range to standardize the input features. We set the window size to 30 days, which allows the LSTM model to consider the past month's data to make predictions and capture the monthly cyclicality in the data.

After splitting, we applied (LSTM) network for sequential data prediction to forecast the target variables, which achieved a test MAE of 0.038, and it should be transformed back.

Transformer Model: For our transformer model implementation, we conducted another feature engineering specifically focusing on gas usage and transaction count-related features. This augmentation aimed to capture nuanced relationships within the Ethereum gas usage data, potentially improving the model's predictive capabilities. Despite these efforts, the transformer model exhibited an RMSE of 57032228, this indicating suboptimal performance compared to RNNs. To address this, further emphasis will be placed on refining preprocessing techniques, optimizing model architecture, and fine-tuning hyperparameters. To improve the transformer model requires additional feature engineering, particularly in enhancing gas usage and transaction count-related features, prioritizing the tuning and better features facilitating the transformer model's accuracy in predicting Ethereum gas used by future blocks.

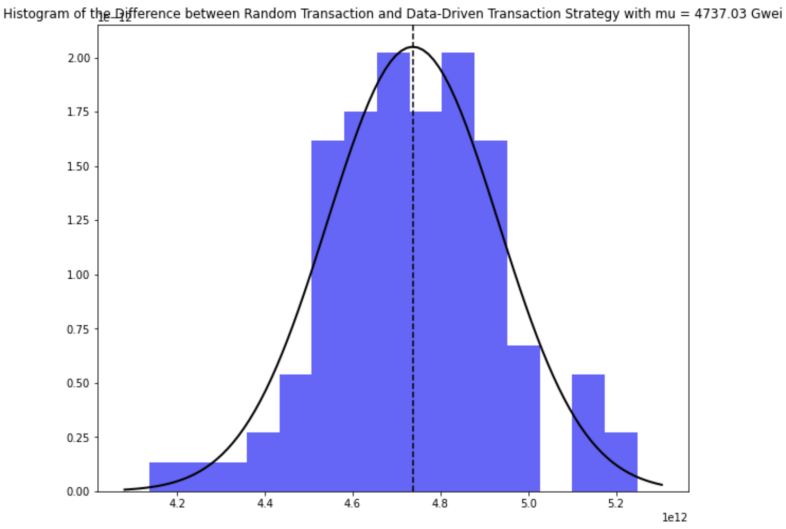
Business Application

Although the block-level model did a decent job in predicting the total gas used in the next block, the output alone might be useless. It is hard to make transaction decisions solely based on the total gas used and the corresponding base fee for gas price in the next block. And thus we decide to aggregate our dataset to minute level by taking the sum, average and total counts of the variables, take the average base fee per gas as the target variable, and introduce a new model based on the less granular data. Because of the setting, the output of this model is the expected average base fee per gas for the next minute. In this case, when it is required to make a transaction in the next minute, the outputs from the block-level model and minute-level model can serve as a good reference: if the base fee per gas in the next block is predicted to be lower than the predicted average base fee per gas in the next minute, then a transaction in the next block should be made because it can save money. Otherwise, the transaction can wait till the base fee per gas for the next block is predicted to be lower than the average.

To test if this strategy is feasible, we ran a simulation on the data in January 2024. The block level model used here was the model mentioned in the previous section, with the number of iteration equal to 150 and depth equal to 3, and it was trained on all the data in 2023. In this example, the minute level model was also a catboost regression model. After preprocessing, sliding window analysis and hyperparameter tuning, the hyperparameters were set to be 350 for the number of iterations and 5 for the depth, and the model can achieve an average RMSE of 16,234,489,543.23, 16,447,767,842.63, and 17,216,479,138.38 for the training set, testing set and out of time test respectively. The training set average R^2 was 0.53. Then the model was retrained on all the data in 2023.

cbr_iter	d	trn_rmse	tst_rmse	oot_rmse	trn_r2
250	4	17,838,893,359.46	16,304,557,336.93	17,080,814,867.79	0.4287
250	5	17,032,710,205.21	16,157,441,003.93	16,981,014,328.67	0.4792
250	6	16,225,546,310.68	16,252,530,881.21	17,045,367,675.58	0.5274
300	4	17,453,923,817.26	16,443,403,837.94	17,184,944,928.53	0.4531
300	5	16,604,251,854.52	16,316,837,215.26	17,125,010,550.85	0.5050
300	6	15,773,855,136.42	16,433,657,033.01	17,189,184,221.28	0.5533
350	4	17,115,364,067.23	16,604,473,043.15	17,285,670,886.31	0.4741
350	5	16,234,489,543.23	16,447,767,842.63	17,216,479,138.38	0.5268
350	6	15,364,178,899.98	16,625,376,763.65	17,352,406,644.80	0.5762
400	4	16,817,774,546.48	16,775,768,813.05	17,401,162,713.45	0.4923
400	5	15,923,215,514.65	16,610,056,855.09	17,321,473,986.14	0.5448
400	6	15,012,301,908.34	16,758,184,081.85	17,426,865,953.78	0.5954
450	4	16,553,483,555.85	16,905,323,518.95	17,505,793,906.22	0.5081
450	5	15,594,231,387.02	16,740,192,472.48	17,398,887,623.14	0.5634
450	6	14,688,326,854.26	16,882,733,124.01	17,549,967,162.46	0.6127
500	4	16,275,389,399.80	17,018,342,241.95	17,556,165,599.42	0.5246
500	5	15,276,224,509.01	16,852,229,551.13	17,458,082,016.95	0.5811
500	6	14,352,976,770.87	16,954,169,057.64	17,608,754,207.42	0.6302
550	4	16,002,440,513.51	17,077,060,250.12	17,598,049,006.75	0.5404
550	5	14,980,872,028.88	16,876,357,191.87	17,476,118,721.89	0.5972
550	6	14,058,705,777.96	16,971,734,428.50	17,637,719,650.91	0.6452
600	4	15,752,019,558.34	17,154,851,903.92	17,664,749,269.82	0.5547
600	5	14,702,431,526.63	16,875,159,361.37	17,508,871,943.64	0.6121
600	6	13,783,231,466.32	16,991,585,449.21	17,666,399,432.17	0.6590
650	4	15,514,925,256.72	17,172,099,786.56	17,691,885,453.78	0.5680
650	5	14,440,819,031.96	16,880,621,363.30	17,516,705,198.02	0.6257
650	6	13,533,380,510.84	16,984,342,781.75	17,657,812,610.47	0.6713

To run the simulation and see the difference, we assume there is a need to make a transaction that costs exactly one unit of gas every minute, and a default transaction strategy would be to make the transaction randomly within each minute. An alternative transaction strategy to meet the same need would be to within each minute, make the transaction as soon as the predicted base fee per gas in the next block is lower than the predicted average predicted base fee per gas in the next minute. If the output of the block level model is always greater than the output of the minute level model within the minute, accept the last record of base fee per gas within that minute and make the transaction. With 100 rounds of simulation in this setting, the strategy that uses models based on data with different granularity level is expected to save around 4737.03 Gwei of gas, and as of 08:40:00 May 6, 2024, the exchange rate is 1 GWEI equals $0.058336037997, which means that our strategy could save around $276.3 per month.



Similar transaction strategy can stem from this prototype. With models trained on data with different granularity levels, the strategy can serve as a reference to decide when to make the transaction if it has to happen within the next couple minutes or hours or even days.

Next steps: More Features, Better Models

Reference

Ethereum Blockchain Public Dataset - Google BigQuery, bigquery.cloud.google.com/dataset/bigquery-public-data:ethereum_blockchain. Accessed 07 May 2024.

Nvidia Corporation (NVDA) Stock Historical Prices & Data - Yahoo Finance, finance.yahoo.com/quote/NVDA/history/. Accessed 07 May 2024.

S&P 500 Index (^SPX) Historical Data - Yahoo Finance, finance.yahoo.com/quote/%5ESPX/history. Accessed 07 May 2024.