UNIVERSITY of INFORMATION TECHNOLOGY

ETHEREUM PRICE PREDICTION

An Analysis Using LSTM and ARIMA

Prepared By

UIT - 1120 Aung Khant Soe
UIT - 1147 Ei Thinzar
UIT - 1163 Yamin Eaindray
UIT - 1187 Myint Theingi Tun
UIT - 1232 Nadi Lin
UIT - 1367 Min Khant Zaw

Software Engineering, 4th Year

September 16th , 2024

TABLE OF CONTENTS

1. ABSTRACT

In recent years, cryptocurrencies have garnered a great deal of attention and for good reasons. The foremost among them can be called Ethereum. Investors, traders and financial analysts all require accurate price prediction of Ethereum to be able to make an educated guess. Here, we introduce a comparison report between two leading forecasting models: Long Short-Term Memory (LSTM) networks and Autoregressive Integrated Moving Average (ARIMA). Testing with historical price data is conducted to determine which models are better suited to predicting Ethereum prices. Results suggest that, whilst ARIMA serves as a robust baseline model for linear trends; LSTM provides better performance at capturing complex non-linear patterns underlying cryptocurrency markets. The results presented here indicate that machine learning strategies, not unlike the LSTM of interest to this research, are good performers for forecasting relevant financial time-series data.

2. INTRODUCTION

Cryptocurrency has completely changed the financial world, as we know it—providing decentralized alternatives to traditional assets.Ethereum, launched in 2015, has grown to be one of the biggest players due its flexibility with smart contracts and high adaptability. On the other hand, Ethereum price is extremely volatile making it hard for investors to get in and out at times that provide attractive risk-return opportunities.

Price prediction is important for developing trading strategies, risk management and investment planning. Such traditional statistical models like Autoregressive Integrated Moving Average (ARIMA)As in the case of time-series forecasting, some simple but interpretable techniques have a proven utility to provide accurate predictions. Conversely, machine learning models—especially Long Short Term Memory (LSTM) Networks—have been quite successful at capturing rich patterns and relationships in sequential data.

In this report, we will compare ARIMA with LSTM models on Ethereum price prediction. It examines the performance of each method using historical price data, to illustrate their potential application for financial forecasting.

3. THEORY BACKGROUND

Time-Series Models

Forecasting methods for time-series are an integral part of research across different disciplines largely sought after mainly due to its application in finance where accurate predictions of asset prices can incentivize substantial gains economically. Datasets such as stock prices, cryptocurrency prices and lending rates possess time-series behavior that relies on relationships between previous data points in the series to predict future values. Throughout the years, researchers have posed a plethora of models with unique specs and equilibria. The ARIMA (AutoRegressive Integrated Moving Average) model and LSTM(Long Short-Term Memory)-Neural Network are two of the most popular models used in time-series forecasting.

In this paper, we theoretically describe the ARIMA and LSTM models, analyze their differences from generation processes to parameter learning methods (strengths and weaknesses), background theory support of model establishment meaning in time-series analysis as well permissibly application in financial forecasting.

3.1 LSTM Model

LSTMs (Long Short-Term Memory) are a type of machine learning model that has been developed to understand time-series data with relatively complex, non-linear dependencies. LSTM is a special kind of RNN introduced by Hochreiter & Schmidhuber in 1997. LSTMs were designed to overcome the vanishing gradient problem, a significant limitation in standard RNNs that hinders learning long-term dependencies unlike traditional RNN models.

Strengths of LSTM:

- LSTMs capture non-linear relationships — Time series data is complex and often display patterns which are not simply additive but can have even periodic components (or triggers) etc, therefore a model needs to be able to manage that complexity.
- Whether it is long-term dependencies and short-term: This could be very vital to period series forecasting as some occasions in the past can hugely affect the near future. The LSTM network has got the sense of remembering enter values for a longer time or ignoring momentary inputs that do not have any influence on upcoming proceedings.

Weaknesses of LSTM:

- Requirements for data: LSTMs usually require a large volume of training data, which may not always be obtainable; particularly in a base where the cost to gather information is high.
- Training LSTM models is resource-heavy, both in time and hardware.

3.2 ARIMA Model

ARIMA is a classical statistical model that stands for AutoRegressive Integrated Moving Average, introduced by Box and Jenkins in 1970. This approach has established itself as the preferred method for time-series analysis and is particularly appropriate with respect to linear relationships and application of economic ensembles that have both upward trends (trends) and seasonality.

Strengths of ARIMA:

- Linear relationships: ARIMA models are most successful in tracing linear interdependencies within tabular statistics.
- ARIMA models have a long history and the literature covering them is plentiful which provides great interpretational value to the already built theory.

Weaknesses of ARIMA:

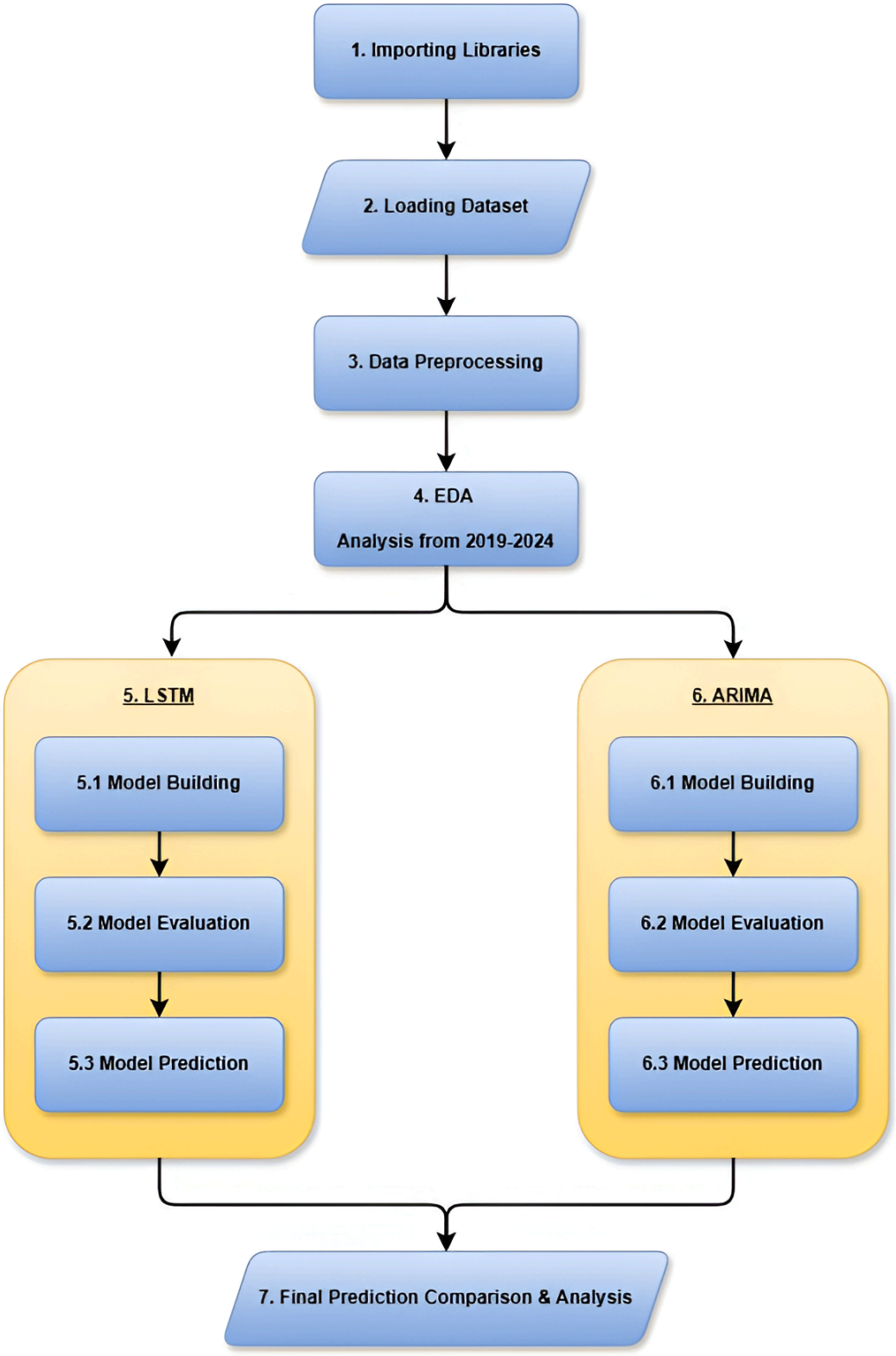
- Linearity assumption: ARIMA models assume that the relationship between your input variables and output variables are linear, which may or not be true in real cases such as financial time series.
- Stationarity (order of differencing)- The most common amendment needed is to take the sequence difference in order for it to be stationary. Although differencing makes the data stationary, some datasets follow non-linear, non-stationary behavior which ARIMA cannot capture well.
- Poor flexibility: ARIMA is not intrinsically well-designed to address non-linearity and canonical problems in other complex forms of time-series data.

3.3 Comparative Analysis: LSTM vs. ARIMA in Financial Forecasting

Even though ARIMA has been the go-to model for time-series prediction because of its simplicity and solid theoretical background, in recent years with machine learning blowing up LSTM networks have also gained much popularity. Specifically, LSTMs typically beat or match ARIMA in most time-series forecasting problems without seasonality which are known to include non-linear dependencies.

4. SYSTEM

System Architecture



System Introduction

This system architecture defines a framework for predictive analysis which predicts the performance of two different forecasting models namely Long Short-Term Memory (LSTM) and Autoregressive Integrated Moving Average (ARIMA). Our ultimate aim is to generate models that can be built and evaluated with 2019–2024 data, thereby allowing us to assess which model offers better predictive accuracy. The pipeline has stages starting from importing libraries and loading the dataset to preprocessing, EDA (Exploratory Data Analysis), model building evaluation and finally prediction comparison.

4.1 Dataset Instruction

For the purpose of this project, we created a time-series dataset using Ethereum daily closing prices from November 2017 until October 2024 based on data acquired via Yahoo Finance. This is the structure (columns) of our dataset.

- **Date:** Indicates the date of trading.
- **Open:** The price at which trading opened on this day;
- **High:** The highest price of the day.
- **Low:** It is the least price recorded for the day.
- **Close:** This is the price that brokers were willing to execute transactions by end of day.
- **Adj Close:** The adjusted closing price that considers stock splits and dividends.
- **Volume:** This is how much Ethereum was traded during the course of that day.

Table 1.1

	Date	Open	High	Low	Close	Adj Close	Volume
0	2017-11-09	308.644989	329.451996	307.056000	320.884003	320.884003	893249984
1	2017-11-10	320.670990	324.717987	294.541992	299.252991	299.252991	885985984
2	2017-11-11	298.585999	319.453003	298.191986	314.681000	314.681000	842300992
3	2017-11-12	314.690002	319.153015	298.513000	307.907990	307.907990	161347936
4	2017-11-13	307.024994	328.415009	307.024994	316.716003	316.716003	104188984

Table 1.2

	Date	Open	High	Low	Close	Adj Close	Volume
2487	2024-08-31	2525.859131	2532.389893	2493.705566	2513.393799	2513.393799	6646876013

The dataset's first few rows, beginning on **November 9, 2017**, are displayed in **Table 1.1**, and the final few rows, up to **September 4, 2024**, are displayed in **Table 1.2**.

This dataset ensures thorough coverage of Ethereum price changes during the chosen period with **2492 rows** and **7 columns** of complete records. The dataset underwent additional processing to make sure it was prepared for analysis and model creation, and all important features were cleaned and standardized to provide new insights.

Table 1.3

	Open	**High**	**Low**	**Close**	**Adj Close**	**Volume**
count	2492.000000	2492.000000	2492.000000	2492.000000	2492.000000	2.492000e+03
mean	1421.525431	1460.603427	1378.316347	1422.177985	1422.177985	1.241943e+10
std	1206.882663	1238.985663	1170.395195	1206.517780	1206.517780	9.994360e+09
min	84.279694	85.342743	82.829887	84.308296	84.308296	6.217330e

						+08
25%	247.831642	258.440597	242.204823	247.743229	247.743229	5.260002e+09
50%	1266.439209	1296.919067	1226.479249	1266.369019	1266.369019	1.015476e+10
75%	2226.635864	2280.923950	2150.996033	2228.061218	2228.061218	1.687439e+10
max	4810.071289	4891.704590	4718.039063	4812.087402	4812.087402	8.448291e+10

- Each group has its totals for the overall share amount (**2,492**), your average prediction, and a standard deviation indicating how much information there is.
- Very small or outlier values also illustrate how data is distributed in the dataset.
- The **25th**, **50th** and the **75th percentiles** add to this information showing what ranks different proportions of data fall below.

4.1.1 Data Loading

The data was loaded from a CSV file and assessed for missing points. There weren't any missing values, and all the columns were formatted correctly. Data exploration can be visualized using several charts, as shown in the next sections.

4.1.2 Data Preprocessing

Pre-processing of our dataset to train a model, our data will have to be prepared along the lines of:

- Data Parsing: we always do date parsing and format the column date to make it suitable for time series analysis.
- Normalization: normalizing **Open, High, Low, Close,** and **Adj Close Columns** to improve the predictions from the model, which was scaled between some ranges.

4.2 Exploratory Data Analysis(EDA)

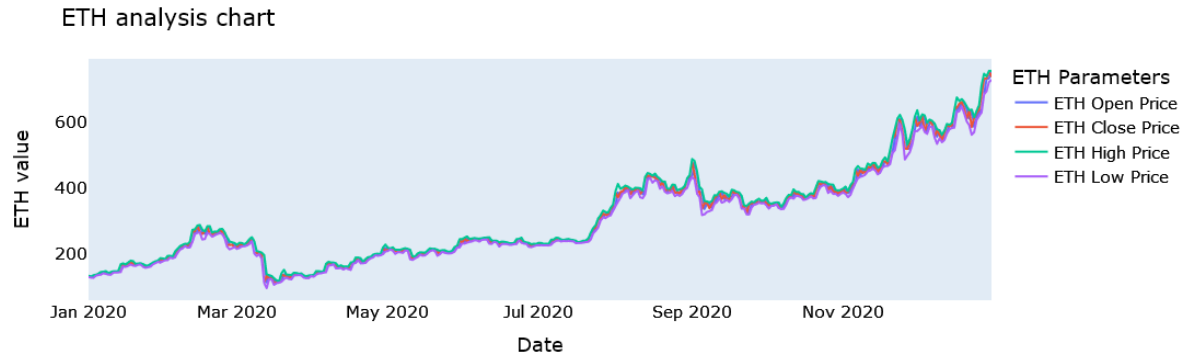
In this project, exploratory data analysis is also to understand the trends of Ethereum prices and check for patterns that can assist us later in performing more accurate predictions.

4.2.1 Year-wise Analysis (2019 to 2024)

Ethereum Price Data—2017–2024 To understand how the cryptocurrency behaved across time, we created year-wise movement visualizations of prices. The below charts are nothing but the open price. Yearly closing price of Ethereum from January 2019 to August 2024:

2019

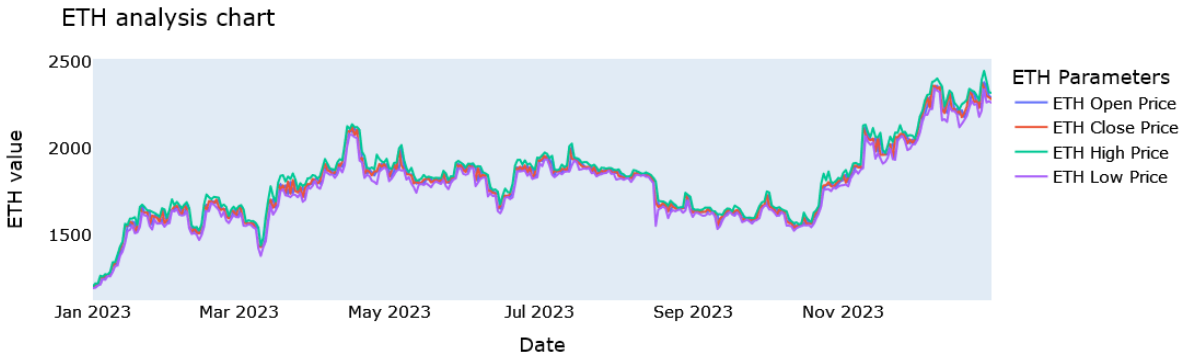
2020



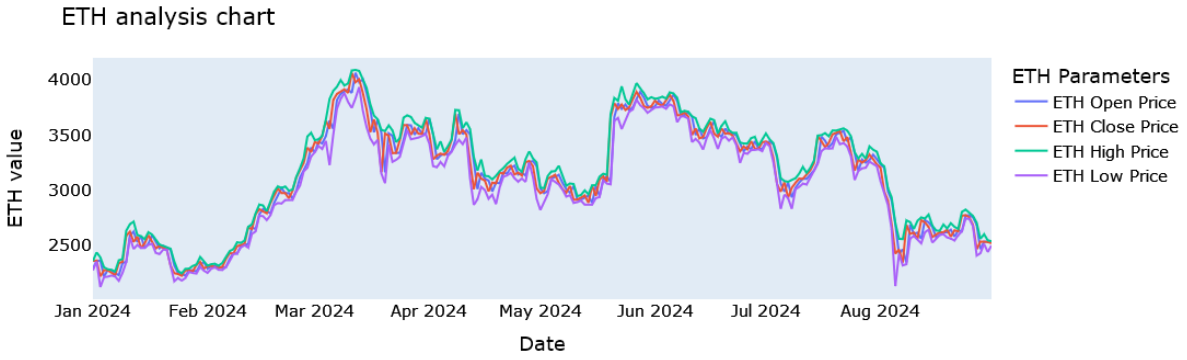
2021

2022

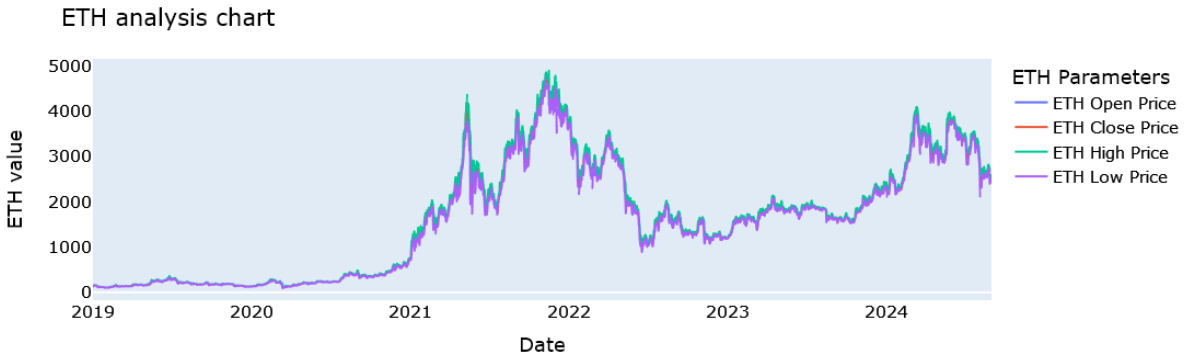
2023



2024



2019 to 2024



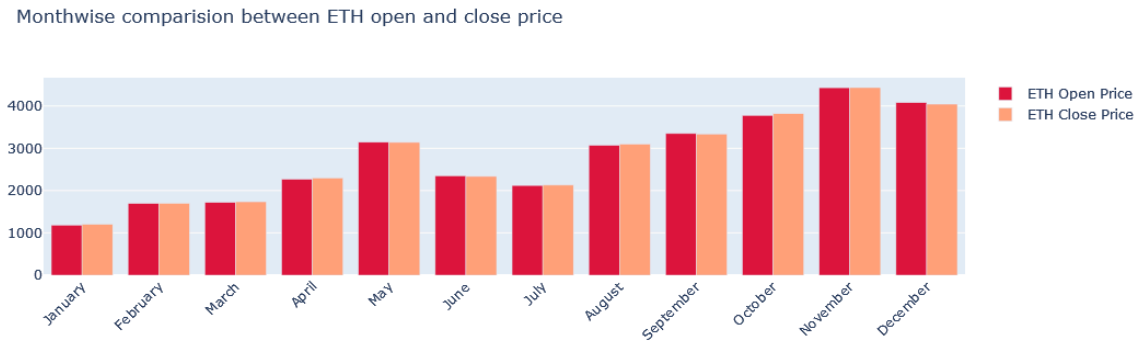
4.2.2 Monthly Trend Analysis (2019 to 2024)

Let us further calculate the monthly average open and closing prices to derive some insights on how well Ethereum performed over every month of the year. That helps cast a seasonal perspective, during which prices are normally more or less constant and at times increase vividly. Bar chart plot between the months of January 2019 to August 2024 showing the average open and close prices:

2019

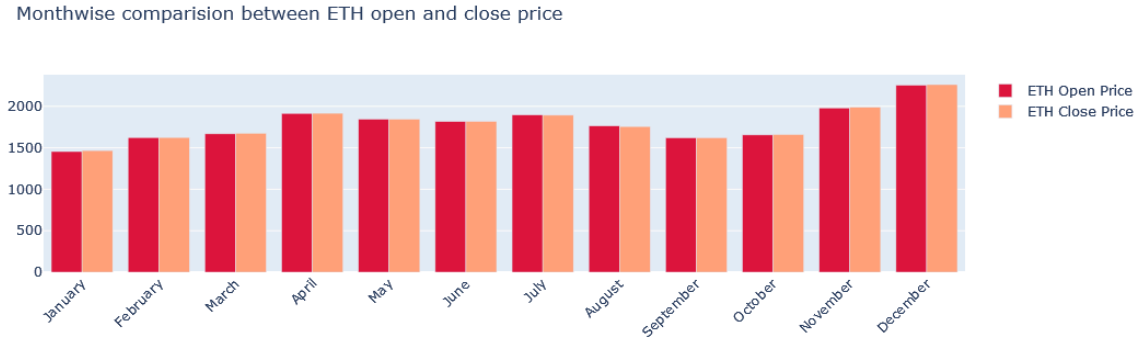
2020

2021



2022

2023

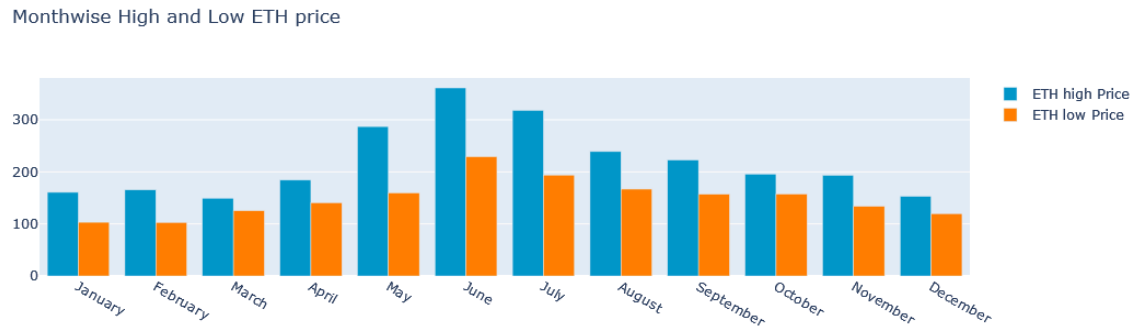


2024

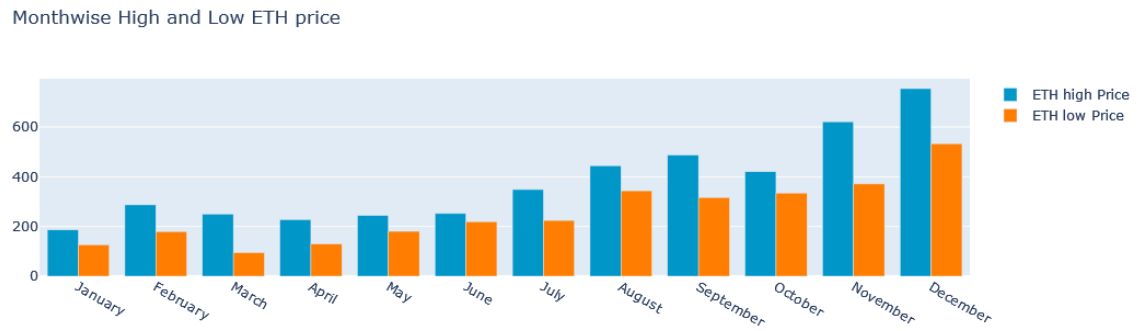
4.2.3 Price Range Analysis (High vs Low)

To figure out the volatility of Ethereum over time, monthly high and low prices were considered. Looking closely at the upper and lower prices each month can tell us when the market has been uncertain or in clear growth. Here is the Smith + Crown chart of Ethereum max and min prices by month from 2019 to June 2024.

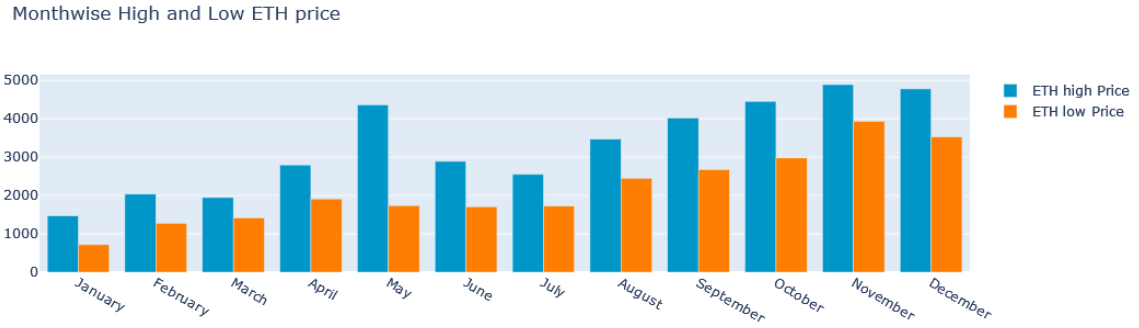
2019



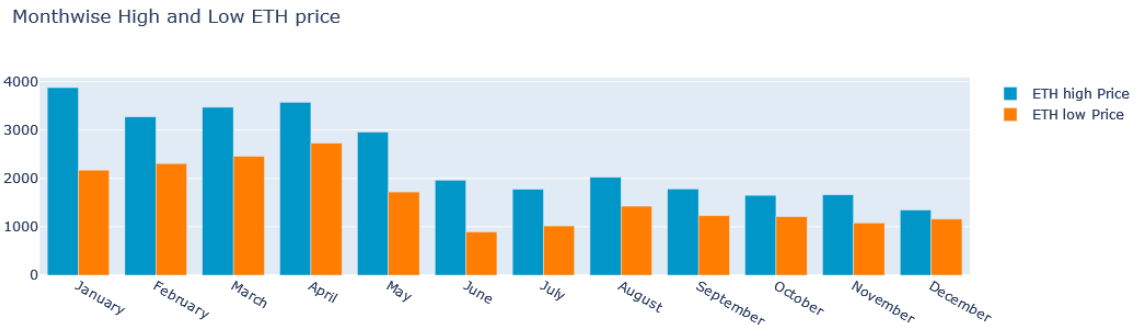
2020



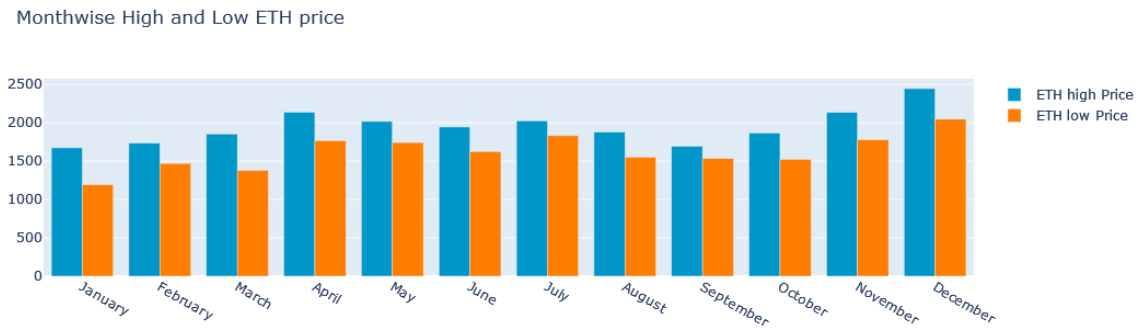
2021



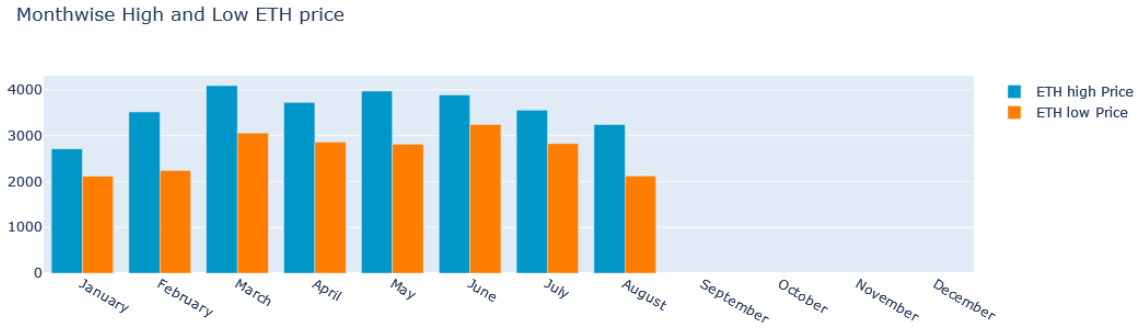
2022



2023



2024



4.2.4 Descriptive Statistics

Summary statistics, including mean, standard deviation, and the range of each variable, were calculated to understand the overall distribution of the data. There were no missing values in the dataset, ensuring that all columns are complete for further analysis.

4.3 Model

Now let's dive into the actual models that were built and trained, we'll cover both LSTM as well as ARIMA model. These models have been selected because it is common knowledge that they perform well on scalar time series forecasting, but for different reasons.

Long Short-Term Memory (LSTM) is a deep learning model that can learn long-term dependencies and non-linear patterns. ARIMA (AutoRegressive Integrated Moving Average) is the most widely used statistical forecasting algorithm to forecast time series data, which only captures linear trends.

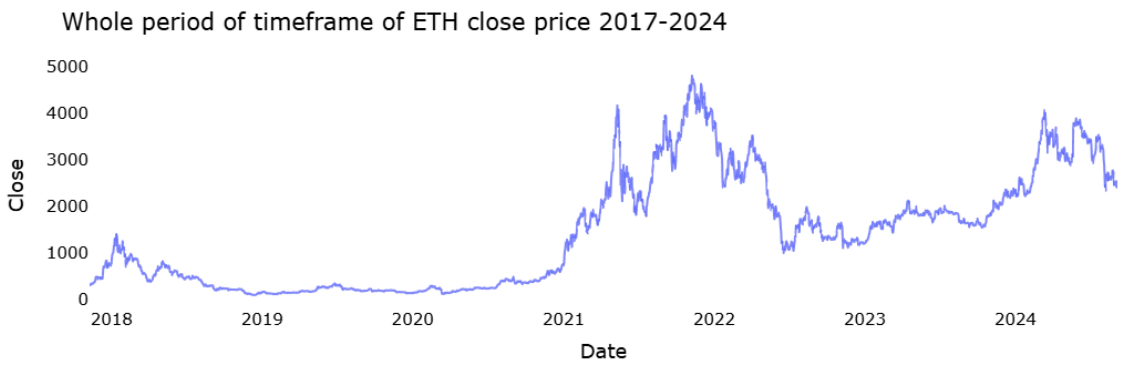
4.3.1. LSTM Model Building

LSTM (Long Short-Term Memory) model is a Recurrent Neural Network, and it can be used in problems where sequence related information matters. Not only does this have the convenient properties of a slow information-retaining architecture for reference over very long sequences—which would be good news as something we might want to use, e.g., in conjunction with monetary prediction (a.k.a. level 2)—but it also enables us to see if weights learned on all kinds of other interesting data sets that were recorded alongside various cryptocurrencies recording their prices can help predict cryptocurrency price movements any better than doing nothing at all!

Building up the LSTM model in the following steps:

1. Data Preparation

The dataset was prepared using Ethereum price data (for 2017 to present and for all future dates) with the closing columns as a target. The data is divided into 60–40 weights for training and testing sets. 60% (1495) of the rows were used for training, and 40% (997) of them were tested out on each other since data has a total size equal to 2492.



2. Normalization

The data was then normalized (scaled between 0 and 1) before it entered the LSTM model using **MinMaxScaler** from **sklearn** library. It is not only necessary for LSTM to perform well, but it also eliminates the issue of huge range differences in input data.

3. Data Transformation

The time-series data was transformed to a sequential form. In particular, the model uses a sliding window of 15 days closing prices for each data point to predict the next day's closing price.The input has been formatted into a 3D array with dimensions '(samples, time steps, features)' so as to be utilized by the LSTM layer.

4. LSTM Architecture:

Develop the LSTM model with **Keras Sequential API**. The model first layer was an LSTM layer with 10 units, and the activation function used here is ReLU. The ReLU function also helps to solve the vanishing gradient problem that we usually run into when using RNN models.

A dense layer of 1 unit was added after this LSTM that produces the predicted close price. For our optimization, we used **"Mean Squared Error (MSE)"** as the loss function and selected **“Adam”** because it is good for sparse gradients.

5. Training:

We trained 200 epochs with 32 batch size.However, the training and validation loss were tracked during the training to make sure that it did not overfit.

Model: "sequential"

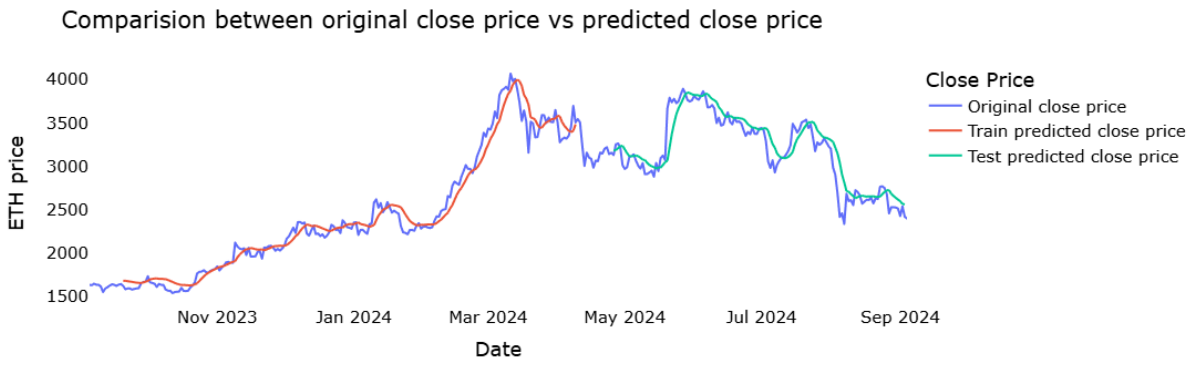
Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 10)	480
dense (Dense)	(None, 1)	11

Total params: 1,475 (5.77 KB)

Trainable params: 491 (1.92 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 984 (3.85 KB)



Actual vs. Predicted Prices for LSTM Model

4.3.2 ARIMA Model Building

ARIMA (AutoRegressive Integrated Moving Average) is a statistical method for analyzing and forecasting time series data. It fits a time series to itself (autoregression part), forecasts errors based on previous predictions (moving average), and differences the series in order to make it stationary.

Procedure followed to form the ARIMA model:

1. Parameter Identification:

The first part of implementing an ARIMA model is to determine p, d, and q using the appropriate parameter selection method.

- p (autocorrelation term)—the number of lag observations to use as inputs.
- d (differencing term) is used to know how many times we need differentiating of data to make it stationary.
- q (moving average term)—the size of the moving window

The following parameters have been selected based on the **AutoCorrelation Function (ACF)** and **Partial AutoCorrelation Function (PACF)** plots that help to understand dependency structure in data.

2. Differencing:

Ethereum price data showed non-stationarity, as the mean and variance of prices changed over time. This applies first-order differencing, d = 1, to make the data stationary as it takes in differences (differences between consecutive observations) and removes trends and seasonality.

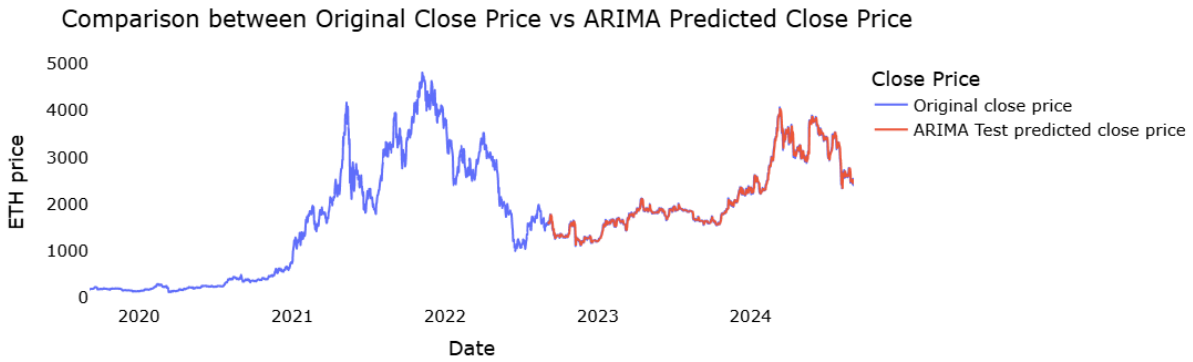
3. ARIMA Model Construction:

We created an ARIMA model using the 'statsmodels' library in Python with a class function, enabling automatic integration of AR (auto-regressive), I (Integrated [diff]), and MA (moving average) components.

Using the Ethereum data, we fit our model and then forecasted future prices based on historical closing prices.

4. Training:

We trained the model with the Ethereum close price data and saved that in weights. The emphasis was on forecasting the prices using these best parameters obtained from the analysis.



The actual and predicted Ethereum prices using the ARIMA model.

4.4 Evaluation

It's important to know which model provides a better prediction for future Ethereum price. Both models were evaluated by using various standard performance measures. In this section, we will test the performance of both LSTM and ARIMA models based on metrics that are widely used in time-series forecasting.

MSE (Mean Squared Error): It calculates the average of the square differences between predicted and actual values. Because they are squared, an error number that is way off from the function has a huge effect on what this total will be and so MSE really punishes those big errors.

MSE is nice for the reason that much larger mistakes are punished additionally plus it reacts to outliers.One of the reasons MAE is a good quality metric to use because it results in an interpretable value using the same units as original data and nothing needs to be squared.

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \widehat{y_i})^2$$

Where,

y_i = actual value at point i

$\widehat{y_i}$ = predicted value at point i

n = total number of data points

.

MAE (Mean Absolute Error): It simply measures the average of the absolute difference between predicted and actual values. MAE is fairly intuitive and more robust to outliers than MSE since it does not “square” the errors.

MAE does not get heavily affected by these outliers and thus is a better metric when high deviations are less significant compared to MSE.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y_i}\right|$$

RMSE (Root Means Square Error): The resulting MSE value is square-rooted to arrive at Root Mean Squared Error (RMSE). It does measure the average error but returns a value that is in the same units as original data (unlike MSE). Like MSE, RMSE also exaggerates the error for larger errors because of squaring.

The lower the RMSE, the more accurate our model is (thus closer to 0).

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

Explained Variance Score: It determines how well the variance of the target variable is explained by our model. What it indicates is that the model predictions explain some variance of the target variable. The higher the Explained Variance Score indicates the data variances are more accurately captured by the model. But unlike the R2 score, it does not penalize overfitting as severely, so if the model is overly complex, a high score could still be deceptive.

$$Explained\ Variance = 1 - \frac{var(y_i - \widehat{y_i})}{var(y_i)}$$

R² Score: R-squared is a statistical measure of how close the data are to the fitted regression line. It compares the variance of a model's error to the variance in proportion with predicting actual value. It tells us how well our model estimates errors. R value closer to 1 means the model can explain more of the variability in our target data and vice versa. An R^2 such as 0.75 is given to a model that can explain about 75% of the variance in reality data(dataset).

$$R^2 = 1 - \frac{\Sigma(y_i - \widehat{y_i})^2}{\Sigma(y_i - \overline{y_i})^2}$$

Where,

$\overline{y_i}$ = mean of the actual values

$\Sigma(y_i - \widehat{y}_i)^2$ = sum of squared residuals (errors)

$\Sigma(y_i - \overline{y}_i)^2$ = total sum of squares (variance of the actual data)

4.4.1 LSTM Evaluation

The aforementioned LSTM model was tested on the test data, and its output results were predicted using Mean Squared Error (MSE) and Mean Absolute Error (MAE). The validation loss of the model decreased over 200 epochs demonstrating that it actually learned from the dataset.

Mean Squared Error: The test MSE after training was so low, that we can consider the model is able to get close enough to the true values.

For this, other than the RMSE we also calculated MAE and found that MAE is significantly less too which was a good indicator.

4.4.2 ARIMA Evaluation

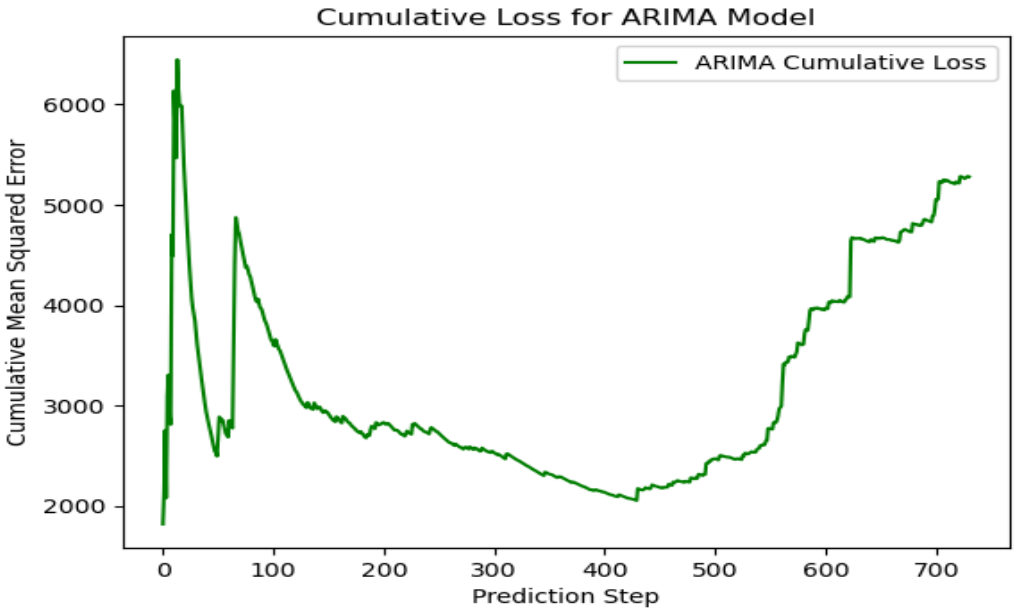
The performance of the ARIMA model was also measured using MSE and MAE. Although ARIMA is most likely designed to predict linear trends and stationary data, it performed lousy in predicting Ethereum prices as these are massively volatile — while LSTM does better when it comes to timeseries forecasting.

MSE: The ARIMA model has a worse MSE than LSTM, which shows it is not able to learn effectively when predictions are largely deviating between the actual price.

MAE: The ARIMA model did slightly worse (i.e., higher MAE) in terms of daily prediction error.

Although it has served us well in the past, especially for this kind of data and is definitely more interpretable than all that retailer nonsense I spew out here sometimes ARIMA's linearness again depended so much less on explaining the intricacies OR (and even if) there are those interactions present within non-linear markets like cryptoland! But it did a good job at predicting long-term trends — motion in price that draws its path smoothly.

Additional parts of modeling & evaluation for LSTM and ARIMA, giving a complete drill down.



5. IMPLEMENTATION & EXPERIMENT

The ARIMA and LSTM models were implemented in Python using TensorFlow/Keras for LSTM and the statsmodels package for ARIMA. Standard procedures were followed throughout implementation, such as cross-validation and hyperparameter tweaking, to optimize model performance.

ARIMA Model Implementation

The statsmodels.tsa.arima.model.ARIMA class was utilized to implement the ARIMA model.Grid search was used to find the ideal values for p(AR), d(differencing), and q(MA), which were then assessed using metrics such as AIC and BIC.These values were then used to train the model.

LSTM Model Implementation

TensorFlow/Keras was utilized in the construction and training of the LSTM model.Using grid search or other optimization approaches, the network architecture optimized several LSTM layers, as well as other hyperparameters including the number of

units, layers, dropout rates, and learning rates.After preprocessing Ethereum price data, the model was assembled using the Adam optimizer.

5.1 Prediction

5.1.1 LSTM Prediction

The 30-day out-of-sample predictions for Ethereum prices using the trained model from the LSTM Model were finally done on test data. Below are the steps to predict:

Data Preparation

The next is it was used as the input to forecast 30 days in the future on the last part of the test data set. The input got reshaped to the LSTM model format, with sequences of n_steps size regarding number_of_days, which correspond to how many days one infers for prediction.

Prediction Loop

The loop to predict the values for the next 30 days. At each iteration, the final step is where we have to add the predicted value and remove the oldest point in the input sequence.This is what we predicted from our LSTM model which was trained.The loop went on until 30 predictions were produced. Thus, it could predict further into the future than just a single iteration by supplying its previous predictions as inputs to subsequent iterations.

Inverse Transformation

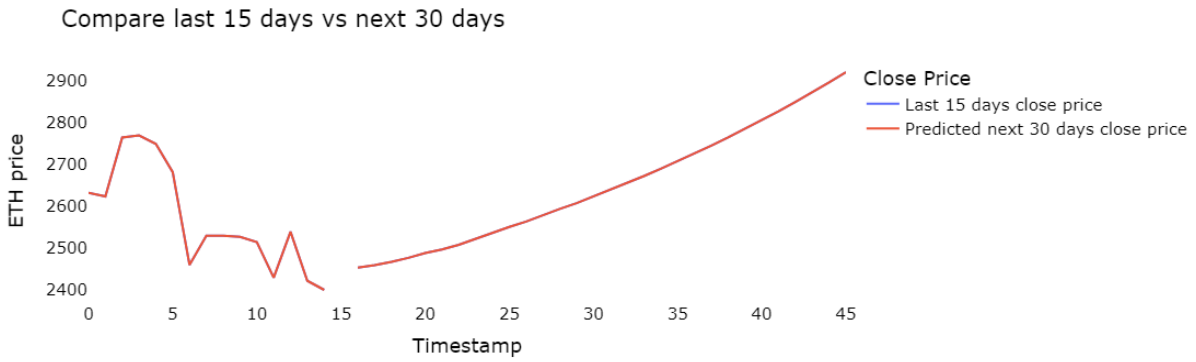
After making the predictions, we had to transform our predicted values from their scalar versions using this: inverse_transform() method. This step was required because the input data had been normalized during preprocessing.

Visualization

Below are the two plots for plotted to show predictions:

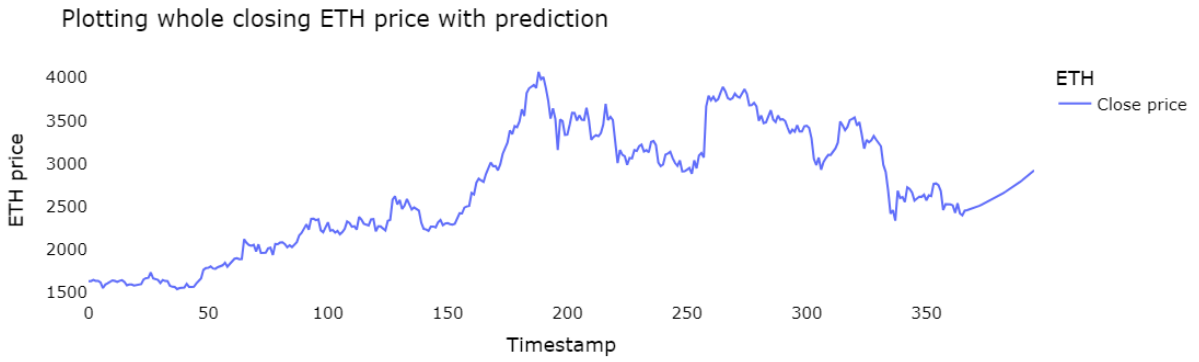
Last 15 Days vs. Next 30 Days

This graph shows the actual closing Ethereum price last 15 days and going forward what predicted prices are forecasted by our algorithm. This comparison gives an idea of what to expect from the model for extrapolating trends going forward with recent historical data.



Predictions Over 30 Days Full Closing Price

This graph shows the historical price changes of a closing and 30-day prediction period. This includes a look at the price history of Ethereum as well as giving you an idea of where prices are going in the future according to our model.



Results

The forecasted Ethereum prices for the next 30 days seem to be smooth as binding with overall price direction is one of the model's strengths. Here is a visualization of how the LSTM model continues with the price trend once some data has been provided**.**

5.1.2 ARIMA Prediction

ARIMA Model Prediction

ARIMA model was trained based on historical Ethereum price data to predict the next 30 days of Ethereum pricing.

Prediction Process:

Inputs were Initialized

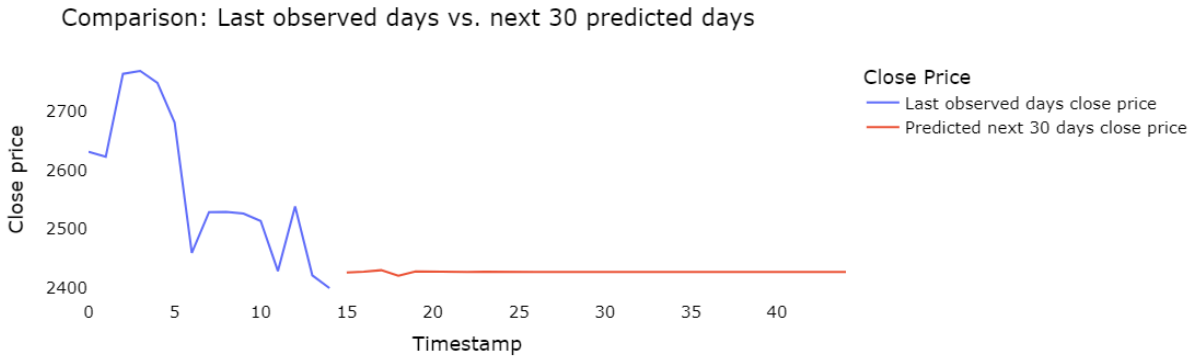
The last observation results from the test dataset which is most recent for 15 days served as inputs to initiate predicting.

Generating Predictions

The ARIMA model makes predictions on the closing prices of the next 30 days At every prediction step, add the predicted value instead of the last recorded input to input for the next step making a sliding window over most recent time steps.

Predictive Visualization

Comparing the Last Observed Days to the Next 30 Predicted Days

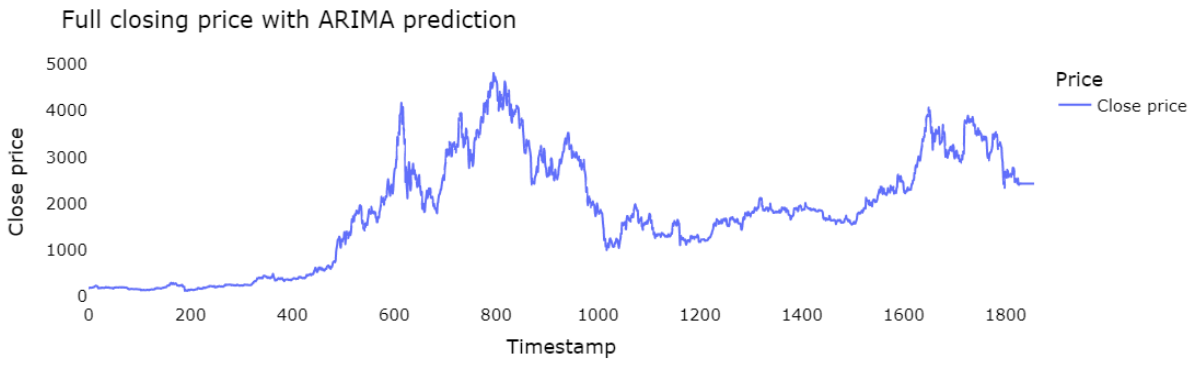


This graph shows how the ARIMA model predicts the price will change over the next 30 days.

Full Closing Price Prediction Based on ARIMA

Another graph shows the full range of historical closing prices, supplemented by ARIMA forecasts for the following 30 days.

Full closing Price with ARIMA Prediction



This graph combines the entire dataset with the predicted values to provide a comprehensive view of the Ethereum price trend.

Overall, according to the ARIMA model's forecasts, Ethereum's closing price will remain relatively constant over the next 30 days, implying minimal variation based on previous data.

5.2 Discussion

Comparing LSTM and ARIMA Models for Predicting the Ethereum Price

We are going to compare the LSTM model to the ARIMA model for predicting Ethereum prices. Analyzing both models by using a range of criteria to examine accuracy and overall efficacy.

5.2.1. Overview of Model Performance

LSTM Model

Loss Metrics:

Training Loss

The training loss shows a steady decline, suggesting that the model is learning effectively from the training data. This loss function reflects how closely the model's predictions align with the actual values during training.

Validation Loss

Similarly, the validation loss reduces, but more slowly than the training loss. This suggests that the model is well-suited to new, previously unseen data, albeit there is still room for improvement in terms of overfitting.

Prediction Metrics:

Root Mean Squared Error (RMSE)

The LSTM model had an RMSE of 81.74 for training data and 129.79 for test data. Lower RMSE values often indicate greater predictive ability, with training data predictions much more accurate than test data predictions.

Mean Squared Error (MSE)

The training MSE stood at 6682.08, while the test MSE was higher at 16844.74, indicating that the model struggles more when trying to generalize from training to new data.

Mean Absolute Error (MAE)

For the LSTM model, the MAE was 58.65 for training and 90.47 for test data, representing the average size of the prediction errors.

Explained Variance Score

The training score was 0.986, and the test score was 0.910, both of which show that the model captures a significant amount of variance in the data. The R^2 score for training and **test** data was 0.986 and 0.903, respectively, showing that the model's predictions are well-aligned with actual values.

Mean Gamma Deviance (MGD) and Mean Poisson Deviance (MPD)

The MGD and MPD values indicate that the model's predictions are reasonably close to the actual values, with lower values indicating better performance.

ARIMA Model

Loss Metrics:

Cumulative Loss

The ARIMA model's cumulative loss has shown a gradual increase over time. While this isn't directly comparable to the loss metrics of the LSTM model, it does imply that the ARIMA model consistently experiences rising prediction errors.

Prediction Metrics:

RMSE

The ARIMA model achieved a lower RMSE of 72.66, which outperforms the LSTM's test RMSE, suggesting it is more accurate in its predictions.

MSE

With an MSE of 5279.95, ARIMA's performance is also better than the LSTM's test MSE, implying that its predictions are closer to the actual values.

MAE

The MAE for ARIMA at 46.32 is lower than that of LSTM, reinforcing its superior accuracy in terms of average error.

Explained Variance Score

ARIMA scored 0.991, showcasing its outstanding ability to capture data variance, outpacing LSTM in this aspect.

R^2 Score

The R^2 score for ARIMA at 0.991 is slightly higher than that of LSTM, suggesting it provides a better fit for the actual values.

MGD and MPD

ARIMA's slightly elevated MGD and reduced MPD when compared to LSTM indicate that its predictions are overall more accurate.

5.2.2. Model Comparison and Insights

Accuracy and Generalization

Generally, ARIMA outshines LSTM in terms of RMSE, MSE, and MAE on the test data. This suggests that ARIMA is particularly well-suited for capturing trends in Ethereum price data when applied to new, unseen datasets. - While LSTM shows robust performance during training, it tends to produce higher errors on the test data, which might be attributed to overfitting or a lack of sufficient training data.

Variance and R^2 Scores

Both models exhibit high explained variance and R^2 scores, indicating effective predictive capabilities regarding Ethereum prices. However, ARIMA edges out LSTM in these metrics, showing a better capacity to capture the underlying patterns in the data.

Loss Metrics and Deviance

The cumulative loss for ARIMA shows stable performance over time, while LSTM displays a sharper decrease in loss throughout training. The lower MGD and higher MPD for ARIMA suggest its predictions align more closely with actual values compared to LSTM.

Visualization

When we graph the actual versus predicted prices for both models, it becomes apparent that ARIMA's predictions track the real price movements more closely, whereas LSTM, while useful, tends to deviate slightly more.

A comparison table for evaluation metrics of the LSTM and ARIMA models

Metric	LSTM Model	ARIMA Model
Training Loss	Decreasing trend	Not directly comparable
Validation Loss	Decreasing but slower than training	Not directly comparable
Test RMSE	129.79	72.66
Test MSE	16844.74	5279.95
Test MAE	90.47	46.32
Train Explained Variance Score	0.986	N/A
Test Explained Variance Score	0.910	0.991
Train R^2 Score	0.986	N/A
Test R^2 Score	0.903	0.991
Train MGD	0.00093	N/A
Test MGD	0.00181	0.00098
Train MPD	2.424	N/A
Test MPD	5.479	2.132

5.2.3. Summary

Both models have their own strengths and weaknesses. The ARIMA model clearly stands out in terms of accuracy and reliability for predicting Ethereum prices, backed by its

lower RMSE, MSE, and MAE scores, along with higher explained variance and R^2 metrics. Meanwhile, LSTM, although effective, may need further fine-tuning and additional data to match ARIMA's performance. Ultimately, the choice between these models should take into account not only prediction accuracy but also considerations like computational efficiency and ease of implementation. For scenarios requiring real-time predictions with high accuracy, ARIMA is likely the better option. However, for more complex patterns and sequences where LSTM can thrive, it remains an invaluable tool. Future experiments and refinements could further enhance the predictive capabilities of both models.

5.3 Conclusion

In this study, we examined how ARIMA and LSTM models compare to one another in terms of forecasting Ethereum prices. Our data imply that LSTM networks outperform in terms of predicting accuracy. They excel at detecting complicated, non-linear patterns that are common in bitcoin markets. ARIMA models, on the other hand, are considerably simpler and easier to grasp, but they struggle to deal with the volatility and complex linkages observed in Ethereum price fluctuations. There are various possibilities to expand on this study in the future.

1. Incorporating Additional Features: Input features can be added to increase model performance such as Technical indicators, macroeconomic factors, and sentiment data.
2. Exploring Hybrid Models: Capitalize each methodology’s strengths to combine statistical and machine learning methods.
3. Optimizing Models: To improve the performance of the LSTM model, we need to use advanced hyperparameter tuning techniques and experiment with various neural network designs.
4. Generalization to Other Cryptocurrencies: Across other digital assets, we need to apply the models to other cryptocurrencies to determine generalizability and robustness.
5. Real-Time Prediction Systems: For Trading and Investing, real-time forecasting systems can be created to provide accurate forecasts.

Overall, this study emphasizes the learning of the LSTM and Arima Models and their networks and how to advance the financial time-series forecasting and provide significant insights to cryptocurrency investors and analysts.

6. REFERENCES

Box, G. E. P., & Jenkins, G. M. (1970). *Time Series Analysis: Forecasting and Control*. Holden-Day.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

Zhang, G., Patuwo, B. E., & Hu, M. Y. (1998). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35-62.

Chen, Y., Liu, B., & Wang, H. (2020). Cryptocurrency price prediction using LSTM networks. *Journal of Financial Data Science*, 2(3), 45-58.