Group Project 3 - Cryptocurrencies

William Swann,
Md Aaraf Azad,
Chris He,
Colar Wang,
Dylan Webster,
Mehul Datta

March 2025

Contents

Abstract

The volatility of cryptocurrencies is a major challenge that makes long-term price prediction unreliable, especially over a long time period. This is due to the fact that the volatility of these cryptocurrencies has a significant effect on the performance of a portfolio. In this report, we present the development of the portfolio program, including the models factored in, and assumptions made. The results of these methods are discussed, and the reliability of these predictions is compared to real data. Multiple models were considered and compared, Long Short-Term Memory (LSTM), and two Stochastic Differential Equation (SDE)-motivated models, simple SDE and Ornstein–Uhlenbeck (OU) Process. The LSTM model outperforms the SDE-motivated models extensively, mostly owing to its natural tendency to correct for over-diffusion due to cells installed in the neural network algorithm, which can ascertain how much of data fed into the model to forget, input, and then reintegrate back into the algorithm again (the use of Forget Gate, Input Gate, and Output Gate in the LSTM cell). The model tends to invest a majority into stable currencies, which hold their value while deriving all the returns by investing a smaller percentage into high-variance currencies like Bitcoin and Ethereum, the high-return coins.

1 Introduction

In the modern age, almost all financial transactions can be simplified down to alterations on a spreadsheet. Many argue that this is insecure, as it is prone to security faults such as hacking. Cryptocurrencies are based on blockchain, which is a new method of data storage, divided into a chain of individual blocks that each represent a single transaction. These data 'blocks' normally consist of 3 pieces of information:

- Amount - The value of the transaction
- Address - A unique identifier for the specific block
- Previous Address - The unique identifier to the previous block

If the block is edited, the Address changes. This makes it easy to identify discrepancies, like data alterations, as the "Previous Address" of the next block will no longer match the "Address" on the current block. The second strength in blockchain's security over traditional currencies is that anyone can have a copy. This helps prevent errors, as it is easy to see that they have occurred, since a significant portion of the blockchain copies will have the error, whereas a hack could feasibly change significantly fewer.

Cryptocurrencies have their own faults. One of which is that many would argue that due to the cost of resources expended to check the blockchain, cryptocurrencies are too expensive and environmentally unfriendly (Reiff, 2021)[1]. Another downside of cryptocurrencies is that they do not have tangible comparisons. Despite the fact that the modern pound sterling can buy barely a gram of sterling silver in the current economy, the psychological links guarantee that currency has value, something cryptos lack. This is one of the main reasons that they are so volatile.

Cryptocurrencies have grown unsustainably in the last 10 years, reaching their highest media attention in 2017 with YouTubers, like Jake Paul, advertising them. The most popular cryptocurrency is BitCoin, a "Proof-of-Work"[1] currency, which peaked at £89, 327.23 on the 20th January 2025 (Coinbase, 2025)[3].

[1]Proof-of-work blockchains are secured and verified by virtual miners around the world racing to be the first to solve a mathematical puzzle. (Coinbase, 2020)[2]

As mathematics students who have won £12.5M in the lottery, £10M will be invested in a well-diversified crypto portfolio. This portfolio should focus on long-term returns (10 years) and take into account different risk preferences for the individual. Portfolio options are scarcely available online, with most only offering suggestions, such as Clarity In Crypto's (Clarity in Crypto, 2021)[4] moderately risky portfolio:

Cryptocurrency	**Percentage**
Bitcoin	35%
Ethereum	35%
Cardano	10%
Polkadot	10%
Solana	10%

Table 1: Cryptocurrency Portfolio Distribution

Recognising this, the best solution to invest in a mathematically diverse portfolio is to develop one using available data and rigorous testing.

The first section of the report provides an overview of the data collection and the limitations faced. This is followed by an outline of the methods used to transform the raw data into a suitable model to predict future trends. The Results section discusses the findings of these methods and assesses the reliability of these predictions compared to observed data. Additionally, the development of the portfolio program is explained, including the models factored in, and assumptions made. Finally, a conclusion to the project is described and explained.

2 Data Collection

To build the portfolio, the 50 largest cryptocurrencies (by total market capitalisation) were examined, and their closing prices during the 2022-2024 period were observed. Different selections could be made, such as rapidly growing currencies, however due to the decentralised nature of cryptocurrency (Scott Likens, 2023)[5], there is no legislation preventing price manipulation, such as pump and dump schemes (La Morgia et al., 2024)[6]. These are blockchain events driven by notorious intentions and are therefore not predictable. Schemes like this occur spontaneously, and due to the direct influence on prices, they cannot be modelled effectively. To counter this, we made sure to only invest in the top 50 well established currencies.

From these 50 currencies, we carefully measured each currency, monitoring fluctuations, volatility, and correlations. After concluding that 7 were too volatile to model effectively, as the predictions were too uncertain, often having 95% confidence intervals of 3 times the current price, we removed them. This reduced our data set to 43.

To test our methods, we first used a very simple linear model to predict future prices, taking the gradient over the last 365 days as the expected return. We found that regularly many of these currencies did not end up being utilised in our model. After some exploratory analysis into correlational and covariate matrices, we found that many cryptocurrencies had high correlations with others, meaning that, for example, Bitcoin and Doge-coin have almost identical pricing. These currencies are virtually useless in portfolio diversification. To fix this, the data set was optimised by creating a program that identified any currency with a correlation of 0.75 or higher, calculated each currency's respective volatility, and removed the more volatile one. This reduced our data set to only 17 cryptocurrencies.

3 Models

3.1 Assumptions

As this study is directed towards long-term investing rather than daily price changes, it assumes that the price conditions of cryptocurrency follow the weak market hypothesis (Yi et al., 2023)[7], and are dependent on a daily average model. This method makes the model treat prices as discrete, and not continuous movements. Portfolio construction is based on estimates of historical returns, volatility, and covariance to determine risk. Filtering out highly volatile assets that could not be effectively modelled, only well-established cryptocurrencies were selected. The covariance matrix used for risk estimation is based on a 360-day rolling period that balances short-term and long-term trends. whereas portfolio optimisation is based on Markowitz's efficient market frontier (Siddiqui, 2022)[8]. It does not incorporate a risk-free asset because cryptocurrency markets are inherently risky due to their decentralised nature[2], nor does it take intraday volatility into account, as it was based only on average daily prices.

3.2 Parameter Considerations

The cryptocurrency prices are extraordinarily volatile and possess a notoriously complicated relationship with conventional economic factors (Polizu et al., 2023)[26]. Therefore, the primary focus was intensified on the historical price fluctuations. This was decided based on the random walk nature of the price movements, which suggested very high stochasticity, and there was a reasonable suspicion that the price factor possessed a Markov Property (where the future trajectory depends only upon the present position, if they are first order, and on finite number of past steps if they are higher-order, essentially denoting a dependency to some extent) (Arajo et al., 2023)[23]. The type of model that will be able to capture the extraordinary volatility would have to be models that can learn from the historical trends rapidly, adapt, and predict accordingly, which falls within the bounds of machine learning capabilities. Therefore, the data set would be focused on training data, and thus, price would be the primary parameter to be considered.

3.3 Model Considerations

The data points observed had a temporal component, leading to the consideration of time-series models to capture the trend of price changes with respect to time. Therefore, 3 models were considered:

- Long Short Term Memory (LSTM)
- Stochastic Differential Equation (SDE) motivated model
- Ornstein–Uhlenbeck (OU Processes), which was also motivated by SDE processes, with some parameter modifications considered.

3.3.1 Long Short-Term Memory

Long Short Term Memory (LSTM) is a Recurrent Neural Network (RNN) Model (Amidi et al., 2019)[19] that can capture non-linear and complex dependencies in time series data. The input

[2]even USD-coin, a crypto supposedly mapped to the price fluctuations of the USD is extremely volatile

data was fed in sequential form because the data collected was an average for one day and, thereby, discrete. The values were first normalised by the **Batch Normalisation** process (Bjorck et al., 2018)[35], condensing the values in the range $(0, 1)$, and the model performed as follows:

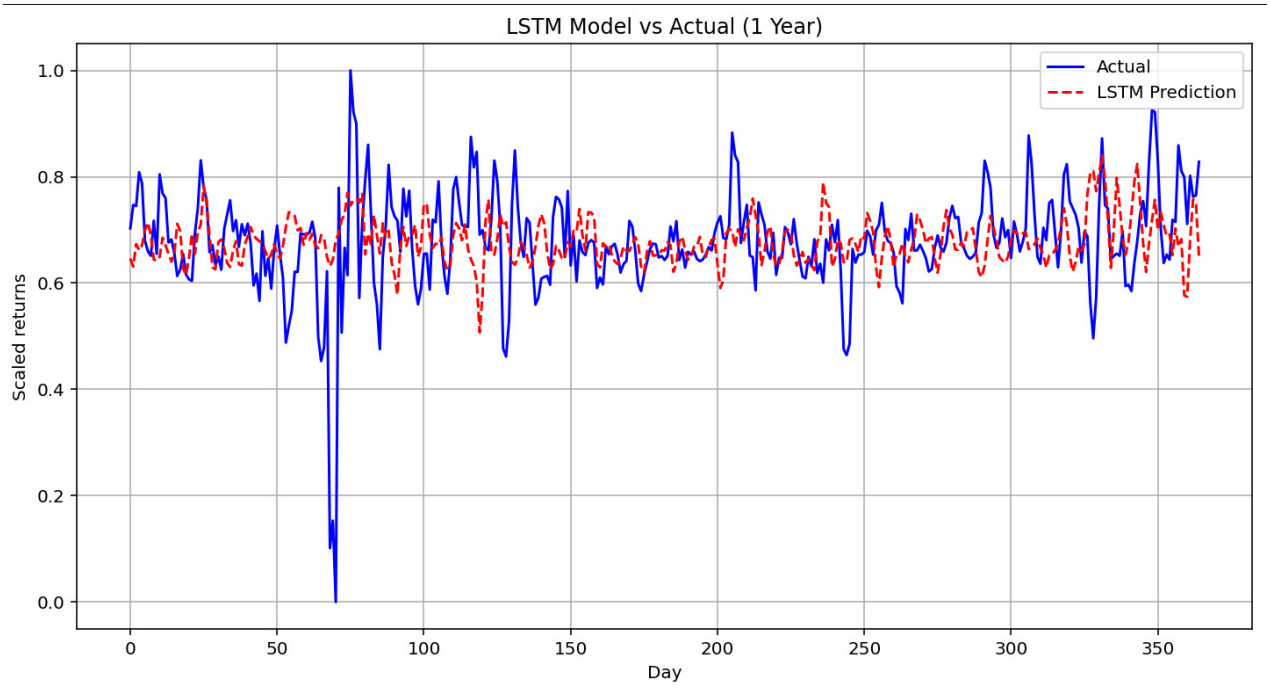

Figure 1: LSTM Predictions vs. Actual

From the graph, it can be observed that the model starts performing better as time increases, better able to mesh with the ebb and flow of cryptocurrencies. Toward the tail of the data, the predictions better mimic trends displayed by the price fluctuations, relative to the initial stages, when the model seems to miss the plunge down. Therefore, other models were considered and explored to better capture the overall trend of available data.

3.3.2 Stochastic Differential Equation

Cryptocurrency price movements emulate a random walk on a continuous time spectrum (Kaur, 2024)[10]. This is mainly due to the price movements going only in one of two directions, up and down, and therefore, Stochastic Differential Equation (SDE) models were considered. The SDE Model considers time as a continuous variable using the Wiener Process, which is a stochastic (random) process on a continuous time parameter. Two versions of SDE-motivated models were considered, the simple SDE model, with a deterministic component (the drift factor), and the Ornstein–Uhlenbeck Process, which makes the drift factor stochastic. This allows the model to incorporate both drift, and diffusion (term denoting stochasticity), demonstrated in the following equation:

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \tag{1}$$

Where X_t denotes the state variable, varying with time, t, μ refers to the long-term mean to which the process converges (also known as the drift element), σ represents the stochastic fluctuations (diffusion element), and W_t denotes the Wiener process that converges to a geometric

Brownian Motion. The performance for the simple SDE model with a deterministic component is shown:

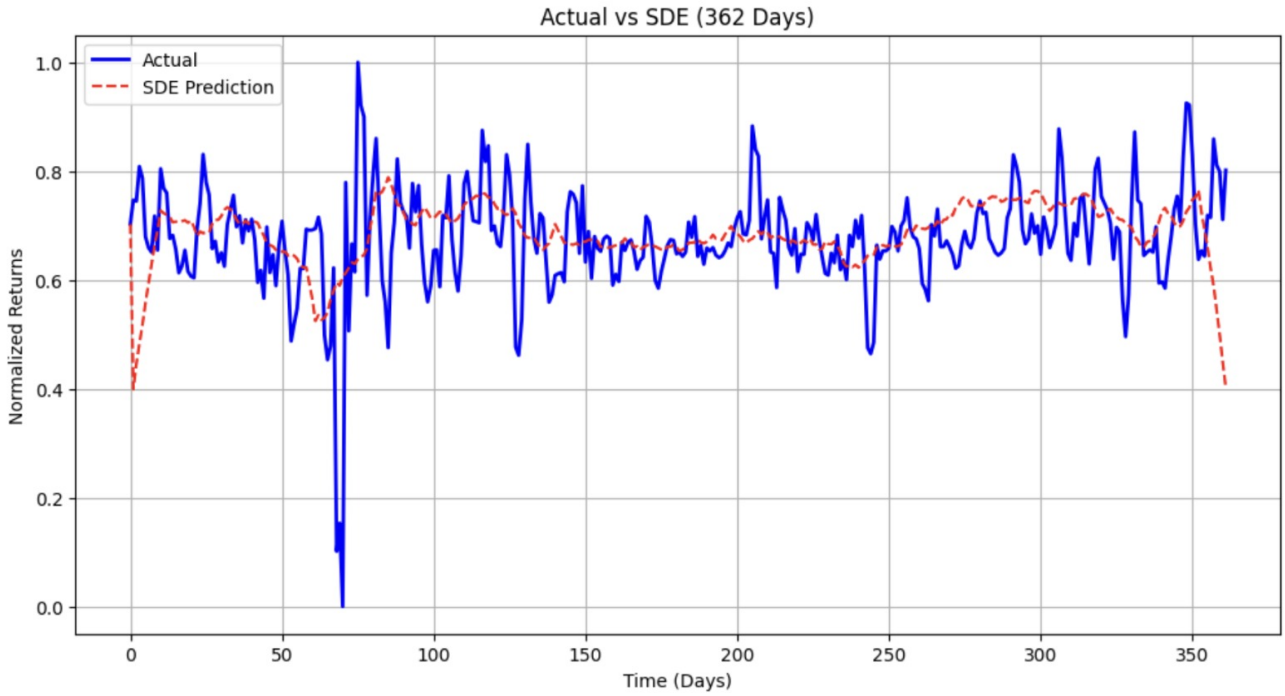

Figure 2: SDE predictions vs. Actual

It can be observed that towards the start and end of the predictions, significant ridges are formed for the simple SDE model. This is an indication that the very high levels of stochasticity are taking over the deterministic aspect, worsening predictions. In the middle section, the SDE model predicts a good average for the actual values, however, lacking the propensity to ebb and flow as per the data acquired. This is an indication that any deterministic factor must be extinguished and made stochastic, as was done in the following model.

3.3.3 Ornstein–Uhlenbeck Process

The Ornstein–Uhlenbeck (OU) Process is an adaptation of the previously mentioned SDE model, but this time it corrects for the over-diffusion problem (a phenomenon when the stochastic component is so large that it causes the solution process to become excessively noisy and spread out) (Wikipedia Contributors, 2024)[11]. The new equation that powers the OU model is the following:

$$dX_t = \theta(\mu - X_t)dt + \sigma dW_t, \tag{2}$$

where X_t, μ, and σ are aligned with the aforementioned definitions, and θ refers to the stochastic element controlling the rate of mean-reversion. The prediction graph therefore looks like this:

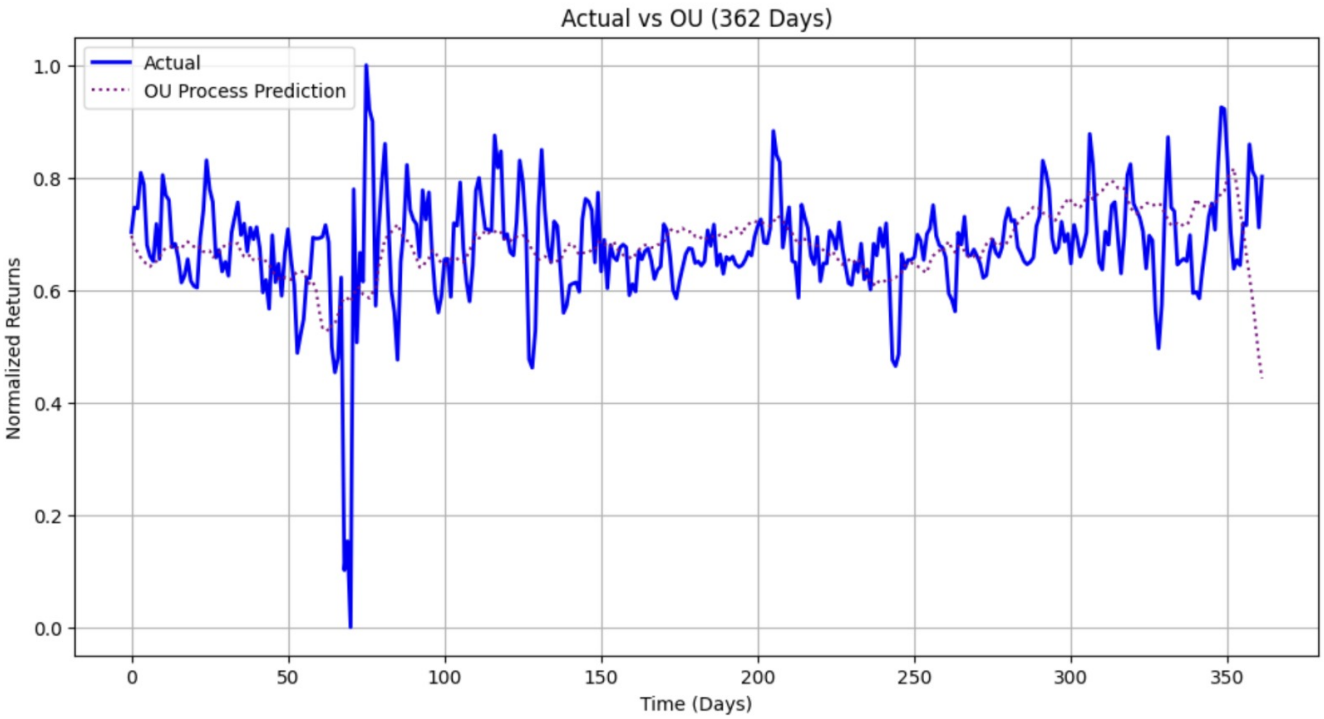

Figure 3: OU Prediction vs Actual

From the graph, it can be observed that the OU Process is better tailored to the ebb and flow of actual data better than the SDE model, owing to its ability to adhere stochasticity to the drift component and compensate for the over-diffusion. There is also a reduction in the ridges formed at the beginning and end of the data set, demonstrating its better suitability to capture actual data trends. However, it still does not align very well with the trend of price fluctuations.

3.4 Model Choice

To choose one of these models, the following measures have been used for justification:

Table 2: Evaluation Metrics for Model Comparison

Metric	**Definition**	**Mathematical Expression**
MSE (Mean Squared Error)	Measures the average squared difference between actual and predicted values.	$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
MAE (Mean Absolute Error)	Measures the average absolute difference between actual and predicted values.	$\frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
RMSE (Root Mean Squared Error)	Square root of MSE, giving error in original units.	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
AIC (Akaike Information Criterion)	Balances model fit and complexity. Lower is better.	$2k - 2\ln(L)$
BIC (Bayesian Information Criterion)	Similar to AIC but penalises complexity more heavily.	$k\ln(n) - 2\ln(L)$

Here, y_i represents the actual values, $\hat{y}_i$ are the predicted values, n is the number of observations, k is the number of parameters in the model, and L is the likelihood of the model given the data. These metrics will be used to objectively compare the performance of the model. Four specific lengths of time were of interest: Year Long, Half Year, One Month, and Half Month. These time lengths are of interest because the varying time lengths shall showcase different degrees of volatility, and test the models on different environments, providing them with a different level of uncertainty and extent of available data. Thus, the models will be tested upon how much they learn, their learning pace, and the quality of their predictions.

By running the models in a Python program, we obtained the following:

Table 3: Year Long

Model	**MSE**	**RMSE**	**MAE**	**AIC**	**BIC**
SDE	0.010886	0.104334	0.069284	-614.58	-606.79
OU Process	0.012384	0.111283	0.078309	-612.72	-601.04
LSTM	0.000207	0.014395	0.011022	-2066.24	-2027.32

Table 4: Half-Year Long

Model	**MSE**	**RMSE**	**MAE**	**AIC**	**BIC**
LSTM	0.000101	0.010072	0.008561	-637.311799	-598.373989
OU Process	0.028160	0.167810	0.125486	-642.142303	-635.745308
SDE Model	0.036076	0.189937	0.143711	-599.305074	-596.106577

Table 5: One Month Long

Model	**MSE**	**RMSE**	**MAE**	**AIC**	**BIC**
LSTM	0.00015	0.012248	0.009992	-230.145377	-206.325022
OU Process	0.033973	0.184319	0.146394	-97.465332	-94.662937
SDE Model	0.054003	0.232386	0.190097	-85.561285	-84.160087

Table 6: Half-Month Long

Model	**MSE**	**RMSE**	**MAE**	**AIC**	**BIC**
LSTM	0.000207	0.014370	0.012184	-93.278071	-81.241218
OU Process	0.004047	0.063618	0.048296	-78.645876	-77.229775
SDE Model	0.019757	0.140559	0.110039	-56.863793	-56.155743

From the tables above, it can be concluded that the LSTM model is by far the best performer and has been chosen to model the data set.

3.5 Inference and Investigation

From the numbers denoted in tables 3, 4, 5, and 6 above, it can be observed that for the AIC and BIC values, the LSTM model consistently records a value lower than or approximately equal to the other models for the four lengths of time considered. In addition to these values, the MSE, RMSE, and MAE values for LSTM consistently outperform and overshadow the other models. Despite the fact that the model consumes data in sequential bits, it has managed to outperform other models that focus on time as a continuous spectrum.

Many factors were included in this observation. Firstly, the acquisition of data. The data was translated to CSV form, where the prices were not recorded in continuous real time, but rather in discrete chunks of the day (an average price of the day represents the price of a cryptocurrency at that specific day), thus making the input data discrete. This data is then fed into the models as sequential inputs rather than a continuous spectrum of data, and the models aim to utilise these discrete chunks to approximate a continuous price value prediction for all the cryptocurrencies. This is beneficial for the LSTM model since it is designed for such discrete data processing, but becomes challenging for the other SDE-motivated models, since their time parameter thrives when there is a continuous flow of data, rather than discrete entries. When mapping time and price to continuous variables (using linear extrapolation between days), the LSTM still performed better. The LSTM model learns more effectively through the feeding of discrete data, since it processes, outputs, and then reintegrates that output into its input function, only to run it again and repeat the process iteratively. This ensures that the model is learning with time and, as more data is fed, it can capture trends and understand points of fluctuations much more rapidly, adapting accordingly. Such a fast rate of learning is not present amongst the SDE-driven models.

Secondly, the SDE-motivated models suffer from over-diffusion. The cryptocurrency price data acquired showcase their extreme volatility, and their random movements (stochasticity) are modelled by the diffusion element, namely the Wiener Process (Wikipedia Contributors, 2020)[12]. The Wiener Process captures the random walk performed by the crypto prices. When models are trained using acquired data, the long-term mean to which the process tends to, μ, which is also known as the drift factor, gets overwhelmed by the over-diffusion aspect of the data (very high stochasticity). This is due to it being a deterministic parameter in the SDE-motivated models, which results in the overall process being dominated significantly by the Wiener Process, thus creating ridges towards the start and end of the predictions. This makes the overall prediction model worse. The OU Process attempts to address this problem by attaching a parameter, θ, which controls the convergence rate for the drift factor to the estimated true mean of the data set, adding some degree of stochasticity to the otherwise deterministic factor. The effects are shown below:

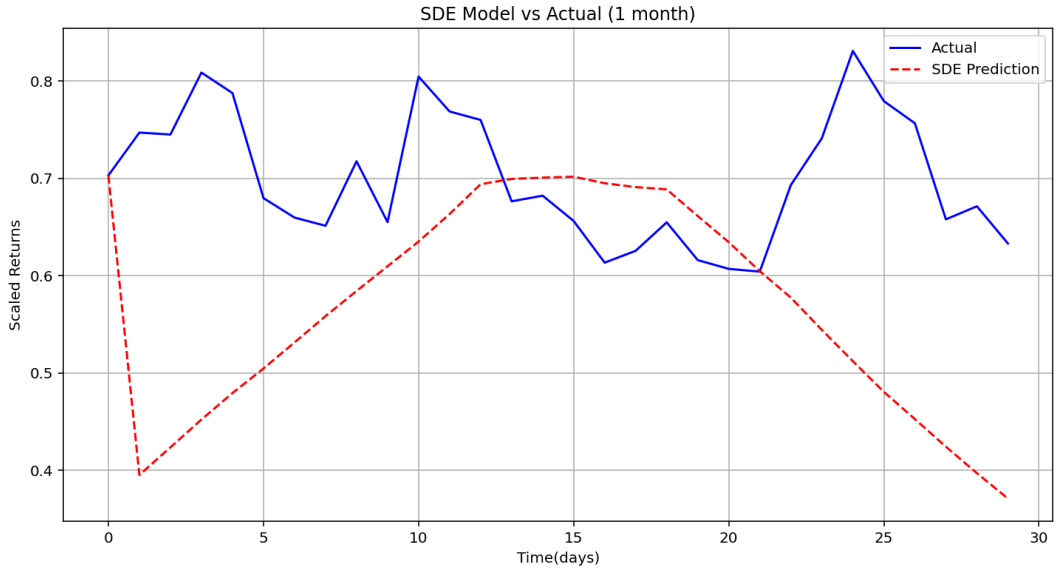

Figure 4: SDE predictions vs Actual (One Month)

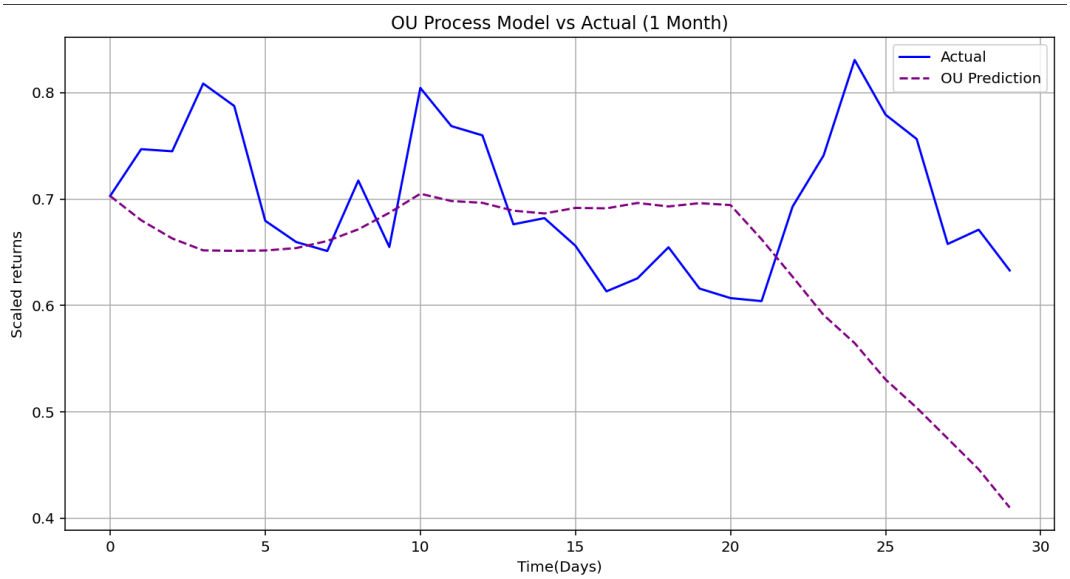

Figure 5: OU predictions vs Actual (One Month

The discrepancies are most observable in the depth of the ridges produced because of the over-diffusion problem on both the processes. The simple SDE model, with its deterministic drift parameter, being overwhelmed by the Wiener Process, produces a high rate of descent, and a relatively gradual ascent is followed. On the other hand, in the OU Process, there is a noticeable dip at the beginning; however, it is much more shallow in contrast to the simple SDE model.

3.5.1 Enhancing SDE Models

The metrics have shown that under standard programming construction, LSTM outperforms all of the SDE models. Therefore, their step sizes (called window sizes) were tweaked in order to optimise performance.

Price movements of cryptocurrencies have always largely been associated with random walks in major mathematical literature (Web3, 2024)[13], and therefore there is an impetus to modify the SDE-driven models so that a better prediction can be extracted. One of the most common ways to alter the models is to decrease the time-step size so that the models can learn, capture, and project trajectories based on previous data.

By programming the simulations in Python, the step-sizes were designed as functions of window sizes such that they are directly proportional to each other, where the greater the window size, the greater the time-step, and vice versa. The following pictures show the cases for different window sizes:

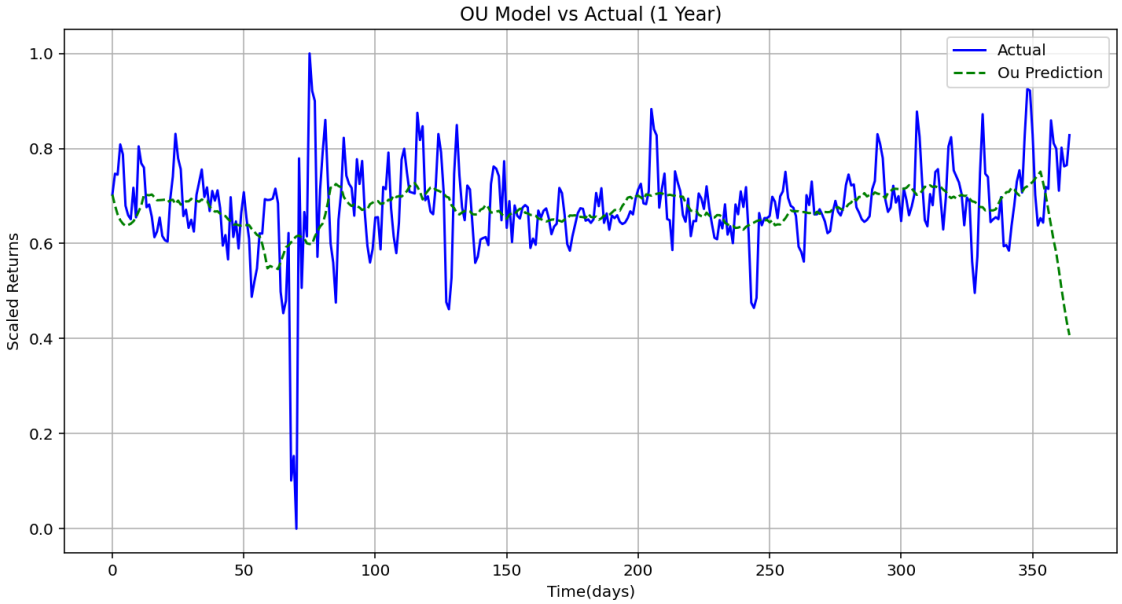

Figure 6: OU Process (Window Size = 24)

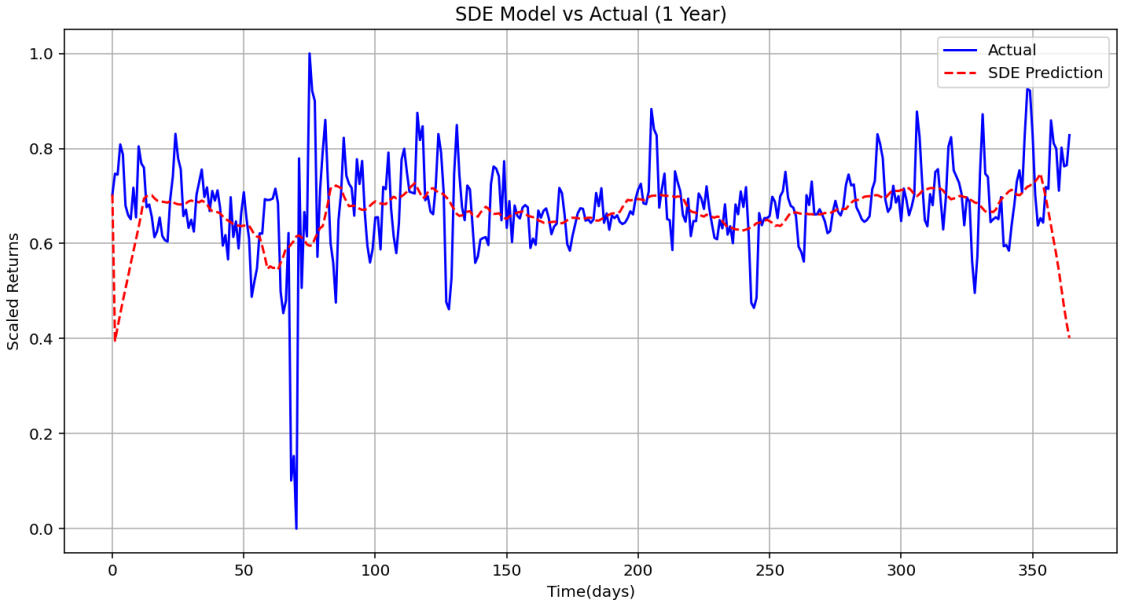

Figure 7: SDE Model (Window Size = 24)

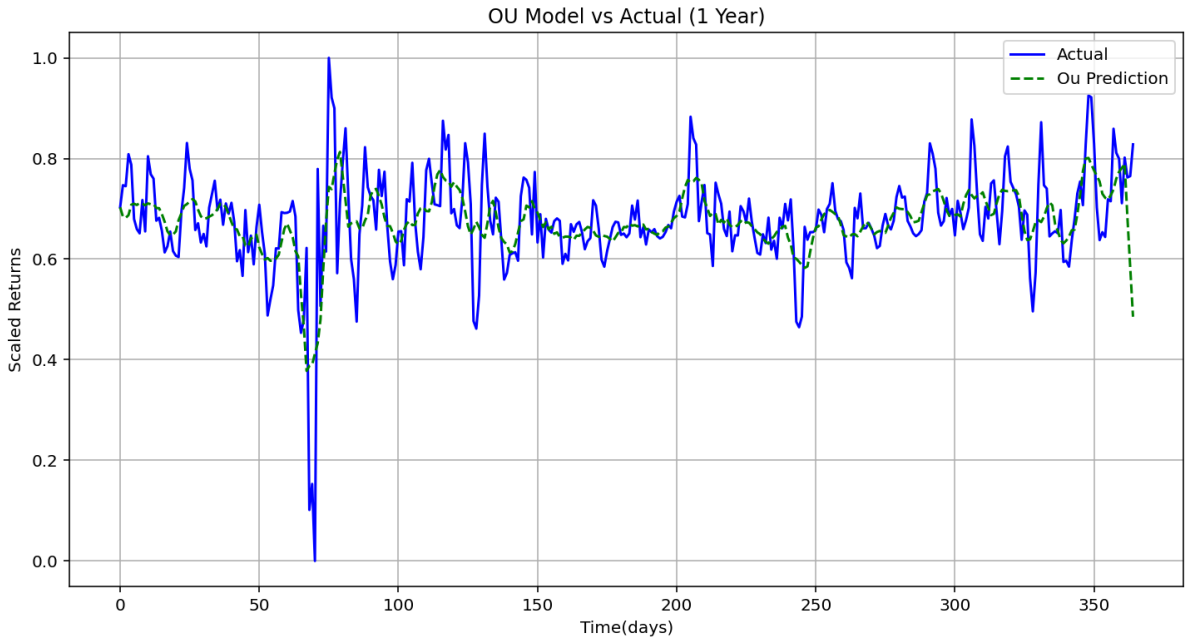

Figure 8: OU Process (Window Size = 8)

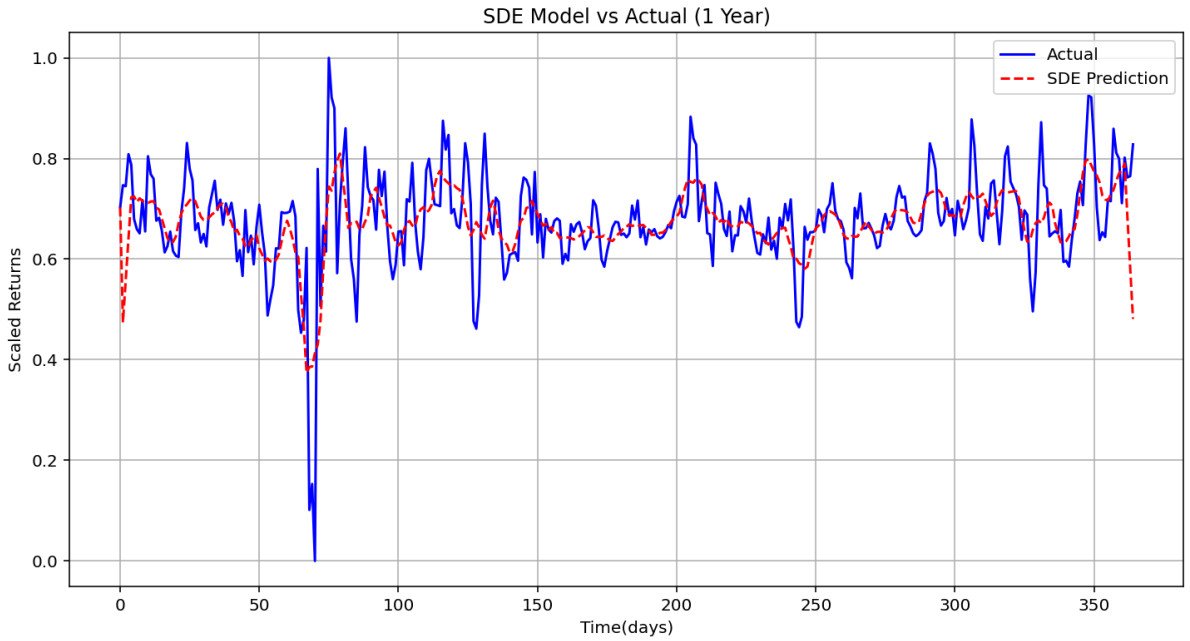

Figure 9: SDE Model (Window Size = 8)

From the graphs, it is clear that reducing the step-sizes better fits the actual data, and the predictions become more accurate. This is observed in the increase in the jaggedness of the predictions over time. Similarly, the ridges created because of the over-diffusion recover much faster with a smaller window size. There is, however, a risk of overfitting the data. This is because there is no built-in measure for overfitting prevention. Even when the step sizes are shortened, the models do not outperform the LSTM model because the latter is a machine learning model,

and can take a degree of decisions and retain some past information, making its predictions and projected trends much more aligned with reality. This is demonstrated by the AIC and BIC values of the tweaked models:

Table 7: Tweaked Model Comparison

Model	MSE	RMSE	MAE	AIC	BIC
LSTM	0.000234	0.015297	0.011421	-199.427	-195.536
OU Process	0.005773	0.075979	0.053126	-83.991	-82.821
SDE Model	0.006079	0.077971	0.054222	-82.344	-81.564

3.6 Expansion of LSTM

3.6.1 Neural Networks

A **neural network** is a computational model motivated by the structure and functioning of biological neural networks such as the human brain (IBM, 2021)[15]. They are composed of **nodes** arranged in **layers** (a group of interconnected neurons that process information in a connected way, acting as a singular computational unit in the network) constructing a neuron. The layers are as follows:

- **Input Layer:** Receives provided data.
- **Hidden Layers:** Perform transformations using weights, W, (numerical values associated between neurons that flag the degree of significance of an input data; $W \in \mathbb{R}^{m \times n}$, where m denotes the number of rows, which is equal to the number of neurons in the previous layer, and n denotes the number of columns representing the number of neurons in the next layer, while each entry, $w_{ij} \in \mathbb{R}$; $\mathbb{R}$ denoting the real numbers) and activation functions (in the context of neural networks, this function defines the output of a node given its input and weight transformations).
- **Output Layer:** Produces final predictions or classifications.

Nodes can be thought of as a linear regression program that connects to other nodes via defined relationships such as weights and the activation function. The essence of each node is to predict future values via running linear regressions within themselves, and the connection between each node determines how much influence a data set or data point has on the output, providing a framework to differentiate between impactful and impact-less data. Every node has within itself the input data, weights, bias, b, (a constant attached to the functions that allow values to shift in the negative or positive direction, preparing the algorithm for more diverse predictions; $b \in \mathbb{R}$), and a threshold (a specified value that dictates the activation of a neuron, essentially flagging a decision for the output values generated previously, and can be any number appropriately chosen).

When data is fed into a neural network model, it is injected into a node at the input layer. The data point is then multiplied with the weight values and added to the biases whose values themselves are determined by a process called **backpropagation**, the input data is transformed and sent to the hidden layer, where it is processed by the activation function. The value received by the activation function is the input value for that function. The resulting number received from

the function's operations is then further multiplied with weights and summed with biases that were predetermined through the use of backpropagation technique, at the output layer. A neural network with multiple nodes at the input, hidden, and output layer is referred to as a multi-layer neural network. A general multi-layer neural network, as such, is called a feed forward neural network since the input values move in only one direction towards the output.

3.6.2 Backpropagation Principles

Backpropagation is a type of algorithm used to compute the gradient of the loss function with the weights in a neural network. It works by propagating errors backward through the network using the chain rule of differentiation (Bergmann et al., 2024)[16].

For a neural network with a loss function L (a mathematical function that quantifies the difference between the predicted output of a model and the actual target values such as MSE for instance), weights W, and activations a, the gradient of the loss for the weights is computed using:

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial W} \tag{3}$$

where:

- $\frac{\partial L}{\partial a}$ represents the gradient of the loss function with respect to activations. It tells the direction and degree the activation values must change to reduce the loss function by sign and magnitude respectively.
- $\frac{\partial a}{\partial W}$ represents the gradient of activations with respect to weights, informing the degree of change necessary for the weights to improve prediction or determine impact upon the output. For instance, a large positive value shall indicate that the weight needs to be adjusted extensively for it to refine prediction, whilst a near-zero value shall indicate that the impact of the weight upon the output is negligible.

The derivative with respect to matrices and vectors can be taken by treating every entry as an independent variable (Oak, 2024)[17]. Using the chain rule, the derivative of the loss function, L, is taken with respect to the activation function and then multiplied by the derivative of the activation function with respect to the weight parameter. By computing the rates of change as outlined above, the algorithm updates the weights by injecting errors from the output to the input layer, further informing the performance of the model, informing the algorithm regarding the weights of importance on certain data points, and discarding ones that adversely affect performance.

The error propagates from the output layer back to the input layer, updating the weights using the computed gradients in an iterative process:

$$\frac{dW}{dt} = -\eta \frac{\partial L}{\partial W} \tag{4}$$

Solving this differential equation by separation of variables:

$$\int dW = -\eta \int \frac{\partial L}{\partial W} dt \tag{5}$$

which leads to:

$$W_t = W_0 - \eta \int_0^t \frac{\partial L}{\partial W} dt \tag{6}$$

Where W_t represents the weight matrix at the t^{th} iteration, W_0 represents the initial matrix, and η represents the learning rate established a priori.

For any loss function defined, it would be computationally expensive to brute force the error arising from the loss function for every input of the parameters involved. This is especially prevalent in multi-layered neural networks. Similarly, it would be computationally expensive to figure out the weights for every input present in a data set, and therefore, the computational efficiency and quality of output would be adversely affected. To fix this problem, the backpropagation system is aided by other algorithms, such as the **gradient descent algorithm** (IBM, 2021)[32], making the processing much smoother and smarter, and the learning more rapid and effective for the algorithm.

However, the performance of the feed forward neural network can only be enhanced to a limited extent via the use of such techniques (Alexandre et al., 2022)[18]. Instead, a transformation in the network system is shown to be more effective, and such a model is the **Recurrent Neural Network Model**, which employs recycling output data into the input stream to create much more accurate, consistent, and reliable predictions.

3.6.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural networks designed for digesting sequential data by maintaining a **hidden state** (a function of inputs at the same time step and also previous hidden states) at each time step that captures past information, allowing previous outputs to be used as further inputs (moving values from the hidden layer back to the input layer) (Amidi et al., 2019)[19]. The hidden states act as a "memory" function for the RNN, where at each time step, a new input along with the value from the hidden state is jointly injected into the input layer. This informs the output layer better, enhancing the algorithm's predictive trajectory.

When these hidden states are carried from one time step to the next, there is said to be established a feedback loop. A feedback loop is **unrolled** (dissected into its components) via copying the neural network path for each of the input values. A feedback loop needs to be unrolled only as much as the number of data points. For instance, a data set consisting of 50 inputs would require the feedback loop to be unrolled 50 times. However, the unrolling of multiple neural network paths does not require further parameters to be trained since they share the identical weight and bias parameters over each of the unrolled paths. This enhances the predictive performance of the algorithm, since a greater number of data is now being fed into the input layer, allowing the algorithm to learn better, and capture long term dependencies, making the predictions much more accurate in contrast to Feed Forward Neural Networks.

However, RNNs do encounter the massive problem of exploding and vanishing gradients. When a weight matrix contains values greater than 1 or less than -1, the extent to which the inputs are amplified follows the formula w_{ij}^n, where w_{ij} denotes a specific entry in the Weight matrix and n the number of data points in the set, and also the number of times the feedback loop has been unfolded. This causes the gradient values according to the backpropagation algorithm to explode, since the magnitude of changes required to minimise the loss function becomes very high. Therefore, the algorithm can only take large step sizes and in a finite number of steps, becomes very unlikely to find the optimal value that can minimise the loss function, since the algorithm is jumping around everywhere.

On the other hand, when all the values in the Weight matrix are in the range of $(-1, 1)$,

according to formula 6, the inputs are extensively deflated. This causes the gradient values as calculated by the backpropagation algorithm to be very low in magnitude, instructing little step sizes to be taken to find the optimal value that minimises the loss function. Therefore, in a finite number of steps, it again becomes practically improbable to find that optimal value, since the small step sizes prevent the algorithm from covering much of the data. This problem is addressed by the creation of **Long Short-Term Memory (LSTM)** layer, which combats the exploding and vanishing gradient problem by the addition of **cells** (the basic computational unit that processes information at each time step within a sequence; acting as a memory unit that retains information from previous time steps by passing a hidden state to the next cell), allowing the RNN to better understand the context of sequential data like time series (Christpher Olar, 2015)[20].

3.6.4 Long Short-Term Memory (LSTM)

LSTMs extend RNNs by introducing **gates** (mechanism in neural networks that controls the flow of information, determining the degree of past information kept, updated or discarded at each time step) to regulate the flow of information, addressing the problem of slow learning, exploding and vanishing gradients (Banoula, 2023)[21]. There are two distinct pathways through which data points can travel. Thus, the LSTM cell allows the handling of data points dynamically, as two distinct data points are injected into the cell at the same time. There is a long-term route which has no weights or biases to transform the value. The lack of weights causes the values to flow through a series of unrolled units of neural networks without causing the gradient to vanish or explode. However, the value is transformed by the outputs from the second route.

The second route consists of the hidden state, which showcases the short-term "memories". These are directly connected to weights and bias values, that modify the inputs. This is achieved primarily by the application of activation functions such as the Rectified Linear Unit (ReLU), Sigmoid function, and Hyperbolic Tangent function (tanh). Their formulation is as follows:

1. **ReLU:**
$$f_t = max(0, x_t) \tag{7}$$
2. **Sigmoid Function:**
$$\sigma(x_t) = \frac{1}{1 + e^{-x_t}} \tag{8}$$
3. **Hyperbolic Tangent:**
$$\tanh(x_t) = \frac{(e^{x_t} - e^{-x_t})}{(e^{x_t} + e^{-x_t})} \tag{9}$$

   where x_t is the t^{th} input.

These functions are embedded at the gates mentioned above, and their following functions have been listed:

- **Forget Gate**: Determines how much past information to retain. When an input enters from the input layer to the LSTM cell, it first has to pass through the Forget Gate. At the forget gate, the sigmoid function has been used. The function bounds the values between $(0, 1)$, and it gives a spectrum of decisions that can be taken regarding the data. When the value enters the gate, there are weights and biases that transform the value, before injecting it into the sigmoid function. The function then churns out a value in the aforementioned range. The

sigmoid function tells how much of an input travelling through the long-term route should be allowed to pass to the next stage. Whatever the value of the sigmoid function is calculated to be at the forget gate, it is then multiplied to the value at the long-term route, since that is the portion of that value required to be recalled.

- **Input Gate and Cell State Update**: When the value travelling along the short-term route is injected into the LSTM cell, the value is also put into the Input Gate and Cell State Update simultaneously. In the Cell State Update, the value is transformed via weights and biases, and then passed into the hyperbolic tangent function. The function binds the value between $(-1, 1)$. Simultaneously, the input value is also passed through the Input Gate, where weights, biases, and sigmoid function transform the value. The final product of these two transformed values is then summed to the value travelling along the long-term route, which is output as the new long-term value.

- **Output Gate**: Controls the output based on the cell state. The new long-term value is directly injected into the Output Gate, where it immediately enters the tanh function. Any value in the range of $(-1, 1)$ is produced, representing the potential short-term memory. The LSTM now decides how much of this new potential short-term memory to pass on via the use of a sigmoid activation function, taking the initial input along the short-term route. The product of this new number and the number output from the tanh function creates the value that represents new short-term memory. The process is then iteratively repeated for new inputs.

3.6.5 How the model works

The LSTM model used to capture cryptocurrency trends is an advanced version of the basic form outlined above. The LSTM cells have been unrolled, with each training instance (training data being fed into the neural network nodes with LSTM cells in place) being unrolled for 3 time steps. The unrolling operation is done implicitly via the TensorFlow program (TensorFlow, 2019)[22]. Due to the overtly volatile nature of cryptocurrencies, their price values differ significantly, and to counter that, the prices were normalised by the following equation:

$$x'_t = \frac{x_t - x_{min}}{x_{max} - x_{min}} \tag{10}$$

where x'_t denotes the normalised value, x_t denotes the input, x_{max} denotes the largest and x_{min} the smallest price values for a given cryptocurrency. The normalisation binds the values in the range of $(0, 1)$, and makes calculations easier and less expensive computationally. The primary motivation for normalising the sequence of inputs was to ensure that the algorithm did not jump around due to the massive changes that may be required for the highly volatile nature of the dataset. This prevents the gradient values from drifting too far, and by design, addresses the over-diffusion problem that plagues the SDE-motivated models. It iterates through the list of cryptocurrencies in the dataset. It takes the value and passes it through the weight and biases, which are determined by TensorFlow and are optimised by the **Adam Optimiser** algorithm (Kingma et al., 2014)[34] (it was chosen since it is a **stochastic gradient descent (SGD)**(Ruder, 2017)[33] algorithm, which would be able to capture dynamic learning rates, correct bias, and make the whole algorithm much more efficient):

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W} \tag{11}$$

$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b} \tag{12}$$

where the loss function, L, is chosen to be the MSE ($L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ for n number of data points), W, b referring to the weight and bias respectively, and η, the learning rate, is set at 0.001 (controlling step size). The output is fed into the **Dense Layer** (a fundamental layer where every neuron is connected in the previous layer to every neuron) of the neural network after passing through the LSTM cell. The values are made to pass through two stages: The first stage contains the $ReLU$ activation function according to equation 7, and the second stage has a linear transformation in the form of $y_t = Wx_t + b$, where x_t denotes the value coming out of the first stage in the dense layer. The final value produced is the predicted value, and this is repeated iteratively for all the data points, providing predictions for all the cryptocurrencies in the dataset.

The primary reason the LSTM model outperforms the SDE-motivated models is due to the data being fed into the model in a sequential form, and its iterative processes naturally mitigating the over-diffusion problem, enabling better handling of long-range dependencies by selectively retaining and forgetting information across sequences. This provides an ideal atmosphere for binary decision-making situations such as modelling price movements of cryptocurrencies, since there are practically two trajectories for the price values to move towards (up and down only due to the high volatility). This is evident from the following pictures, relative to figures 4 and 5:

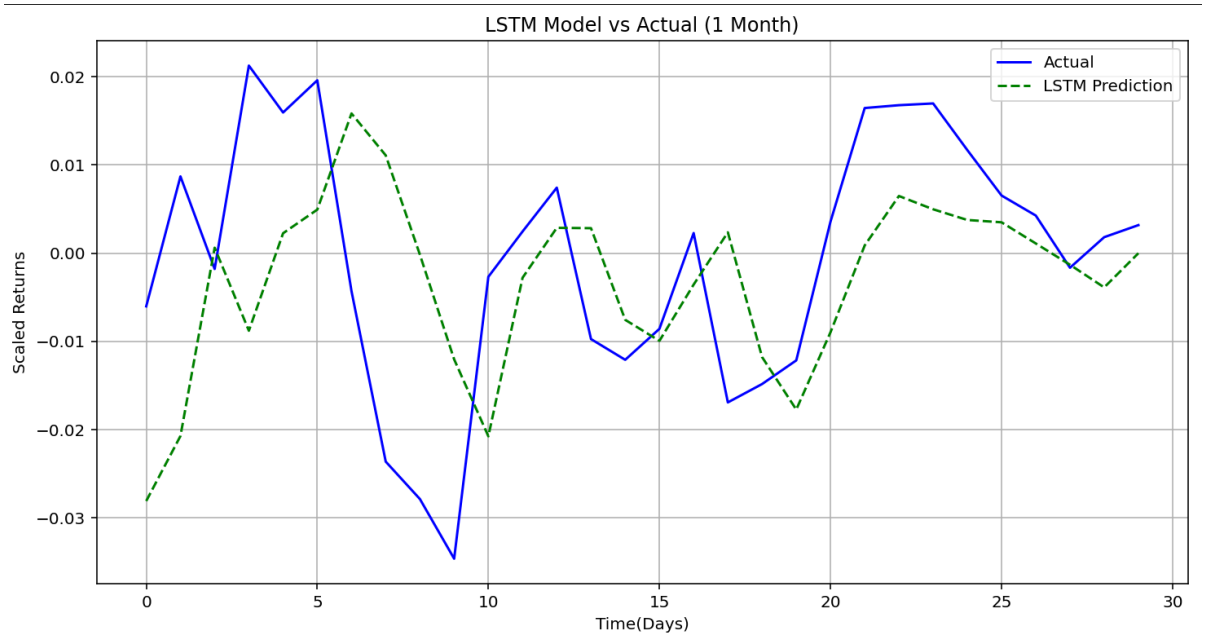

Figure 10: LSTM Predictions vs. Actual (Half-Month)

By comparing figures 4, 5, and 10, it can be inferred that the trends predicted by the LSTM model mimic the trend much better than the SDE-motivated models. This is due to the previously mentioned properties of gates, decision-making properties through sigmoid function, and being unaffected by over-diffusion problem, since it reads the data effectively and captures trends much more profoundly, providing much better predictions, further validated by the training loss versus

validation loss graph[3]:

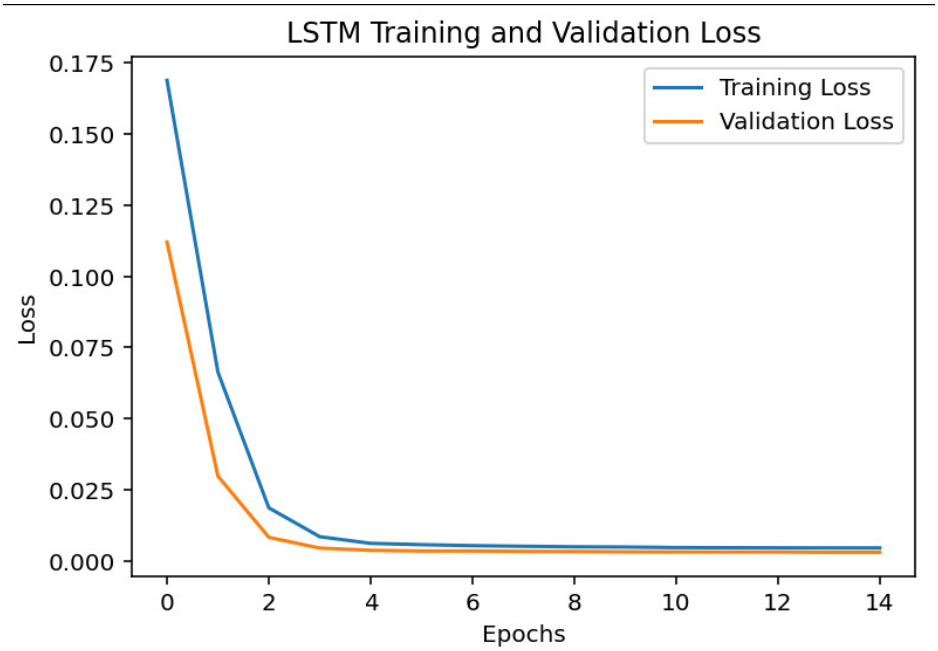

Figure 11: Training and Validation Loss of LSTM

3.7 Model Conclusion

From all the aforementioned tests, tweakings, and simulations, it has been demonstrated that the LSTM model outperforms the SDE-motivated models extensively, mostly owing to its natural tendency to correct for over-diffusion due to the cells installed in the neural network algorithm, which, by the use of sigmoid function, can ascertain how much of data fed into the model to forget, input, output, and then reintegrate back into the algorithm again (the use of Forget Gate, Input Gate, and Output Gate). Although reducing step sizes helps for the SDE models, they do not win over the simplicity, accuracy, and versatility of the LSTM model due to its intrinsic capability to address the over-diffusion problem by preventing memory values from drifting away through the use of condensation functions such as tanh. This leads to LSTM ultimately being used as the engine behind cryptocurrency price prediction, price values, and will act as the core motivation for predicting trends that will inform the user about the current state of the market, dictating the portion of wealth to invest, short, and sell, while providing an estimate of expected returns with the recommended portfolio.

4 Methods

The method of acquiring the data and arranging into a time-slotted Excel spreadsheet has resulted in discrete data, which is very well captured by the LSTM model when compared to a general SDE model and the Ornstein–Uhlenbeck Processes. The metrics confirm the choice of model as the most logical one to pursue, leading to its application in the prediction algorithm.

[3]The dropout factor affecting the training loss does not affect the validation loss and thus, it is lower than the training loss.

4.1 Portfolio Creation

From the predicted returns and the co-variance matrix, we can calculate the portfolio with the global minimum risk and the diversified portfolio with the equations:

$$\boldsymbol{\xi_g} = \frac{1}{\alpha}\mathbf{1}V^{-1}$$

$$\boldsymbol{\xi_d} = \frac{1}{\beta}\mathbf{r}V^{-1}$$

where $\boldsymbol{\xi_g}$ refers to the global minimum variance portfolio (portfolio with minimum variance given all feasible portfolios) vector, $\boldsymbol{\xi_d}$ represents the diversified portfolio vector, $\mathbf{1}$ is the row unit vector, V is the matrix of co-variates, and $\mathbf{r}$ is a vector of the expected price change of stocks, and for any row vector, $\mathbf{v}$, $\mathbf{v}^T$ refers to the transpose of the vector.

The scalars, α and β, are defined as:

$$\alpha = \mathbf{1}V^{-1}\mathbf{1}^T$$

$$\beta = \mathbf{1}V^{-1}\mathbf{r}^T$$

$$\gamma = \mathbf{r}V^{-1}\mathbf{r}^T$$

From there, it is possible to calculate the mean-variance efficient portfolio (Chen, 2019)[24] for a specific return using the formula:

$$\hat{\boldsymbol{\xi}} = \frac{\gamma - \beta\mu}{\alpha\mu - \beta^2}\boldsymbol{\xi_g} + \frac{\alpha\mu - \beta}{\alpha\gamma - \beta^2}\boldsymbol{\xi_d}$$

This then returns weightings for each currency, which will provide that return with the minimum risk. Using this, an Efficient Market Frontier can be created (Team, 2024)[25], showing the minimum risk for the input return. By getting the general risk preference (low/medium/high) from the user, the target return can be estimated based on the average returns of all cryptocurrency. While this method provides the best portfolio based on the parameters, it does not take into account any personal preferences of the user, although that cannot be quantified into a mathematical model and requires the user to make that decision independently.

4.2 Prototype Model

The prototype consists of two modules designed to optimise the run-time and enhance usability. These modules are the Data Scraper and the Portfolio Generator. By separating data collection from portfolio generation, the prototype ensures smoother performance, reduces computation time, and allows for greater flexibility in user interactions.

4.2.1 Data Scraper

The Data Scraper first imports the necessary modules and sets the working directory to the folder where the program is saved. This ensures smooth execution and correct access to previous data without requiring user intervention, improving usability.

Next, the program reads data from `cryptocurrencies_prices_list.csv`, verifies that the order of cryptocurrencies matches the search algorithm, and saves the most recent data entry. It

also sets today's date, retrieved from the `datetime` module, as the end date. Automating these steps reduces user input and makes the program more efficient.

Once the start and end dates are confirmed, the scraper collects data for each cryptocurrency[4], averaging the prices over the day before appending the results to the dataset. Over time, this continuous accumulation of data improves the models accuracy.

CoinGecko (CoinGecko, 2025)[9] - the data source - however, limits free users to retrieving data from to last 365 the days. If the most recent entry in the CSV file is from further away than a year, the program generates a new CSV containing only the last 364 days of prices.

Separating the scraper from the main program improves portfolio generation efficiency. Due to website restrictions that prevent API abuse, the bot introduces randomised pauses between 25 to 60 seconds per request. As a result, a complete run takes at least 7 minutes. Running the scraper separately ensures that users can generate multiple reports with different risk preferences without unnecessary delays or computational power.

The final component of the Data Scraper program is an error-handling mechanism that interprets common host-site errors and relays them in a user-friendly format. For example, messages such as: `"Rate limit hit for {crypto_id}. Waiting before retrying..."` make troubleshooting easier, improving the program's accessibility

4.2.2 Portfolio Generator

Similarly to the Data Scraper, the Portfolio Generator imports necessary modules and sets the working directory to ensure seamless operation.

The user is then prompted to enter a risk preference: low (1), medium (2), or high (3). Each risk level determines the number of cryptocurrencies included in the portfolio, based on their volatility. For example, a low-risk portfolio consists of only Bitcoin, Ethereum, and USD-Coin.

The user then specifies a desired return value, which plays a key role in portfolio diversification calculations. The program verifies that the input is between 0 and 1, raising a ValueError if it falls outside this range.

The Portfolio Generator consists of three main components: the Model, Diversified Portfolio Calculation, and PDF Generation.

- Model: This component employs an LSTM-based prediction model (described earlier in the report) to forecast cryptocurrency prices for the next 30 days. The expected returns from these predictions are computed, and various graphs are generated to illustrate the price trends over the last 360 days and projected values for the next 30 days. These graphs are saved as PNG files for later use in reports.

- Diversified Portfolio Calculation: This step follows the mathematical framework outlined in section 4.1. It incorporates the user's desired return, expected returns from the model, and a covariance matrix calculated from the last 360 days of data. Using rolling data for the covariance matrix ensures that recent trends are taken into account without overshadowing long-term relationships between cryptocurrencies.

- PDF Generation: Finally, the program compiles the portfolio report into a well-structured PDF. This document includes the date generated, relevant graphs, and information on the Minimum Variance Efficient Frontier (4.1). It presents expected returns alongside a simple

[4]Cryptocurrencies List: Bitcoin, Ethereum, USD-Coin, Terra-Luna, Maker, Eos, Lido Dao, Huobi Token, Near, Internet Computer, Aptos, Monero, Bitcoin Cash, Chainlink, Uniswap, Cardano, Tron

explanation, such as "Short 0.01 in Bitcoin", making it easy for users to interpret their results. The PDF is saved in the program directory for easy access.

By structuring the prototype in this modular fashion, the program optimises both data collection and portfolio generation while maintaining user-friendliness and efficiency.

5 Results

The pdf report will provide the user with the predictions of each cryptocurrency provided in the designated risk band, as well as percentage results of portfolio weights.

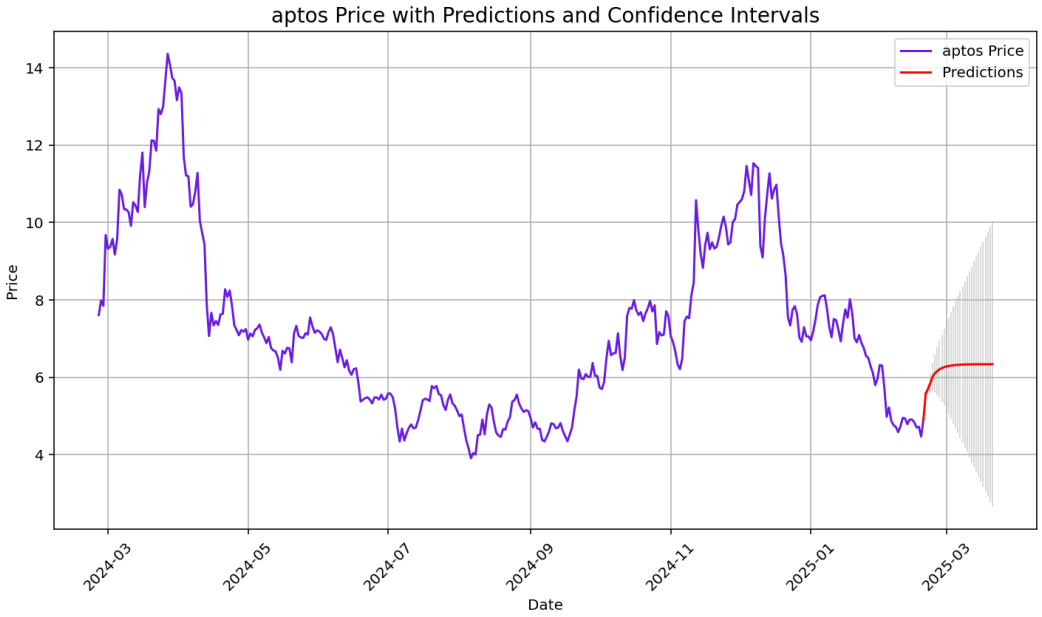

Figure 12: Price Prediction Examples

Figure 12 shows an example of this, as it estimates both future prices and confidence intervals. The red line shows the prediction for 30 days. Initially, the model prediction is precise, however, after about a week it starts to plateau. The line then tends to level out and the 95% confidence intervals widens towards the tail of the predictions.

When the report is produced, some values are positive and negative regarding the percentage to be invested. The inference is that the user should buy the positive percentages and short the negative percentages. An example of such a report produced is as follows:

Crypto	Expected Return	Portfolio Proportion	Recommendation
bitcoin	6123.527199	0.000027	Invest 0.000
ethereum	206.958997	0.000174	Invest 0.000
usd-coin	0.003582	-1.039078	Short 1.039
terra-luna	0.000005	3.117321	Invest 3.117
maker	10.201601	0.000268	Invest 0.000
eos	0.000100	-0.510820	Short 0.511
lido-dao	0.027117	-0.351953	Short 0.352
huobi-token	-0.039384	-0.215939	Short 0.216

The generated portfolio will lie produce an expected return of 0.2 and lies on the efficient market frontier (EMF) as demonstrated in Figure 13 (Hanicova)[14]. This means that the portfolio minimises risk for the level of return (0.2) by diversifying out the of non-market risk.

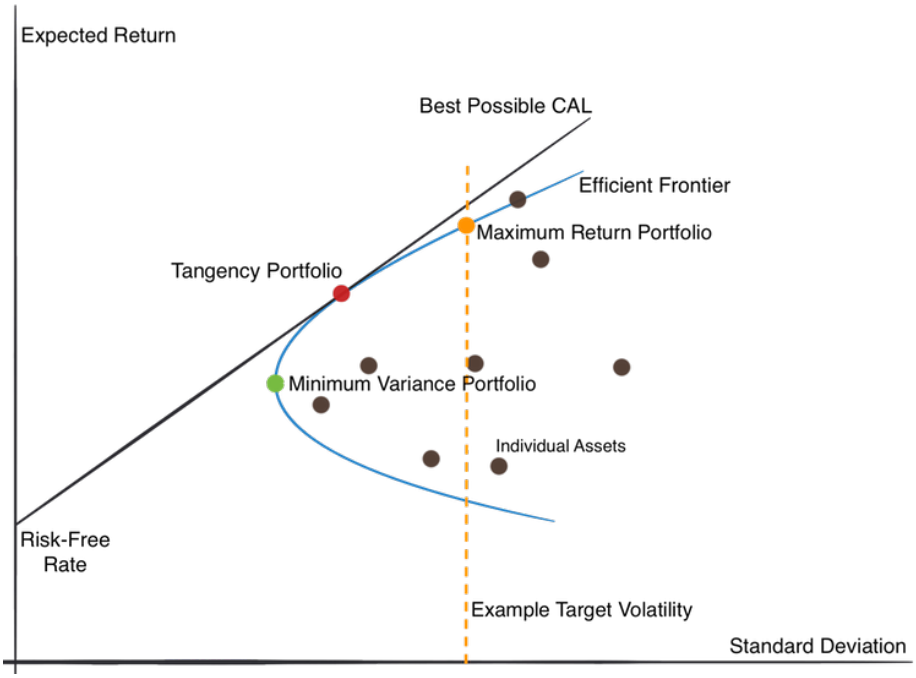

Figure 13: Efficient Market Frontier

The overall volatility of the portfolio will be lower than the variance for each individual currency since it uses shorting to mitigate the risk. The main benefit of portfolio diversification is that even though a single cryptocurrency might be very volatile, and its predictions may not capture a very narrow range, as demonstrated in figure 12, the overall volatility of the portfolio will be lower.

Through testing, it was discovered that the model tends to invest sparsely in Bitcoin and Ethereum due to their high volatility, investing heavily into USD coin and Terra-Luna due to their relative stability. After further analysis, it can be determined that the model tends to invest a majority into stable currencies, which hold their value while deriving all the returns by investing a smaller percentage into high-return, high-variance currencies like Bitcoin.

6 Conclusion

6.1 Overview

Due to the fact that the model only predicts cryptocurrencies, every asset considered for the portfolio is a risky asset. Many portfolios consider a risk-free asset to help reduce risk. The model could be further improved by including a risk-free asset, such as fixed term deposits with a set maturity date; however, since cryptocurrency is a risky investment medium, no such cryptocurrency asset exists. The model tends to favour low-volatility, high-return coins. This does not impact model performance, although it may not match a user's preferred investment strategy.

The report suggests that the optimised portfolio allocates most of its wealth in stable coins, as opposed to Bitcoin and Ethereum, which are more volatile. The model meets expected returns by investing very small proportions into these currencies, and displaces this risk with the stable currencies.

The cryptocurrency market has a range of unique challenges when prioritising portfolio optimisation, one of which is the shear number of currencies available, and another is that they are individually very volatile. It is also heavily affected by new legislation and the media. Traditional portfolio creation techniques, such as (Investopedia (2023))[31] Markowitz's Modern Portfolio Theory (MPT) lie under the assumption that correlation is stable and there are predictable returns, making them less valuable when dealing with volatile assets. To address this, we use machine learning-driven portfolio optimisation, in tandem with MPT in this study, with a focus on Long Short-Term Memory (LSTM) to forecast the prices of cryptocurrencies.

6.2 Limitations

The model is limited by data collection, as CoinGecko allows historical retrieval of data from only 365 days prior for free users. The volatility of cryptocurrencies is a major challenge that makes long-term price prediction unreliable. SDE-based models are prone to over-diffusion since the extreme volatility of the currency implies very high levels of stochasticity in the behaviour of price bands, which overshadows any deterministic components (there is no sustained tendency to revert to any specific value). Both types of model are purely statistical and do not consider external factors such as adoption rates, regulations, or market sentiment. When it comes to portfolio recommendations, the model does not consider the behaviour of an individual investor beyond an assumed risk tolerance. Thereby, the model is not tailored per the desires of the user, but rather has an underlying basis (derived from strict statistical definitions) which defines levels of risk. Although LSTM outperformed other models, it may still suffer from over-fitting, especially when using a higher batch size. The model does not factor in external macroeconomic factors that impact cryptocurrency prices such as global or national economic trends. (Turner, J. (2023))[42]It also has not incorporated black swan events such as the 2007 financial crisis, or the (Canan Sevgili, Laudani, P., Parodi, A. and Chiumento, A. (2025))[41] COVID-19 pandemic, that directly affect the market and large shifts have been observed in the price bands owing to such events. Due to the nature of such events, the model has no way of predicting impactful changes like these. Portfolio allocation remains static and lacks live real-time rebalancing based on market conditions and data regarding parameters that influence macroeconomic atmosphere and trends, it is only upon updating the data it can be.

6.3 Comparison

Due to the fact that there are almost no widely available or free-to-access cryptocurrency portfolios, comparing the program with others was not feasible. Compared to a best available portfolio, 1, the program generates a portfolio with a much lower theoretical risk value, which implies that the portfolio generated is much safer to invest in; however, many factors remain unconsidered. For example, the assumption that the price of cryptocurrencies follows the weak market hypothesis (future prices only rely on their historical prices) is not exactly representative of reality since macroeconomic atmospheres, trends, and media coverage also significantly affect cryptocurrency prices.

When compared against conventional methods of constructing portfolios, some differences arise for the developed model. The LSTM model considers prices as time steps and analyses trends and price patterns to predict future price bands, which is contrastive to MPT, which simply utilises the correlation between assets and historical volatility to optimise asset allocation in the portfolio. This implies that MPT assumes a static correlation structure, unlike the dynamic construction adopted by the LSTM model. Due to the overtly volatile nature of cryptocurrency relative to assets such as stocks and bonds, the model tends to invest in the more stable cryptocurrencies, and diversify through risk-parity strategies, balancing the rate of return against an acceptable degree of volatility and risk.

Further, comparing the results with (vBarone, 2021)[30] Momentum and factor-based investing (investing based on how assets have performed recently, where investments tend to rise if an uptick in price was observed, as buyers bet on the trend holding, and vice versa), such as high Sharpe ratio tests (Fernando, 2024)[36], the model incorporates momentum to a degree. This level of behaviour is very natural among investors and is replicated in the model. However, the period between holding and selling is much shorter for the cryptocurrency market compared to the regular stock market. This is well covered by the LSTM model, as it can adapt to the new influx of data injected at every iteration and provide new predictions based on updates.

6.4 Future Improvements

Looking into the future, a major factor in the crypto space is how the market changes based on market sentiment and new regulations, using a (Financial Services, 2021)[40] Natural language processing (NLP) based sentiment analysis from different news sources and social media outlets. This has the potential to improve the accuracy of the pricing points.

Similarly, utilising finer machine learning models also possesses the potential to enhance predictive aptitude. Alternative versions of the model, such as: Transformers (Ankit, 2022)[37], Gated Recurrent Units (GRUs) (Chung et al., 2014)[39], and hybrid models [28] like CNN-LSTM (Convolutionary Neural Networks with adhered LSTM) or Attention-LSTM Kang et al., 2023)[38], promise to make predictions more accurate and reliable. Transformers use self-attention mechanisms that allow them to consider all previous steps simultaneously. Unlike LSTM which processes them one by one, GRUs are more computationally efficient as they have fewer parameters; however, they are not as accurate as LSTM (Yang et al., 2020)[27]. The advanced models have properties that the simple LSTM lacks, such as:

1. A hybrid model, for instance CNN-LSTM, is effective in detecting short-term patterns, so preprocessing data before it is input into the LSTM would allow LSTM to focus on broader

trends (Omarkahn et al., 2022)[28].

2. An Attention-LSTM allows the model to allocate different weights to past observations; this helps reduce volatility as it would prioritise critical price changes over less extreme fluctuations (Subhradipta, 2023)[29].

The few aforementioned measures will streamline the computing process and provide more reliable and accurate predictions.

Furthermore, API connectivity can be integrated to real-time data sources, which will provide real-time influx of data that will not only enhance predictive capabilities, due to an increase in available data, but also keep the algorithm updated to the latest possible information, upgrading its aptitude. The predictions would be more accurate, precise, and reliable. Backtesting this data against historical and live data would increase the robustness of the model in an extremely volatile market such as the crypto space. Similarly, fine-tuning the risk aptitude and measure by making the degree of risk in the range of $(0, 1)$ (where 0 currently denotes lowest possible risk and 1 the highest) to a continuous spectrum (where the user will be able to choose any number in that range per their risk preference) or increase the number of options available to the user in discrete form such as introducing new levels of risk selection: Very low, low, low intermediate, intermediate, high intermediate and high. This will make the model more user-friendly and attractive to the consumer due to the greater capability of personalisation. The integration of NLP, which was previously mentioned, to integrate information from media coverage would also enhance performance, as input from more parameters would ensure better learning and increase predictive aptitude of the model.

A User Guide

Introduction This is a comprehensive guide to the **Portfolio Generator** and **Crypto Price Updater** used for cryptocurrency data collection and portfolio optimisation

Prerequisites

- Install required libraries:

```
pip install numpy pandas matplotlib tensorflow scikit-learn time requests
```

- Ensure Python 3.x is installed.

How to Run

1. Update the price data if necessary (run `Crypto Price Updater.py`).
2. Open a python environment.
3. Navigate to the appropriate directory.
4. Run the script
5. Follow the on-screen prompts that appear when running if using the Portfolio Generator
    - Choose a risk level (1-Low, 2-Medium, 3-High)
    - Enter an expected return goal (0-1)
6. The script generates a PDF report in the directory named `Crypto_Investment_Report.pdf`.

The program reads existing data from `cryptocurrencies_prices_list.csv`. It then uses the **Minimum Variance Efficient Frontier (MVEF)** model to optimise portfolio allocation. It then applies the LSTM model to generate predictions for the prices of cryptocurrencies. Produces a PDF report with:
- Expected returns for each cryptocurrency
- Optimal portfolio allocation
- Cryptocurrency price predictions presented graphically

References

[1] Reiff, N. (2021). What's the environmental impact of cryptocurrency? [online] Investopedia. Available at: https://www.investopedia.com/tech/whats-environmental-impact-cryptocurrency/.

[2] Coinbase (2020). What is 'proof of work' or 'proof of stake'? [online] Available at: https://www.coinbase.com/en-gb/learn/crypto-basics/what-is-proof-of-work-or-proof-of-stake [Accessed 13 Feb. 2025].

[3] Coinbase. (2025). Bitcoin. [online] Available at: https://www.coinbase.com/en-gb/price/bitcoin [Accessed 13 Feb. 2025].

[4] Clarity in Crypto (2021). Clarity in Crypto. [online] Clarity in Crypto. Available at: https://www.clarityincrypto.com/crypto-portfolio-allocation [Accessed 13 Feb. 2025].

[5] Scott Likens (2023). Making sense of bitcoin and blockchain. [online] PwC. Available at: https://www.pwc.com/us/en/industries/financial-services/fintech/bitcoin-blockchain-cryptocurrency.html.

[6] La Morgia, M., Mei, A., Sassi, F. and Stefa, J. (2024). The Doge of Wall Street: Analysis and Detection of Pump and Dump Cryptocurrency Manipulations. Ithaca, pp.2–8.

[7] Yi, E., Yang, B., Jeong, M., Sohn, S. and Ahn, K. (2023). Market efficiency of cryptocurrency: evidence from the Bitcoin market. Scientific Reports, [online] 13(1). doi:https://doi.org/10.1038/s41598-023-31618-4.

[8] Siddiqui, O. (2022). Markowitz Efficient Frontier. [online] Learnsignal. Available at: https://www.learnsignal.com/blog/markowitz-efficient-frontier/.

[9] CoinGecko. (2025). [online]. Available at: https://api.coingecko.com/api/v3/coins/crypto_id/market_chart/range.

[10] Kaur, G. (2024). What is random walk theory, and what are its implications for cryptocurrencies? [online] Cointelegraph. Available at: https://cointelegraph.com/learn/articles/random-walk-theory-what-are-its-implications-for-cryptocurrencies.

[11] Wikipedia Contributors (2024). Ornstein. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Ornstein%E2%80%93Uhlenbeck_process.

[12] Wikipedia Contributors (2020). Wiener process. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Wiener_process.

[13] Web3 (2024). Random Walk Theory and Its Cryptocurrency Implications. [online] Medium. Available at: https://medium.com/coinmonks/random-walk-theory-and-its-cryptocurrency-381d3434a1f8.

[14] Daniela Hanicova. Available at: https://quantpedia.com/markowitz-model/

[15] IBM (2021). What is a Neural Network? [online] Ibm.com. Available at: https://www.ibm.com/think/topics/neural-networks.

[16] Bergmann, D. and Stryker, C. (2024). What is Backpropagation? — IBM. [online] Ibm.com. Available at: https://www.ibm.com/think/topics/backpropagation.

[17] Oak, O. (2024). Training ANNs: A deep dive into Backpropagation and Gradient descent. [online] Medium. Available at: https://medium.com/@omkarsoak/training-anns-a-deep-dive-into-backpropagation-and-gradient-descent-bea84ff5273c.

[18] Alexandre, J., Nauber, F., Paulo, J., Li, T. and Fong, S. (2022). Artificial neural network-based approaches for computer-aided disease diagnosis and treatment. Cognitive and Soft-Computing Techniques for the Analysis of Healthcare Data, (2022), pp.79–99. doi:https://doi.org/10.1016/b978-0-323-85751-2.00008-6.

[19] Amidi, A. and Amidi, S. (2019). CS 230 - Recurrent Neural Networks Cheatsheet. [online] Stanford.edu. Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks.

[20] Christopher Olar. Understanding LSTMs [online] Github. Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[21] Banoula, M. (2023). Introduction to Long Short-Term Memory(LSTM) — Simplilearn. [online] Simplilearn.com. Available at: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm.

[22] TensorFlow (2019). TensorFlow. [online] TensorFlow. Available at: https://www.tensorflow.org.

[23] Araújo, T. and Barbosa, P. (2023b). Reconstructing cryptocurrency processes via Markov chains. [online] arXiv.org. Available at: https://arxiv.org/abs/2308.07626?[Accessed 4 Mar. 2025].

[24] Chen, J. (2019). Mean-Variance Analysis. [online] Investopedia. Available at: https://www.investopedia.com/terms/m/meanvariance-analysis.asp.

[25] Team, C. (2024). Efficient Frontier. [online] Corporate Finance Institute. Available at: https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/efficient-frontier/.

[26] Polizu, C., de la Mata, M., Kanaster, T., Gupta, S., Guadagnuolo, L., Birry, A., Denslow, N. and Donaghey, C. (2023). Are crypto markets correlated with macroeconomic factors? [online] S&P Global. Available at: https://www.spglobal.com/en/research-insights/special-reports/are-crypto-markets-correlated-with-macroeconomic-factors?[Accessed 4 Mar. 2025].

[27] Yang, S., Yu, X. and Zhou, Y. (2020). LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. [online] IEEE Xplore. doi:https://doi.org/10.1109/IWECAI50956.2020.00027.

[28] Omarkhan, M., Kissymova, G. and Akhemetov, I. (2022). Handling data imbalance using CNN and LSTM in financial news sentiment analysis — IEEE Conference Publication — IEEE Xplore. [online] ieeexplore.ieee.org. Available at: https://ieeexplore.ieee.org/document/9663802.

[29] Subhradipta (2023). Time-Series Forecasting Using Attention Mechanism. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2023/06/time-series-forecasting-using-attention-mechanism/ [Accessed 10 Mar. 2025].

[30] vBarone, A. (2021). Introduction to Momentum Trading. [online] Investopedia. Available at: https://www.investopedia.com/trading/introduction-to-momentum-trading/.

[31] Investopedia (2023). Modern portfolio theory (MPT). [online] Investopedia. Available at: https://www.investopedia.com/terms/m/modernportfoliotheory.asp.

[32] IBM (2021a). Gradient Descent. [online] Ibm.com. Available at: https://www.ibm.com/think/topics/gradient-descent.

[33] Ruder, S. (2017). An overview of gradient descent optimization algorithms *. [online] Available at: https://arxiv.org/pdf/1609.04747.

[34] Kingma, D.P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. [online] arXiv.org. Available at: https://arxiv.org/abs/1412.6980.

[35] Bjorck, J., Gomes, C., Selman, B. and Weinberger, K. (2018). Understanding Batch Normalization. [online] Available at: https://proceedings.neurips.cc/paper_files/paper/2018/file/36072923bfc3cf47745d704feb489480-Paper.pdf.

[36] Fernando, J. (2024). Sharpe ratio: Definition, formula, and examples. [online] Investopedia. Available at: https://www.investopedia.com/terms/s/sharperatio.asp.

[37] Ankit, U. (2022). Transformer Neural Networks: A Step-by-Step Breakdown — Built In. [online] builtin.com. Available at: https://builtin.com/artificial-intelligence/transformer-neural-network.

[38] Kang, Q., Chen, E.J., Li, Z.-C., Luo, H.-B. and Liu, Y. (2023). Attention-based LSTM predictive model for the attitude and position of shield machine in tunneling. Underground space, 13, pp.335–350. doi:https://doi.org/10.1016/j.undsp.2023.05.006.

[39] Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. [online] arXiv.org. Available at: https://arxiv.org/abs/1412.3555.

[40] Financial Services (2023). Financial Services Review —Applications of Natural Language Processing (NLP) in Finance. [online] Financial Services Review. Available at: https://www.financialservicesreview.com/news/applications-of-natural-language-processing-nlp-in-finance-nwid-1132.html [Accessed 11 Mar. 2025].

[41] Canan Sevgili, Laudani, P., Parodi, A. and Chiumento, A. (2025). COVID-19 shut us down five years ago. Here's how its economic impact continues. Reuters. [online] 8 Mar. Available at: https://www.reuters.com/business/healthcare-pharmaceuticals/five-years-economic-impact-covid-19-lingers-2025-03-08/.

[42] Turner, J. (2023). Why did the global financial crisis of 2007-09 happen? [online] Economics Observatory. Available at: https://www.economicsobservatory.com/why-did-the-global-financial-crisis-of-2007-09-happen.