University of Hertfordshire

Department of Physics, Astronomy and Mathematics

MSc Data Science

Project report: 7PAM2002 – Data Science Project

Investigating Different Approaches to Forecast Ethereum Gas Fees

Iniaki A. Boudiaf

Supervisor: Dr Ashley Spindler

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Data Science at the University of Hertfordshire (UH).

It is my work except where indicated in the report.

Abstract

Forecasting has always been a critical tool within businesses, as Web3 gains more traction with an increasing number of Decentralised Autonomous Organisations (DAOs) where many run on the Ethereum Virtual Machine (EVM) forecasting Ethereum gas fees to predict usage costs to deploy the service or product they provide. To accomplish this, several models ranging from statistical to deep learning were assessed for their ability to forecast Ethereum gas fees. Based solely on metrics the Theta model performed best, but N-BEATS was the best at forecasting on the validation dataset. LightGBM was able to achieve great forecasts within the first month. However, as the forecast extends further out, the prediction diminishes, meaning LightGBM is suitable for short-term forecasts.

In contrast, N-BEATS was able to forecast the entire validation dataset (roughly 8 months). Generally, most of the models tested could capture either the trend or volatility of the dataset, but many struggled to capture both. Overall, given the volatility of the Ethereum gas price, many models would struggle to forecast an accurate gas price successfully.

Content

1 Introduction

1.1 Purpose of the report

Ethereum is the second largest cryptocurrency by market capitalisation (CoinMarketCap, accessed 18 July 2022) and has an average cumulative wallet growth rate of 148.8% (YCharts, accessed 18 July 2022). This drastic growth in the Ethereum network demonstrates the emergent age of Web3, the third generation of the evolution of web technologies. As Web3 is based on blockchain technologies such as the Ethereum network, use of these networks will increase alongside it. This can cause a massive paradigm shift in many sectors such as finance, governance, education, and other industries that provide a service that can be digitally distributed. For example, the financial industry has already launched decentralised finance (DeFi) platforms where users can trade, borrow, and lend. However, if this shift were to happen on the Ethereum network, a large problem would arise in which many transaction senders today are already facing; the task of choosing an optimal gas price.

As it currently stands, users sending a transaction on the Ethereum network will define the maximum amount of gas they are willing to spend, typically denoted in gwei, which is one-millionth of an Ether. However, the suggested amount of gas on many popular hot wallets such at MetaMask or Coinbase Wallet, is a recommended gas limit with a low, medium, and high transaction speed. The ability to predict the gas fee would help users optimise the cost of a transaction by providing an estimated gas fee, allowing them to identify when the most suitable time is for them to send a transaction.

1.2 Research methods

In this report, I assessed the performance of several models ranging from statistical to deep learning. The statistical models covered the basics such as Exponential Smoothing and Theta. The machine learning models used were Random Forest and Light Gradient Boosted Model (LightGBM). In addition to five more models, which all use Deep Learning; N-BEATS, DeepAR, LSTM, GRU, and TFT.

1.3 Findings and Limitations

Although the Theta model had the best overall weighted average, the N-BEATS deterministic and probabilistic models performed best on the validation dataset. However, the best performing model throughout all given timeframes during the majority of the back testing was the LightGBM model. However, what was visible with all models was that as the duration of the forecast increased, so did the symmetric mean absolute percentage error (sMAPE), display the limitations of forecasting too far out into the future. Furthermore, given that the Ethereum network is still a relatively new technology and the potential for such a powerful concept is still being explored and updated regularly, leading to significant volatility in the gas price, increasing the challenge of forecasting gas fees.

2 Background

The concept of decentralised currency has been well established for many years. However, many were vulnerable to Sybil attacks, where a single attacker was able to create thousands of fake nodes to obtain a majority share on the decentralized network unilaterally. This issue was solved by Nakamoto, 2009, the developer of the Bitcoin network. By merging a simple decentralised consensus protocol based on nodes which place the network's transactions into a “block” every ten minutes creating a blockchain. To solve the

issue of bad actors spoofing nodes, a proof-of-work mechanism was put in place where you would need a large, currently inconceivable, amount of computing power to attack the network. This became the foundation of many crypto-currencies available today (Buterin 2014).

Currently, the Ethereum network works on a Proof-of-Stake (POS) algorithm where transactions on the blockchain are imposed with a fee known as 'gas' typically denoted in Wei, which is $1\ Ether \times 10^{-18}$, This is how the cost of using EVM is measured.

The cost of different actions on the network, such as the creation and execution of smart contracts, have a globally agreed price determined by the market according to supply and demand. In addition, each transaction also includes a gas limit which refers to the maximum amount of ether a user is willing to spend on the particular transaction. Although, if the gas limit is too low, the transaction is determined invalid due to 'gas failure' and the gas which has already been spent on mining (validation) is lost. However, if the sender sets a gas limit higher than the gas left over from the cost of the transaction is refunded to the sender. Miners are free to pick the transactions they verify, meaning the higher transactions provide greater value and will be mined first, leading to an overall increased gas fee for all users on the network. (Buterin 2014; Mars et al. 2021)

3 Related Work

3.1 Ethereum Related Works

Although there has been research into the mechanisms of Ethereum and its price, the interest for Ethereum gas fees only recently started increasing within the last 2 years (Google Trends, accessed 19 July 2022). While research into Ethereum gas fees is relatively new, there are still several papers with various approaches. Antonio Pierro et al. 2019 discusses and assesses various factors that may influence Ethereum transaction fees. In the article they use 3 methods to determine the significance of multiple features in relation to Ethereum gas fees. The 3 methods used were pair-wise granger causality, Augmented Dickey-Fuller (ADF) test and Pearson correlation.

There has also been research into gas price oracles, which provide gas price recommendations for Ethereum clients. As of today, the largest Ethereum oracle is 'Geth', accounting for over 75% of Ethereum clients. Geth recommends a gas price by observing the 100 blocks preceding the current one and uses the 60th percentile of the minimum gas prices as a recommendation. While this approach works, it does have its limitations such as underestimating the gas fee when there is a surge of pending transactions (Chuang and Lee 2022).

Another approach was introduced by Werner et al. 2020 in which they proposed a deep learning-based Gated Recurrent Unit (GRU) to predict the suggested gas price in conjunction with a novel algorithm to recommend a gas price for the user to submit, resulting in an average of 50% in cost savings for Ethereum gas with minimal delay for the transaction.
Mars et al. 2021 proposed a machine learning approach for gas price prediction in which they assessed 3 machine learning models; Facebook Prophet, Gated Recurrent Unit (GRU) and Long-Short Term Memory (LSTM). Their results showed that the LSTM and GRU both outperformed the Facebook Prophet and Geth

oracle with a mean squared error (MSE) value of 0.008, whereas Geth oracle and Facebook Prophet provided an MSE of 0.016 and 0.014, respectively.

Another paper (Ferenczi and Bâdicâ 2022) also looks at using Recurrent Neural Networks to predict Ethereum gas fees. However, they use Amazon Sage Maker's Deep Auto-Regressive (Deep AR) model using a various number of features assessing the results at 0,3 and 5 features – this returned an MSE of 2.9 e plus 20, which means the results were not normalised, which hinders comparison to previous work. However, they did include Normalised Regression Mean Squared Error (NRMSE) of 0.194, both of these results are for the 5-feature model which performed best.

3.2 Makridakis Competitions

The Makridakis-Competitions are commonly referred to as 'M-competitions'.
M-competitions "are a series of open competitions to evaluate and compare the accuracy of different time series forecasting methods. They are organised by teams led by forecasting researcher Spyros Makridakis" (M-Competitions, Wikipedia, accessed 20 July 2022). The last two competitions that have been completed were the M4 and M5. In the M4, participants submitted 61 forecasting models, which were used to predict 100,000-time series. At the same time, the M5 competition offered hierarchical data of Walmart sales across different stores, categories, and departments.
The main findings from these competitions were as follows:

1. Combining forecasts lead to higher accuracy
2. Simple machine learning methods are superior (in this case)
3. More complex methods can lead to a greater forecasting accuracy
4. Cross-learning can be useful for correlated series
5. Explanatory variables (trend, seasonality, and level) assist accuracy of forecasts

(Makridakis et al. 2020; Makridakis et al. 2022).
The M4 competition used multiple metrics to determine the winner. To define the overall performance of the models, they introduced the overall weighted average, which was the arithmetic mean of the symmetric Mean Absolute Percentage Error (shape) and the Mean Absolute Squared Error (MASE) (Makridakis et al. 2020). The winner of the M4 competition used a hybrid model using Holt-Winters exponential smoothing in conjunction with a neural network. Most of the top models in the M4 competition were statistical models that were combined in various permutations.(Petropoulos and Makridakis 2020). Due to the timeframe and large dataset provided for the M4 competition, many entries, even those using simple statistical models struggled to meet the deadline for entries. One model submitted after the deadline was the Neural Basis Expansion Analysis for interpretable Time Series forecasting (N-BEATS). N-BEATS is a novel deep learning forecasting model that was able to beat the winner of the M4 competition by 3%.

In the M5 competition, most of the methods examined utilized LightGBM, likely due to its effective handling of multiple features. The competition's winner also utilised a combination of LightGBM models taking the arithmetic mean of the models, followed by second place used a combination of LightGBM models and N-BEATS, a deep learning neural network for time series forecasting. The third-place contestant also used a combination of models, although these were deep learning containing multiple LSTM layers. This pattern of LightGBM combinations followed suit for the top 50 entries in which a method description was made available. (Makridakis et al. 2022).

Overall, specifically with forecasting Ethereum gas fees, there seems to be a preference given to models based on the RNN architecture, with models like LSTM, GRU and Deep AR being explored due to the ability for memory retention and being able to represent complex formulas. As well as that, in the m-competitions, most of the winning submissions used combinations of either the same or different models.

4 Methodology

4.1 Time Series Forecasting

A time series is merely a sequence of data points collected periodically or sporadically over a given period of time. Much data collected today is in the form of time series, such as the number of sales, stock prices, weather, and body monitoring. Knowing what has happened is incredibly useful to businesses but seeing how a particular stock would perform or how many items you need in inventory next month is invaluable- this is where time series forecasting is used. Making educated guesses using pre-existing data or future known events is an incredible tool as resources can be delegated to suit future demands and reduce costs. To understand how different models perform in forecasting Ethereum gas fees, I chose various models, including statistical, machine and deep learning. Many of these methods were used or mentioned in relation to the M-Competitions. The other techniques were mentioned in various papers based on Ethereum gas fee forecasting.

4.2 Statistical Methods

4.2.1 Exponential Smoothing

Exponential Smoothing as a time series forecasting method has many variations that are to be used in a variety of circumstances. Exponential smoothing methods are forecasts, which are produced using weighted averages of past observations–these weights will decay exponentially as the age of the o, allowing more recent observations to have a larger weighting in the calculation. The most common variations of exponential smoothing are known as Simple Exponential Smoothing, Holt's Linear Trend Model and Holt-Winters.

The **Simple Exponential Smoothing (SES)** model is most suited to forecasting univariate data with little to no trend or seasonality. SES takes in the parameter alpha (α) which is typically referred to as the smoothing coefficient. (Holt 2004; Forecasting: Principles and Practice (2nd ed).) Since the alpha value is a coefficient for exponential smoothing, it is typically set between 0 and 1, with an $\alpha = 0.5$ used as a starting point for many forecasts. The value of alpha determines the learning rate of the model, where larger values will favour more recent observations, whereas the older observations will decay faster. A value closer to 0 would imply a slow learning rate where older observations are considered and do not decay as quickly.

Holt's Linear Trend (HLT) model expands on SES by introducing a second hyperparameter beta (β). In addition to the predefined alpha parameter, the beta parameter determines the impact that the influence of the overall trend will have on the forecast. Depending on the type of trend that the time series contains, the trend type must be defined accordingly. For exponential smoothing with a linear trend, the trend type is expressed as Additive; for exponential trends, this is described as a multiplicative trend. However, for larger time-series forecasts, the trend may not be strictly defined as either multiplicative or additive over time which is where phi (ϕ), the damping parameter, is introduced. Dampening will reduce the size of the trend over

time eventually converting the trend into a straight horizontal line. Phi, itself, is used to determine the rate of dampening, whereas the type of dampening can also be multiplicative or additive.

For time series that contain seasonality, there is the exponential smoothing method known as **Holt-Winters; building** on top of the HLT model, a new parameter gamma (γ) is introduced to allow for control over the rate at which the influence of the seasonality is controlled. To determine the type of seasonality, once again, there are additive and multiplicative modes which can be selected following the same rules as additive and multiplicative trends. Finally, one more parameter is introduced to assist the seasonality modelling portion of the model, that is, the seasonal period (p) which is given in terms of the timeframe of the series. For example, if you had hourly data with a seasonality period of everyday $p = 24$. (Winters 1960; Holt 2004)

Choosing these parameters can be a difficult task, which is why determining α, β, ϕ and γ is best done by optimising the parameters to find the model with the smallest error.

4.2.2 Theta Model

The Theta model expands on exponential smoothing and naïve drift and is a univariate forecasting method. The approach for the theta model is to modify the local curvature of a given time series. By using a coefficient theta (θ) and applying it to the second differences of the time series:

$$X''_{new}(\theta) = \theta \cdot X''_{data}$$

Where:

$X''_{data} = X_t - 2X_{t-1} + X_{t-2}$ at a given time t.

The result of X''_{new}maintain both the mean and slope of the original data but does not include the curvature; these new time series are referred to as 'theta-lines'. As the value of theta is dilated, the deflation of the time series reduces, this means that a θ value of 0 would time series would be transformed into a linear regression line, whereas the larger values of theta ($\theta > 1$) would magnify the short-term behaviour of the time series.

In a general formularisation of the theta method, the initial time series is decomposed into 2 or more theta lines. Each of the theta lines is then extrapolated separately before being combined into a single forecast. Typically, there would be two theta lines one of $\theta = 0$, and the other $\theta = 2$. When combined, you obtain the linear regression of the time series which is extrapolated in the standard way for a linear trend and the second is extrapolated using exponential smoothing and the theta model forecast is given as $\frac{1}{2}\big(L(\theta = 0) + L(\theta = 2)\big)$ where $L(\theta = x)$is the theta-line of $\theta = x$.

The advantage of the theta model approach is that both short-term and long-term trends are considered and balanced equally. Whereas the theta-line of $L(\theta = 0)$ is there to provide a long-term trend line, $L(\theta = 2)$ provides insight into short-term fluctuations, which provides a balance to wards both aspects of the time series. (Assimakopoulos and Nikolopoulos 2000)

4.2.3 Fast Fourier Transform (FFT)

The Fast Fourier Transform is an algorithm that calculates the Discrete Fourier Transform (DFT) of a sequence(Fast Fourier transform - Wikipedia, accessed 19 August 2022). When first introduced, the DFT

algorithm was incredibly slow, and required $O(N^2)$ arithmetic operations. The FFT algorithm was able to compute sequences of size N in only $O(NlogN)$ operations, which when you have datasets in the hundreds or even thousands make a drastic difference in speed.

This increase in speed is mainly because an FFT algorithm is based on the factorisation of N, while this does imply that if N is a prime number, it will not be suitable for factorisation, there are less popular algorithms that can handle such problems. Plotting the number of arithmetic operations required for a DFT compared to a FFT (Fig 1) clearly demonstrates the FFTs superior performance over DFT for larger datasets.

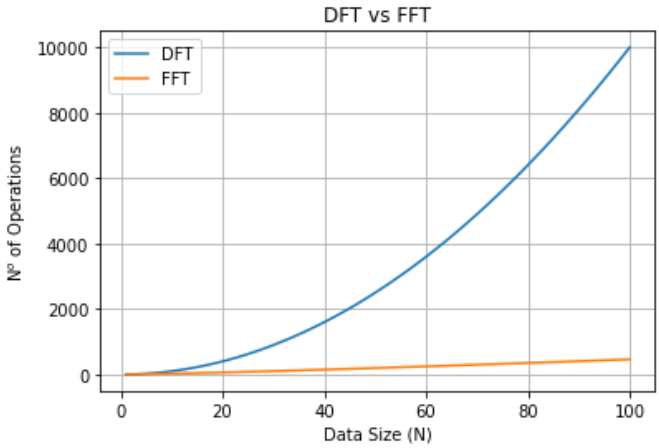

Figure 1: Number of operations for DFT vs FFT

The term FFT covers various algorithms that meet the criteria $O(NlogN)$. The most known and used algorithm for FFT is known as the Cooley-Tukey. A divide and conquer algorithm that recursively breaks the DFT down into a composite of any size $N = N_1N_2$.

4.3 Machine Learning Methods

4.3.1 Random Forest

The random forest is a very simple model; typical parameters include $n_{estimators}$ which define the number of trees the algorithm will build before taking the average of all predictions. Logically, a higher number of trees would slow down computation time as well as theoretically increase performance due to a larger number of samples to average thus making predictions more stable. In addition, there is also the parameter to determine how many features the random forest considers before splitting a node known as $\max_{features}$ and the minimum number of leaves required to split an internal node which is referred to as $\min_{sample\ leaf}$. One of the significant benefits of random forest is the ability for best feature selection, as each tree selects a random subset of features and will find the best feature within that subset. This allows for a broader diversity in the model, in addition to the prevention of overfitting the model (Breiman 2001; Lin et al. 2017).

4.3.2 Light Gradient Boosted Model

LightGBM is based on a decision tree algorithm typically used for classification tasks, however, it can also be used for regression and other types of tasks.

When first introduced, LightGBM proposed two novel ideas: Gradient-based One Side Sampling (GOSS) and Exclusive Feature Building (EFB). GOSS works by organising the absolute value of their gradients in descending order and selecting the top $\alpha \times 100\%$, whilst also randomly sampling $b \times 100\%$ instances from the remaining data. The focus is then shifted by taking the sample data and multiplied with the small gradients by a constant of $\frac{a-b}{1}$ shifting the focus to the under-trained instances without drastically affecting the original distribution (Ke et al. 2017).

Exclusive Feature Building is another aspect of optimising the algorithm. Since data tends to have mutually exclusive features EFB can speed up the tree learning by grouping certain features together, this allows the model to downsize the sample features (Ke et al. 2017). Both features combine to create a model that is both accurate and efficient at forecasting.

4.4 Deep Learning Models

4.4.1 Long-Short Term Memory

LTSM, which stands for Long-Short Term Memory, is a type of Recurrent Neural Network (RNN) and, unlike typical feedforward networks, contains feedback connections. The diversity of applications of LSTM makes it an incredibly lucrative model that can be applied to almost every circumstance, especially time series forecasting.

One of the main benefits of LSTM models is the ability to learn long-term dependencies, especially in sequence prediction problems, such as time series, whereas standard RNNs would have trouble with this type of problem due to the vanishing gradient problem. LSTMs contain three different gates, which are, in their own respect, neural networks. These gates are essentially filters where there is a “forget gate”, “input gate”, and “output gate”.(Cahuantzi 2021)

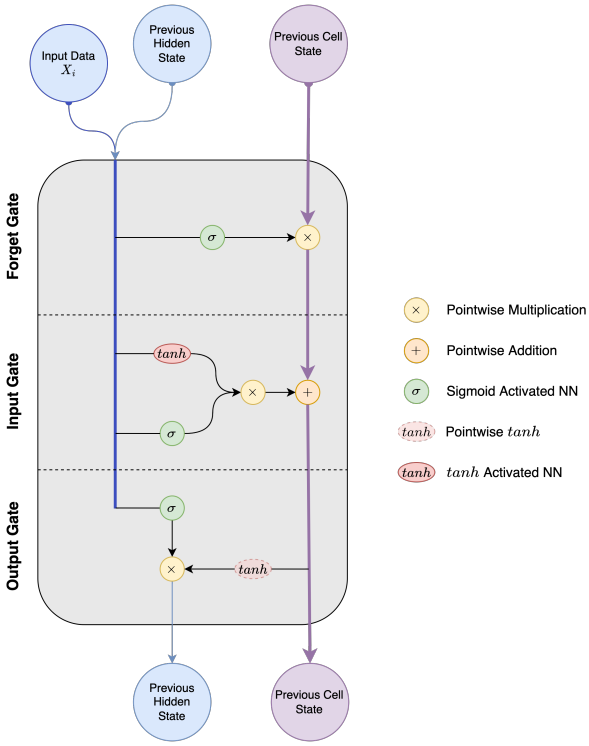

Figure 2: LSTM architecture diagram

LSTMs architecture (Fig 2) allows for the retention of both long- and short-term memory, meaning that trends such as seasonality from yearly to daily can be captured. In addition to ignoring redundant memory items due to the forget gate.(Gers et al. 2000)

4.4.2 Gated Recurrent Unit

Like the LSTM network is the Gated Recurrent Unit (GRU). Introduced by Cho, et al. (2014), the GRU contains gating units that regulate the information that flows throughout the unit, although, with a simple, computationally cheaper architecture (Fig 3).

GRU achieves this by reducing the number of gating signals in a typical LSTM model down to two, the update gate and the reset gate. In addition to the hidden state and cell states being merged. Comparable to the LSTM, there is a gate to handle the short-term memory (i.e., the hidden state) which is the reset gate. Whereas the long-term memory of the network is handled by the update gate.

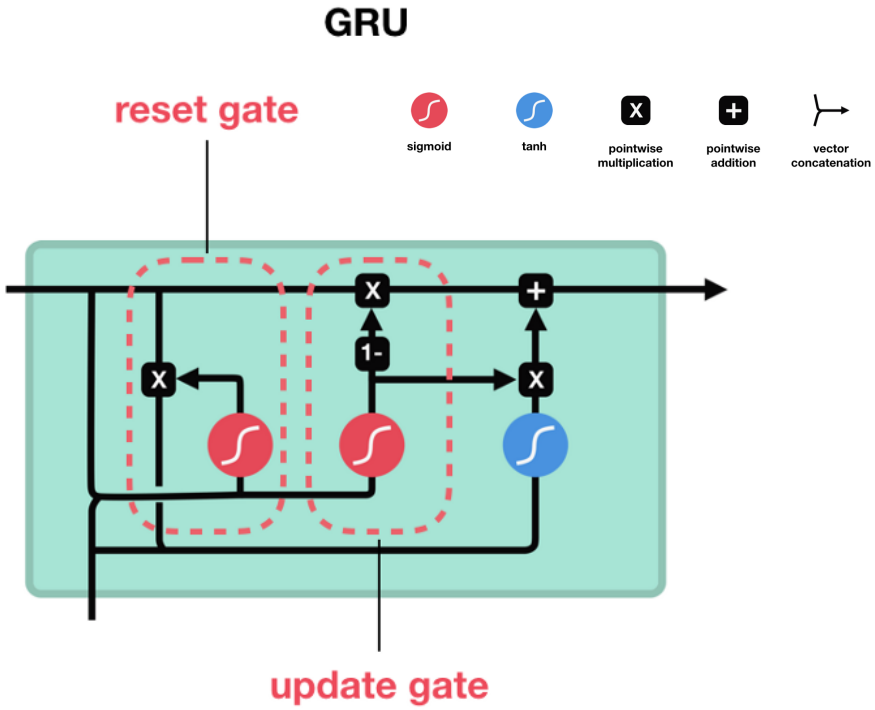

Figure 3: GRU architecture diagram (Phi, 2022)

This approach allows GRU to work more efficiently than LSTM models in cases where the time series data is complex however, GRUs learning capabilities are stressed and accuracy decreases to a point where LSTM is the favourable model (Cahuantzi 2021).

4.4.3 DeepAR

Amazon DeepAR Forecasting algorithm is a supervised learning algorithm for probabilistic forecasting that uses recurrent neural networks (RNNs). Within Amazon, they use forecasting to predict product and labour demand for critical events throughout the year, such as Prime Day, Black Friday and many more. The AR in the name stands for Auto-Regressive; this is related to how the model predicts values.

The DeepAR algorithm is based on an LSTM-based RNN model, allowing the ability to learn dependencies between items in a sequence, making it ideal for forecasting time series problems. Furthermore, DeepAR combines both LSTM and a sequence-to-sequence model meaning the encoder is trained first on the conditioning data range, then outputting an initial state h. This output is then transferred to the prediction range through the decoder. Both the encoder and decoder share the same architecture (Fig 4).

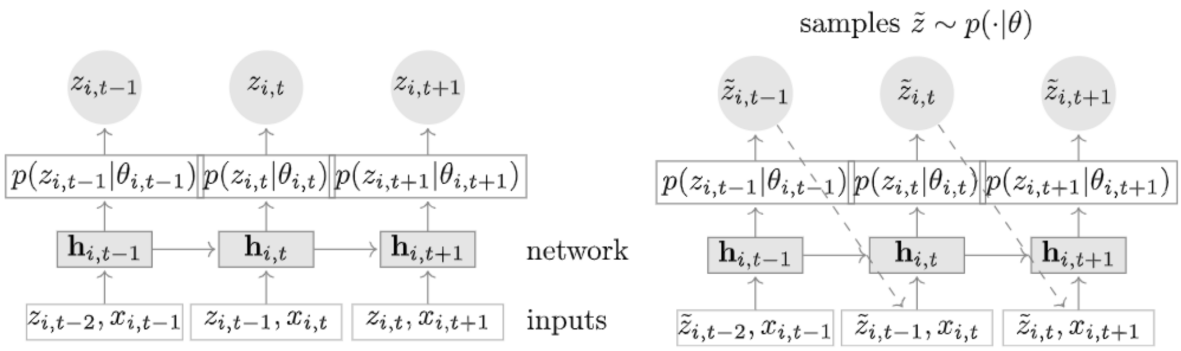

Figure 4: DeepAR architecture diagram (Salinas et al. 2017)

DeepAR conducts its own feature engineering meaning it's able to capture complex group dependant behaviour the model can conduct feature engineering on its own accord without user input. In addition, with the ability to learn from similar models, the method can provide forecasts for models with very little to no data. Therefore, the more features available to DeepAR to train on, the better the expected results should be (Ferenczi and Bâdicâ 2022).

4.4.4 N-Beats

Due to timing limitations in the M4-competitions the author of N-BEATS algorithm couldn't enter the competition, although, comparing the results of N-BEATS to the winner of the M4-competition the N-BEATS model was able to beat the winner by 3%. N-BEATS solves univariate time series forecasting problems using deep learning. The architecture is based on backward and forward residual links and a deep stack of fully connected layers (Fig 5).

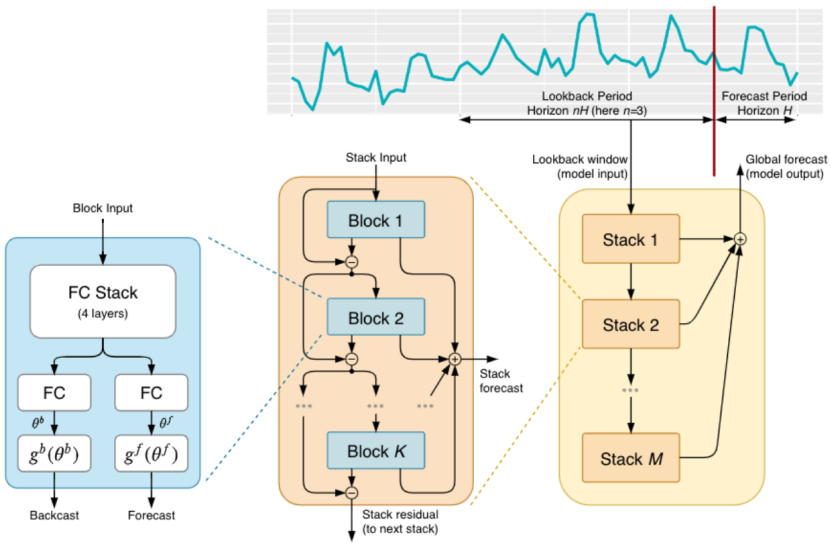

Figure 5: N-BEATS architecture diagram (Oreshkin et al. 2019)

N-Beats design methodology follows three main principles; the architecture should be simple and generic yet deep enough to be expressive. In addition, it should also be able to not rely on input scaling and time-series specific feature engineering. Finally, to be able to explore interpretability, the output of the architecture should be easy to read by humans (Oreshkin et al. 2019).

Since the architecture is specifically designed for time series forecasting N-BEATS allows the stacking of numerous blocks without causing a vanishing gradient problem, as well as taking advantage of ensemble models where the forecast is the sum over several blocks.

4.4.5 Temporal Fusion Transformer (TFT)

The Temporal Fusion Transformer (TFT) model is a multi-horizon forecasting method combined with a "novel attention-based architecture" that draws focus on interpretability (Lim et al. 2021) that provides the ability for the model to learn temporal relationships at various scales by using recurrent sequence to sequence layers for short-term and multi-head attention blocks for long-term dependencies. TFT is comprised of 5 canonical components that grants for increased performance. The architecture of the network means TFT can provide adaptative depth and complexity as a 'one size fits all' solution to multiple time series.

By splitting processing into two parts: local processing, which focuses on the characteristics of specific occurrences, and global processing, which captures the common characteristics of all-time series the TFT architecture means that heterogenous time series can be trained meaning the model can be applied to a wide variety of time series.

4.5 Metrics

Metrics are used to condense large volumes of data into a single value, however, each metric does so in a different way meaning each metric will emphasise only certain characteristics of the errors. Hyndman and Koehler, 2006 suggested that most metrics can be organised into four different categories: Relative-Error Metrics, Percentage-Error Metrics, Scale-Free Error Metrics and Scale-Dependent Error Metrics. By using a combination of at least two of these metric categories I ensure that bias is mitigated, and a justified conclusion can be made. Although since the data I will be using will be normalised some of these categories are not required.

The metrics I chosen to assess the performance of deterministic models are sMAPE, MASE, R2 Score, RMSE and OWA.

Symmetric Mean Absolute Percentage Error (sMAPE) is a percentage error metric bounded between 0-200% where the lower means that there are less errors in the prediction. This bounding eludes the problems of large errors when the true values are close to zero.

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - y_i^*|}{\frac{1}{2}(|y_i| + |y_i^*|)}$$

Where:

y_i = True values

y_i^*= Predicted values

Proposed by Hyndman and Koehler, 2006 Mean Absolute Squared Error (MASE) is a scale-free error metric, calculated using the Mean Absolute Error (MAE) and dividing it by the MAE of in-sample data naïve benchmark, the formulae is given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - y_i^*|$$

$$MASE = \frac{MAE}{MAE_{in-sample,naive}}$$

R2 Score otherwise known as coefficient of determination is used in regression problems such as time series forecasting with a score of 1 demonstrating a perfect fit.

Root Mean Squared Error is another scale-dependent error metric that can avoid mean squared error (MSE)'s loss of unit, although not necessarily needed, the advantages are the same as MSE

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - y_i^*)^2}$$

The overall weighted average (OWA) was introduced in the M4 competition to gauge performance of several models by simply using the combined average of the sMAPE and MASE. However, I introduced the average of both the R2 score and RMSE, so all error metrics used are represented in the OWA. I also inverted the R2 score as it had a range from $-\inf \rightarrow 1$ meaning the lower the OWA score, the better the overall performance of the model.

5 Data

5.1.1 Sourcing Data

The data was obtained from multiple sources, majority of the data was made available by Google BigQuery on Kaggle. Kaggle contained a dataset that updates daily with all the historical information stored on the Ethereum blockchain. Within the dataset there are 7 categories: blocks, contracts, logs, token_transfers, tokens, traces, and transactions. Much of the information held in these categories are strings used to describe the activity occurring on the blockchain, which wasn't needed for prediction of Ethereum gas fees. In addition to the blockchain data, financial data was sourced from Yahoo Finance. The features chosen for this investigation were as follows:

Table 1: Features to be used in models

Source	Feature	Description
Kaggle/Blocks	size	The size of this block in bytes.
	gas_limit	The maximum gas allowed in this block.
	gas_used	The total used gas by all transactions in this block.
	total_difficulty	Integer of the total difficulty of the chain until this block.
	difficulty	Integer of the difficulty for this block.
	transaction_count	Number of transactions in the block.
	timestamp	The timestamp for when the block was collated.
Kaggle/Transactions	gas_price	Gas price provided by the sender in Wei.
	value	Value transferred in Wei.
	block_timestamp	Timestamp of the block where this transaction was in.
Yahoo Finance	ETH/USD	Ether market price in United States Dollars.
	BTC/USD	Bitcoin market price in United States Dollars.
	ETH/BTC	Ether market price in Bitcoin.

These variables gave me a wide variety of information to be used that captured the quantitative aspects of the Ethereum blockchain as well as the potential external influencing factors of Ethereum gas price. All variables aside from gas_price will be used as past covariates to aid forecasting predictions.

5.1.2 Pre-Processing Data

To be able to assess the granger causality and other features of the data, required converting the features into a unanimous 12-hour timeframe. The reasoning behind choosing a 12-hour timeframe was to ensure that the data that was sourced from the Ethereum blockchain did not lose the characteristics such as noise and trend of the data. Due to spikes in the network, there were times where gas prices became outliers–to counter this I mean imputed values that were at least 3 standard deviations from the mean. Whereas the financial data from Yahoo Finance was daily so using mean imputation for the 12th hour of each day would still accurately capture the pricing data.

Furthermore, majority of data was available from around roughly 2015 onwards depending on the source, although, since the Ethereum network is constantly receiving updates, drastic modifications to the gas price algorithm mean that previous data would be irrelevant to forecasting current Ethereum gas prices. Taking this into consideration meant I had to look at the history of Ethereum updates (https://ethereum.org/en/history). The most impactful change of gas prices came in the Istanbul update where the main change affected the Ethereum Virtual Machine (EVM) which the runtime environment for smart contracts. Using that information, I set the timeframe for the data to be 09-12-2019 which was the day after the Istanbul fork till 08-08-2022. Ferenczi and Bâdicâ (2022) went with a training/validation split of 80%/20%, after experimenting with various splits I found this also gave me the best results for the statistical models and would allow a large enough time frame for both validations to ensure the data doesn't overfit and back testing can be applied on the training dataset.

5.1.3 Data Analysis

To decide on features that would positively contribute towards forecasting Ethereum gas prices, I took inspiration from Antonio Pierro et al. (2019) and chose to assess both the correlation of Ethereum gas prices against the other features chosen in addition to the pair-wise granger causality. The strongest correlations were trans_count and btc_usd which had a correlation > 0.5, total_diff had a correlation of 0.0288 meaning there is very little to no correlation. All the other features showed a larger positive correlation but of no great significance. Sorting the correlated features gave a hierarchal order to the data which will allow for better training as I can use Python list slicing to remove the least correlated features quickly to improve accuracy of the prediction models.

The pair-wise granger causality test investigates potential causality between certain time series X and Y, using an F-Test it will test the null hypothesis that the lagged x-values do not explain the variation (Granger-cause) in y-values. Which represent causality related to precedence (Engle et al. 1983). Testing the pair-wise granger causality with lag values of [1,2,3,4,5,6,7,14,28,60,90,120] returned a list of features in which the null hypothesis was rejected which contained every column, although some features such as btc_usd (count = 10) appeared more than others even with a higher correlation such as trans_count (count = 2). Taking this into account allowed me to better understand the effect that each feature will have when forecasting Ethereum gas fees.
Many models such as Naïve seasonality, Holts-Linear Exponential Smoothing, and Theta method take in a seasonality parameter so to optimise such models understanding the seasonality of the data were working on is important. To do this, I used the Auto Correlated Function (ACF) on the Ethereum gas price time series, in addition to using Darts check_seasonality function. This returned a seasonality value of 2 which is clearly visible from the plotted ACF graph (Fig 6) there is a correlated value spike at an interval of 2 lags. This

means the Ethereum gas price data has a seasonality of 24hrs, which is a logical outcome as there will be various times of the day where network traffic is lower causing less demand.

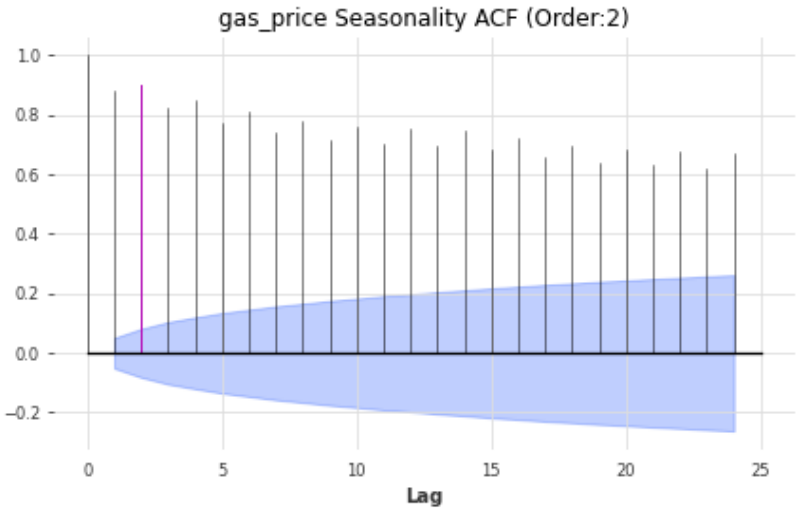

Figure 6: Gas price seasonality ACF

5.1.4 Optimising the models

Optimising the statistical and machine learning models was a relatively straight forward task. Using Darts' grid search function, I entered a list of possible values for each hyper-parameter to find the best combination, following I then did another grid search using the best hyper-parameters and their neighbouring values to find any possible room for improvement. I would do this a few times, depending on the model till the hyper-parameters did not require training. To assess the models that were being produced I used the metrics listed above on the validation dataset.

However, since grid search would create a model for every possible combination, the deep learning models would have taken far too long. For example, if I had put a list of 4 values for 3 hyper-parameters the grid search function would have run $4^3 = 64$ iterations of models where a large majority of them would have underperformed. Instead, I ran a grid search one hyperparameter at a time. Followed by using 'Ray Tuning' a Python module specifically designed for optimisation of machine learning models. Using the Asynchronous Successive Halving Algorithm (ASHA) to find optimal values for the deep learning models, specifically the dropout rate, batch size and input chunk length as well as a variety of other hyper-parameters specific to the model in training.

For the parabolic models I used the optimal hyper-parameters found in the deterministic models as a base and then ran a grid search using Laplace, Gaussian, Quantile Regression, Cauchy, and Exponential Likelihoods to find the best likelihood for the given model.

6 Results

6.1 Deterministic Models

Table 2: Statistical results of forecasting on validation dataset

Metric	Theta	N-BEATS	GRU	FFT	(Holts) Exp Smooth.	(Holt-Winters) Exp Smooth.	Random Forest	(SES) Exp Smooth.	LightGBM	TFT	LSTM
sMAPE	41.063	50.944	51.409	52.300	60.197	60.199	70.914	74.817	80.114	90.866	105.394
MASE	1.227	1.351	1.527	1.411	1.905	1.905	2.523	2.751	3.387	4.358	5.939
R2 Score	-0.432	-0.377	-0.050	-0.291	0.007	0.007	0.456	0.676	1.752	3.783	6.140
RMSE	0.114	0.119	0.147	0.127	0.151	0.151	0.182	0.195	0.250	0.330	0.403
OWA	10.493	13.009	13.258	13.387	15.565	15.566	18.519	19.610	21.376	24.834	29.469

Table 2. shows the performance of each model over the entire validation dataset. The validation dataset ranges over the span of just over 9 months which is 25% of the available data. Looking at the exponential models (Fig 7,8,9) they all follow the same general trend line of SES, although we see the difference between SES against Holt's and Holt-Winters where additional hyper-parameters were introduced however, due to the grid search optimisation used both Holt's and Holt-Winters exponential smoothing have near identical results. From the graphs of exponential smoothing, it's clear the overall trend has not been successfully forecasted and whereas the Holt variations of exponential smoothing are able to somewhat resemble the volatility of the dataset it's not enough to be able to confidently forecast Ethereum gas fees.

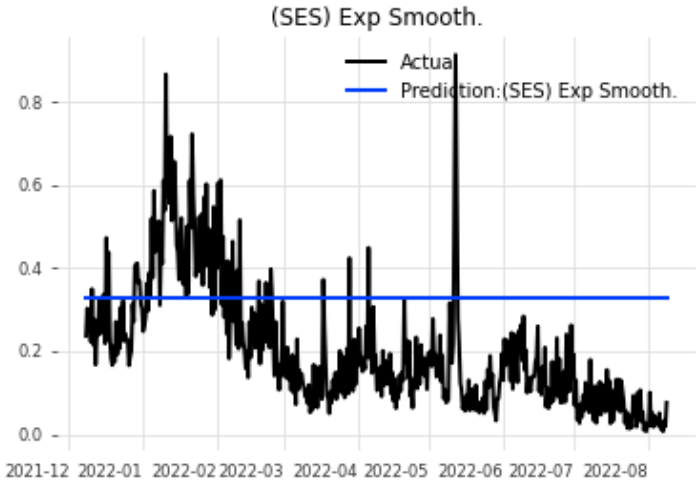

Figure 7: Simple exponential smoothing

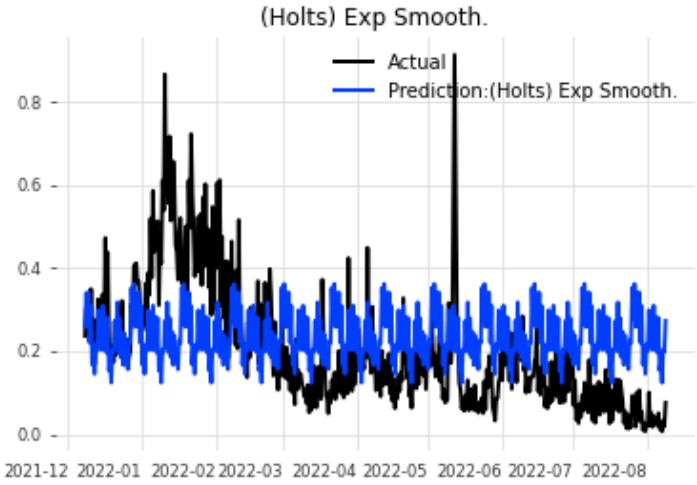

Figure 8: Holts Linear exponential smoothing

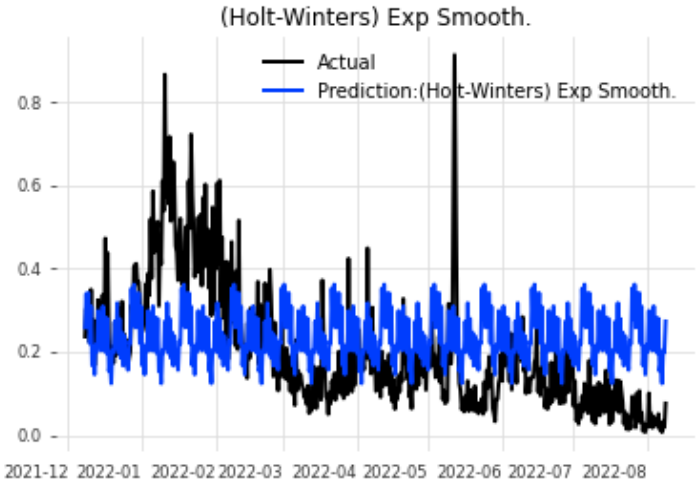

Figure 9: Holt-Winter's exponential smoothing

Moving onto FFT in (Fig 10) the opposite takes place where the FFT model was able to capture the global trend and as the forecast ages there is a decline within the forecast accuracy that is visually verifiable. The Theta model forecast (Fig 11) displays a similar pattern although the forecast contains much more noise which creates the thick blue prediction line. This is likely why the Theta model has the best performance according to the metrics, with a sMAPE of 41.063 and an OWA of 10.493 but looking at the plot (Fig 11) will show a different story once again, the trend is able to be captured but the predictions are inaccurate. This pattern within the statistical models demonstrates one of the greater limitations and why machine learning and deep learning can benefit time-series forecasting.

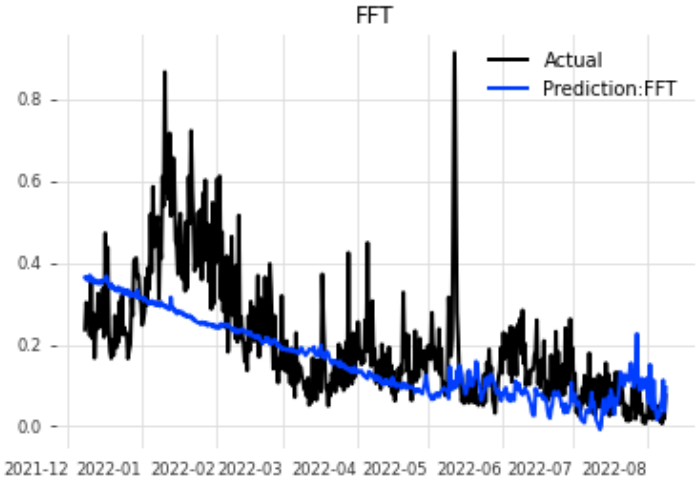

Figure 10: FFT forecast

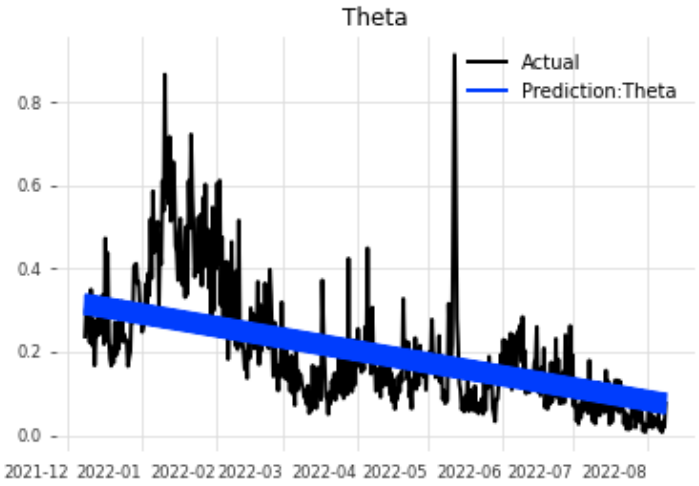

Figure 11: Theta forecast

For the machine learning models there is a clear improvement, even though both the Random Forest (Fig 12) and LightGBM (Fig 13) models contain higher values in their metrics the forecasts are better than the statistical models. One of the main reasons for this is due to the size of the validation set every model so far has struggled with predictions. The Random Forest forecast can capture the volatility of the gas price although struggled with the direction of the trend and remained relatively flat throughout. LightGBM displays one of the best forecasts of all the models during roughly the first 2 months and as the accuracy deteriorates over time this causes the metrics assessing the model to increase.

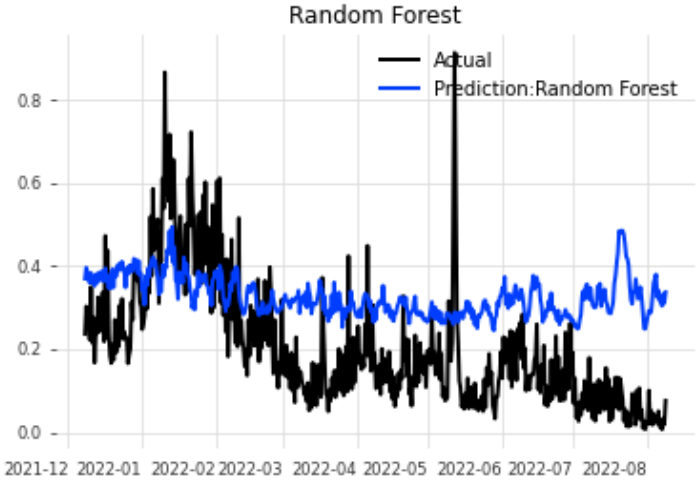

Figure 12: Random Forest forecast

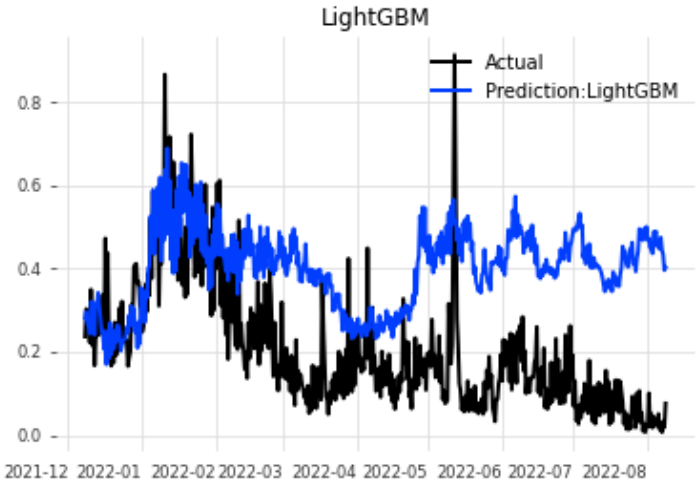

Figure 13: LightGBM forecast

On the bottom end of the performance table, we see the LSTM (Fig 14) and TFT (Fig 15) models. Although they went through the same optimisation procedure as the other two deep learning models, they still had the worst performance with a sMAPE value of 105 and 90, respectively and widely inaccurate predictions with the TFT model being incredibly volatile. This suggests that a different approach to the optimisation should have been considered. On the other end of the results table at the podium, there are the other two deep learning models N-BEATS (Fig 16) and GRU (Fig 17). Although the N-BEATS forecast is rather volatile, it was able to capture the trend of the validation dataset almost exactly demonstrating the ability of deep learning to capture complex trends. Following N-BEATS there is the GRU forecast which is only a simple trend line that seems to ignore the noise within the gas price and predicts only the global trend of the validation dataset.

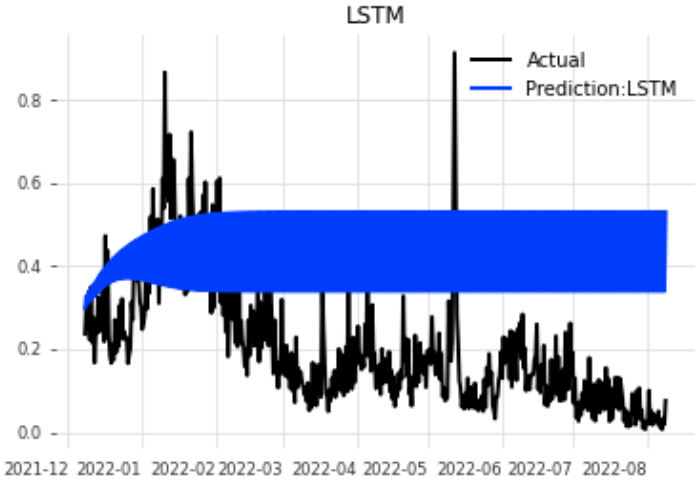

Figure 14: LSTM forecast

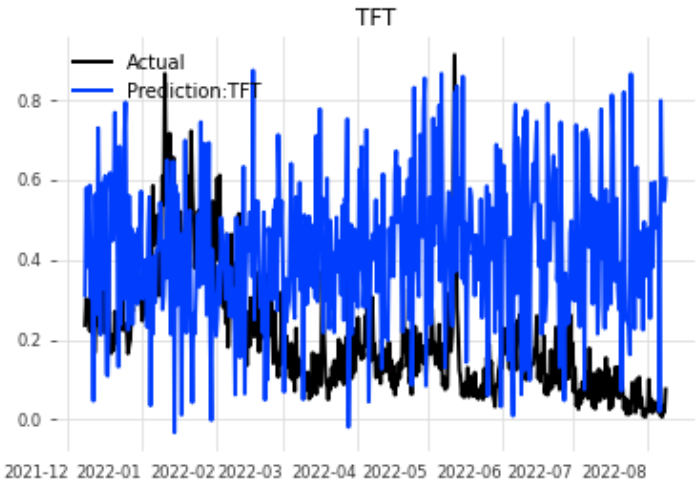

Figure 15: TFT forecast

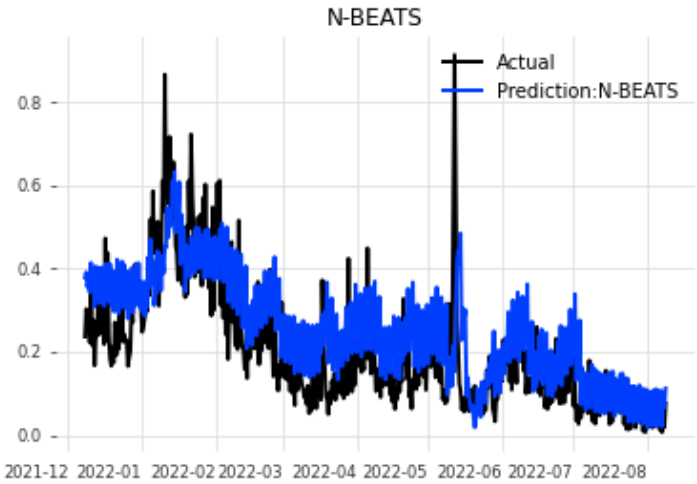

Figure 16: N-BEATS forecast

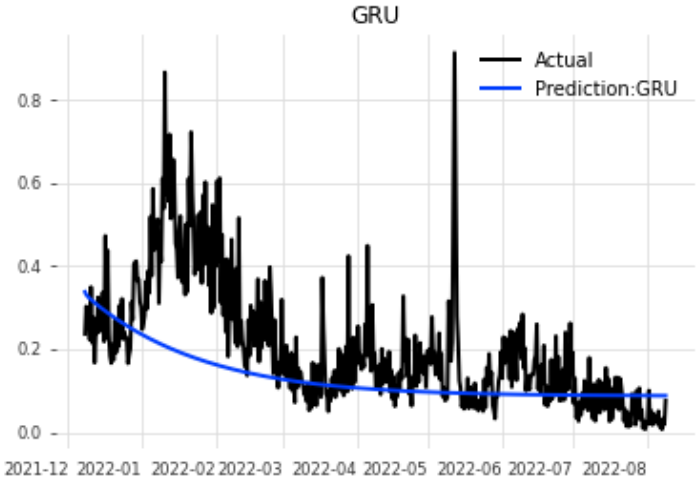

Figure 17: GRU forecast

6.2 Probabilistic Models

Table 3: Statistical results of probabilistic forecasts on validation dataset

Rho Risk	N-BEATS_QuantileRegression	GRU_QuantileRegression	DeepAR_LaplaceLikelihood	TFT_LaplaceLikelihood
0.05	0.266	0.004	0.097	0.080
0.1	0.265	0.116	0.179	0.150
0.5	0.188	0.602	0.638	0.568
0.9	0.046	0.210	0.598	0.699
0.95	0.024	0.117	0.402	0.647
OWA	0.158	0.210	0.383	0.429

The rho risk with specific quantiles in addition to the OWA was used to measure the forecasting capabilities of the probabilistic models. Once again, the N-BEATS (Fig 18) and GRU (Fig 19) models performed the best overall. The N-BEATS probabilistic forecast seems to have even performed better than the deterministic model once again following the almost exact trend of the validation dataset although this time with less volatility. Looking at the forecast N-BEATS was able to keep a very low risk within the 90th + percentile. Notably, the GRU model has a lower risk within the 5th and 10th percentiles implying a more confident forecast at the lower percentiles. However, we see the risk greatly increase when compared to N-Beats for the 50th + percentiles which explains how N-BEATS has the overall lower OWA. When using a probabilistic forecast on the deterministic LSTM model it becomes the DeepAR model due to the way it is compiled within the Darts library. Both the DeepAR (Fig 20) and TFT (Fig 21) model have the highest rho risk for most quantiles 1 and Y-Axis increasing to [-2,2] on each plot to accommodate the 1-99th percentile predictions.

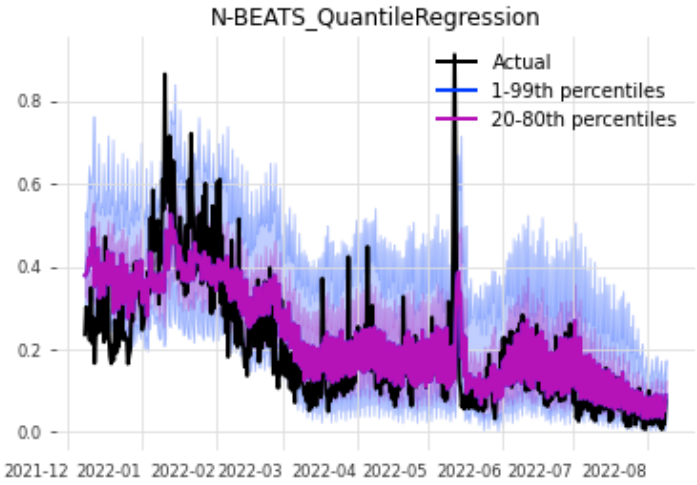

Figure 18: N-BEATS probabilistic forecast

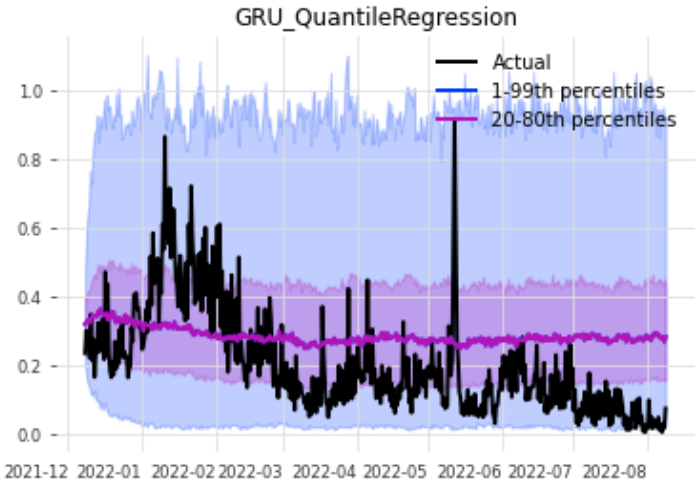

Figure 19: GRU probabilistic forecast

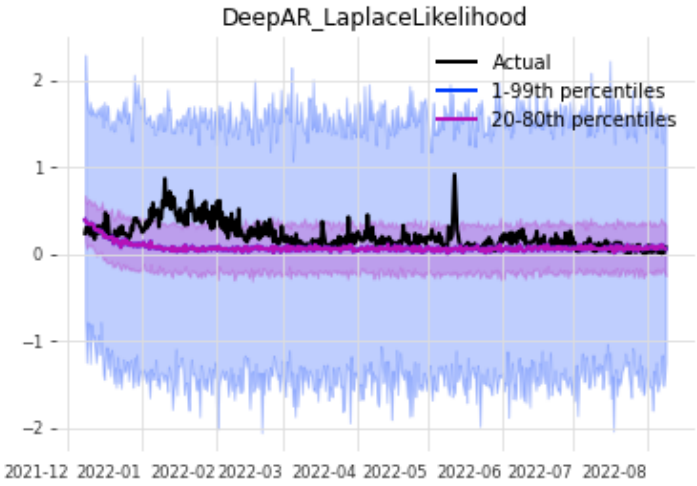

Figure 20: DeepAR probabilistic forecast

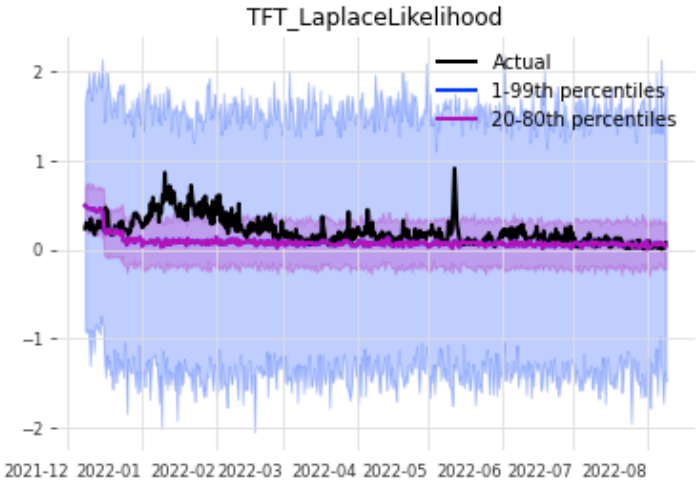

Figure 21: TFT probabilistic forecast

6.3 Back-testing

To assess each models' capabilities to forecast given certain time frames 3 separate back tests were performed on each model using the sMAPE metric to evaluate the performance of each iteration. A back test will take a certain forecast horizon across the time series and produce forecasts according to the length of the forecast horizon given with a stride (typically 1) each iteration till the end of the given time series. In this case, all back tests were performed with a stride of 200 (100 days) and a start of 0.5 which means starting halfway through the time series.

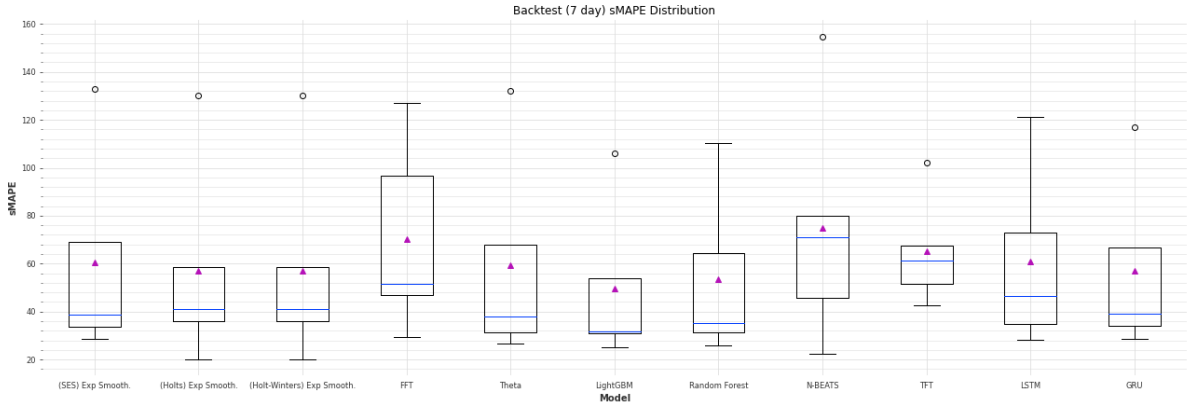

Figure 22: 7 Day back test

The first back test (Fig 22) was performed with a forecast horizon of 14, i.e., 7 days as expected the best preforming model with the lowest median (blue lines) sMAPE was the LightGBM model, on the opposite end there is the FFT, Random Forests and LSTM models all with sMAPE values reaching over 100. However, looking at the median values of all models they have performed relatively well, specifically the models which do not use deep learning.

The next back test was for 28 days (Fig 23) This is where we begin to see the many models, including LightGBM preforming much worse with a larger range of results reaching larger sMAPE values. Whereas most models have a noticeable difference the inter-quartile range (IQR) of the GRU model has shrunk compared to the 7 days back test.

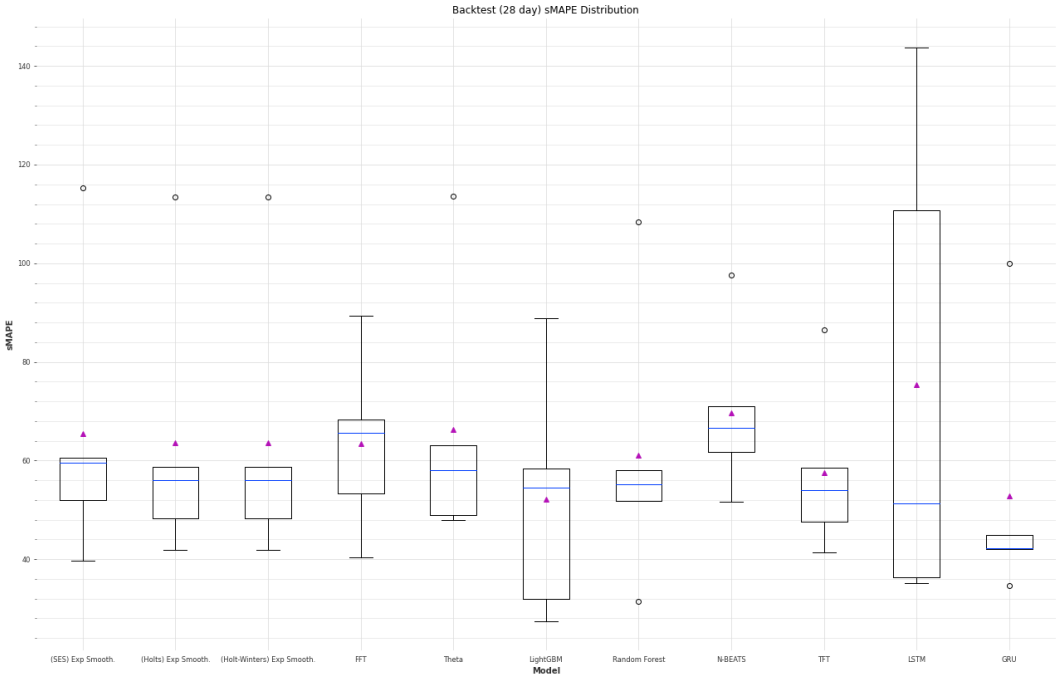

Figure 23: 28 Day back test

The final back test was performed on a forecast horizon of 90 days (Fig 24) once again the performance of the models deteriorate greatly, demonstrating impact of long-term forecasts. Following the same trend as the previous back cast the GRU model seems to have the best general performance of forecast at the 90-day timeframe albeit with a much larger IQR.

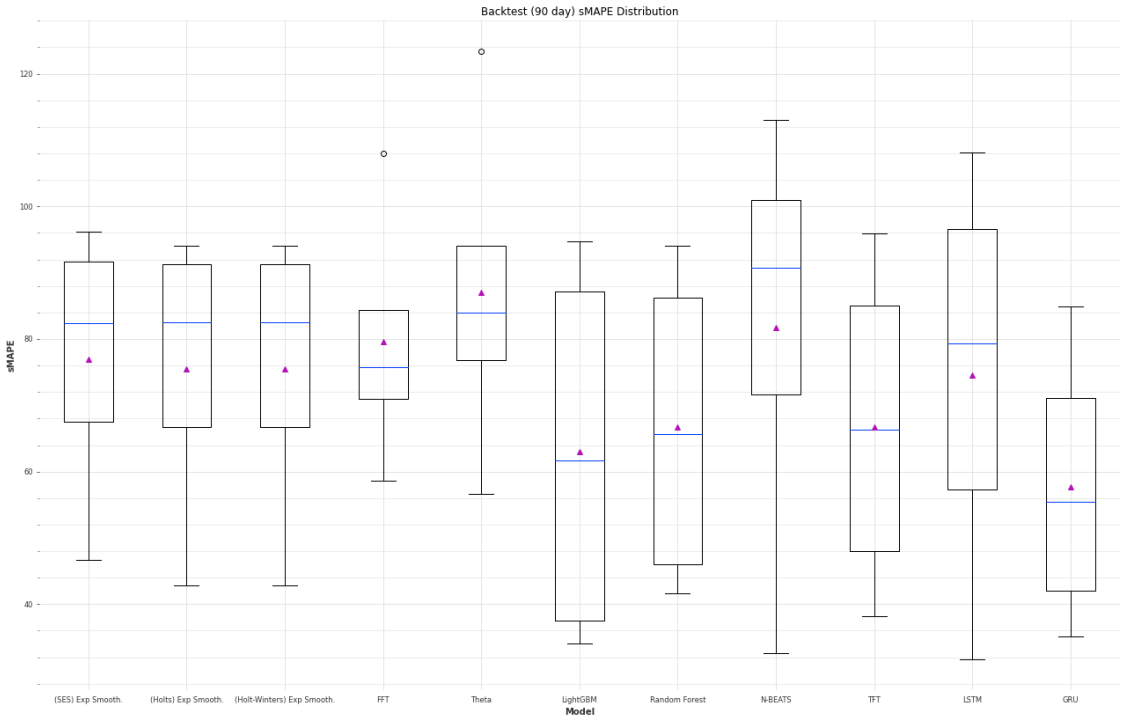

Figure 24: 90 Day back test

7 Discussion

Many of the models tested have been able to capture either, long-term trend, short-term trend, or volatility. However, none have been able to capture all 3 of the characteristics perfectly. Looking at (Table 2.) there is a mix of the different types of models that were tested with the Theta model having the lowest OWA, however, when looking at the Theta forecast (Fig 11) clearly, it would not be suitable to base a gas price prediction from.

The statistical models all performed relatively well and quick according to the metrics used to measure the performance of each forecast. Due to grid search both the Holt's Linear Trend and Holt-Winter's Exponential Smoothing models turned out almost identical. The exponential smoothing models didn't even capture the full trend of the validation dataset in addition, the hyper parameters used only captured the daily seasonality which is clearly shown in the forecast yet provides very little benefit to the prediction. The FFT and Theta models were both able to capture the trend of the data but lacked the ability to predict the volatility of the dataset. On the other hand, the two machine learning models Random Forest and LightGBM were both able to represent the volatility even though the Random Forest model had no trend line the volatility was well captured. This holds even more true for LightGBM although as the time series expands further into the future it is clear there is no trend considered for LightGBM. Given that the FFT and Theta models can forecast the trend where as LightGBM and Random Forest can forecast volatility a combination of these models may prove beneficial.

One of the best forecasts both statistically and visually is the N-BEATS model, outperforming all the other deep learning models with an incredibly accurate visual forecast it's logical to assume that this may be one

of the best models to forecast Ethereum gas fees, although, given the performance N-BEATS in the back tests, it may not be a wise idea to solely use just the N-BEATS model.

Many of the models forecasted all have their respective pros and cons meaning finding the correct balance when combining these models would likely prove to be more successful at forecasting than using a sole model. There is also the possibility of training several of the same models and combining them to create a more successful model, like the winning method of the M4-Competition.

Furthermore, many more improvements can be made. For starters using a larger dataset for training such as hourly blockchain information and excluding financial information which is daily could lead to more accurate results. In addition to training the models on a shorter forecasting horizon as shown in the back testing, the larger the forecast horizon the higher sMAPE value.

8 References

Antonio Pierro, G., C Rocha, H.S., Antonio Pierro, G. and Rocha, H. 2019. The Influence Factors on Ethereum Transaction Fees., pp. 24–31. Available at: https://hal.inria.fr/hal-02403098 [Accessed: 20 August 2022].

Assimakopoulos, V. and Nikolopoulos, K. 2000. The theta model: A decomposition approach to forecasting. *International Journal of Forecasting* 16(4), pp. 521–530. doi: 10.1016/S0169-2070(00)00066-2.

Breiman, L. 2001. *Random Forests*.

Buterin, V. 2014. Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform.

Cahuantzi, R. 2021. *A comparison of LSTM and GRU networks for learning symbolic sequences*. Available at: https://github.com/robcah/RNNExploration4SymbolicTS.

Chuang, C.Y. and Lee, T.F. 2022. A Practical and Economical Bayesian Approach to Gas Price Prediction. *Lecture Notes in Networks and Systems* 309, pp. 160–174. Available at: https://blog.amis.com/a-practical-and-economical-of-gas-price-prediction-d9abe955ac63 [Accessed: 20 August 2022].

CoinMarketCap. [no date]. Available at: https://coinmarketcap.com/ [Accessed: 18 July 2022].

Engle, R.F., Hendry, D.F. and Richard, J.-F. 1983. Exogeneity. 51(2), pp. 277–304.

ETH gas - Google Trends. [no date]. Available at: https://trends.google.com/trends/explore?date=2017-01-07%202022-08-19&q=eth%20fees,ethereum%20gas,ethereum%20gas%20fees,gas%20fees,eth%20gas [Accessed: 19 July 2022].

Fast Fourier transform - Wikipedia, accessed 19 August 2022. [no date]. Available at: https://en.wikipedia.org/wiki/Fast_Fourier_transform?oldformat=true [Accessed: 12 September 2022].

Ferenczi, A. and Bâdicâ, C. 2022. Prediction of Ether Prices Using DeepAR and Probabilistic Forecasting. Available at: https://doi.org/10.1007/978-3-031-08754-7_73.

Gers, F.A., Schmidhuber, J. and Cummins, F. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* 12(10), pp. 2451–2471. Available at: https://direct-mit-edu.ezproxy.herts.ac.uk/neco/article/12/10/2451/6415/Learning-to-Forget-Continual-Prediction-with-LSTM [Accessed: 28 June 2022].

Holt, C.C. 2004a. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting* 20(1), pp. 5–10. doi: 10.1016/J.IJFORECAST.2003.09.015.
Holt, C.C. 2004b. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting* 20(1), pp. 5–10. doi: 10.1016/J.IJFORECAST.2003.09.015.
Hyndman, R.J. and Koehler, A.B. 2006. Another look at measures of forecast accuracy. Available at: www.elsevier.com/locate/ijforecast [Accessed: 22 July 2022].
Ke, G. et al. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Available at: https://github.com/Microsoft/LightGBM. [Accessed: 5 September 2022].
Lim, B., Arık, S., Loeff, N. and Pfister, T. 2021. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37(4), pp. 1748–1764. doi: 10.1016/J.IJFORECAST.2021.03.012.
Lin, L., Wang, F., Xie, X. and Zhong, S. 2017. Random forests-based extreme learning machine ensemble for multi-regime time series prediction. *Expert Systems With Applications* 83, pp. 164–176. Available at: http://dx.doi.org/10.1016/j.eswa.2017.04.013 [Accessed: 23 May 2022].
Makridakis, S., Spiliotis, E. and Assimakopoulos, V. 2020. The M4 Competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting* 36(1), pp. 54–74. doi: 10.1016/J.IJFORECAST.2019.04.014.
Makridakis, S., Spiliotis, E. and Assimakopoulos, V. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* . doi: 10.1016/J.IJFORECAST.2021.11.013.
Mars, R., Abid, A., Cheikhrouhou, S. and Kallel, S. 2021. A machine learning approach for gas price prediction in ethereum blockchain. *Proceedings - 2021 IEEE 45th Annual Computers, Software, and Applications Conference, COMPSAC 2021* , pp. 156–165. doi: 10.1109/COMPSAC51774.2021.00033.
Oreshkin, B.N., Carpov, D., Chapados, N. and Bengio Mila, Y. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. Available at: https://arxiv.org/abs/1905.10437v4 [Accessed: 21 June 2022].
Petropoulos, F. and Makridakis, S. 2020. The M4 competition: Bigger. Stronger. Better. *International Journal of Forecasting* 36(1), pp. 3–6. doi: 10.1016/J.IJFORECAST.2019.05.005.
*Phi, M., 2022. [online] Miro.medium.com. Available at: <https://miro.medium.com/max/1400/1*yBXV9o5q7L_CvY7quJt3WQ.png> [Accessed 9 August 2022].*
Salinas, D., Flunkert, V., Gasthaus, J. and Januschowski, T. 2017. DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks. *International Journal of Forecasting* 36(3), pp. 1181–1191. Available at: https://arxiv.org/abs/1704.04110v3 [Accessed: 21 June 2022].
Werner, S.M., Pritz, P.J. and Perez, D. 2020. Step on the Gas? A Better Approach for Recommending the Ethereum Gas Price. *Springer Proceedings in Business and Economics* , pp. 161–177. Available at: https://arxiv.org/abs/2003.03479v2 [Accessed: 20 August 2022].
Wikipedia Contributors [no date]. Makridakis Competitions - Wikipedia. Available at: https://en.wikipedia.org/wiki/Makridakis_Competitions?oldformat=true [Accessed: 20 August 2022].
Winters, P.R. 1960. Forecasting Sales by Exponentially Weighted Moving Averages. *Management Science* 6(3), pp. 324–342. doi: 10.1287/MNSC.6.3.324.