Faculty of Natural and Mathematical Sciences
Department of Informatics

King's College London
Strand Campus, London,
United Kingdom

7CCSMPRJ/7CCSMUIP

Individual Project Submission 2022/23

Name:	Muhammad Ismail Kamdar
Student Number:	K19009749
Degree Programme:	Data Science
Project Title:	"Predicting Bitcoin Prices: A Multi-Model Approach with Deep Learning and Sentiment Analysis"
Supervisor:	Stefanos Leonardos
Word Count:	13101

RELEASE OF PROJECT

Following the submission of your project, the Department would like to make it publicly available via the library electronic resources. You will retain copyright of the project.

☑ I **agree** to the release of my project

☐ I **do not** agree to the release of my project

Signature:

Date: August 14, 2023

Department of Informatics
King's College London
United Kingdom

7CCSMPRJ/7CCSMUIP Individual Project

"Predicting Bitcoin Prices: A Multi-Model Approach with Deep Learning and Sentiment Analysis"

Name: **Muhammad Ismail Kamdar**
Student Number: K19009749
Course: Data Science

Supervisor: Stefanos Leonardos

This dissertation is submitted for the degree of MSc in Data Science.

Acknowledgements

I would like to acknowledge and give my warmest thanks to my supervisor Dr Stefanos Leonardos, whose guidance and advice carried me through all the stages of my completing my project and writing my dissertation.
I would also like to give special thanks to my parents, Muhammad Hanif Kamdar and Asma Hanif Kamdar, and my family as a whole for their continuous support and understanding during my Masters.
Most importantly, I would like to thank Allah, for getting me through all the difficulties, and providing me with the knowledge and strength to tackle all the challenges head on.

Originality Avowal

I verify that I am the sole author of this paper, except where it has been stated otherwise. I grant the write to King's College London to make paper and electronic copies of the submitted work for purposes of marketing , plagiarism detection, archival, and to upload a copy of the work to Turnitin or another trusted source of plagiarism detection service. I confirm this report does not exceed 15000 words.

Abstract

Accurate Bitcoin price forecasting is becoming more and more crucial for traders and investors as the cryptocurrency sector continues to gain prominence and importance in the financial scene. This project presents an in-depth research into the price prediction capabilities of certain Deep Learning algorithms, specifically the Long Short-Term Memory (**LSTM**), Convolutional Neural Network (**CNN**), Gated Recurrent Unit (**GRU**) and Bi-Directional LSTM (**BiLSTM**) neural networks. The models presented in this reports utilize a diverse range of metrics pertaining to the market and transactional elements of Bitcoin. The research starts by exploring the metrics and features used in our models, including block size, transaction fees and total bitcoin in circulation among others. To enable the creation of reliable and precise prediction models, several datasets pertaining to different elements were obtained, spanning several years. The paper will then look into experimenting with Sentiment Analysis using data scrapped in the form of news articles and applying the sentiment to our prediction models and testing its performance. The LSTM neural network is known for its effectiveness with time series data, and its ability to capture temporal correlations and dependencies. The paper will go through the architecture design, hyperparameter tweaking and training procedures as we attempt to optimise the performance of our model. The use of both qualitative and quantitative measures are used, such as mean squared error and R-squared score to evaluate our model performances. The project explored different feature engineering and selection criteria and the effect of normalisation on the models predictive capabilities. Our research found some very promising results, showing that our neural networks can successfully learn and forecast changes in Bitcoin price by effectively learning and predicting any future trends.

The paper also explores how the predictive models may affect risk management and investing techniques in the bitcoin market. Additionally, it discusses the drawbacks of the suggested strategy and recommends possible directions for further study and development. In conclusion, this paper clarifies presents a comprehensive analysis of using LSTM neural networks and the use of sentiment analysis to predict Bitcoin prices. The work makes a significant contribution to the field of time series analysis and paves the way for future developments in predictive modelling in the dynamic environment of the cryptocurrency markets.

Nomenclature

Bi-LSTM	Bi-Directional Long Short-Term Memory
CNN	Convolutional Neural Network
EMA	Exponential Moving Average
FNN	Feed-Forward Neural Network
GRU	Gated Recurrent Neural Network
LMI	Linear Matrix Inequalities
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MSE	Mean Squared Error
MLP	Multi-Layer Perceptron
NLTK	Natural Language Toolkit
R2	R2 Score Metric
RNN	Recurrent Neural Network
SMA	Simple Moving Average
UTXO	Unspent Transaction Output
α	Learning Rate

Contents

List of Figures

List of Tables

1 Introduction

Bitcoin, a revolutionary decentralised electronic payment mechanism that utilises Blockchain technology, has completely the world of finance. It is a system which provides a guarantee of transaction transparency using a public ledger according to Nakamoto [1], where lists of transactions are stacked in blocks, and each transaction can be traced in the order of when the blocks were created. Unlike traditional markets, Bitcoin has not yet matured in the sense of market stability and has led to extreme volatility as the majority of its trading volume is based on speculation especially considering it does not have any intrinsic value. The ability to forecast trends and predict future prices of cryptocurrency has gained increased importance as many aim to better understand the market volatility to invest in these new high return assets. In recent years, Bitcoin has risen in the world as an acceptable form of payment. However, the cryptocurrency market can be extremely volatile and complex, we require a smart prediction model capable of analysing the temporal patterns and discrepancies in the cryptocurrency market. In order to successfully do that, we will explore various deep learning architectures and their use cases in order to find a model that is suited for our data and capable of making a series of accurate predictions by integrating a variety of market metrics and technical features.
The metrics collected include a series of technical indicators obtained from Blockchain transaction data for Bitcoin, such as hash rate, transaction fees and block sizes among others which will be discussed as we delve further into this paper. Furthermore, the paper will discuss the effects social media and news has on the cryptocurrency price market, by harnessing the capabilities of sentiment analysis. Sentiment analysis is a proces that draws on natural language processing, in order to assign polarity ratings to segmented text which in our study includes news articles and tweets referencing Bitcoin. The polarity ratings range from -1, which signifies a negative sentiment, to 1, which denotes a positive sentiment. The sentiment analysis will be integrated into the price model in order to improve its performance.
The rise in importance of cryptocurrencies as financial assets served as the impetus for this study's topic selection. The need for strong and accurate price prediction has increased, making it crucial for investing strategies.

1.1 Objective

- Application of LSTM, GRU, Bi-LSTM and CNN techniques for time series Bitcoin price predictions
- Model trained on technical Blockchain indicators instead of just price features
- Evaluate the effects of adding Sentiment to Price Prediction models.
- Evaluating performance using MAPE and R2 Score metrics

There are 2 main categories of prediction models used in throughout; classification and regression. Classification models aim to predict the trends of the price, whether the

price will rise or fall while regression models aim to predict the price itself, this study focused on regression based models. In order to implement a solution using deep learning networks, which are known their capacity to understand complex patterns, this paper looks to give an in-depth analysis on the design of the network architecture, collection and preprocessing of the data used to train and test the models and the fine-tuning of the hyper-parameters in hopes to optimise the performance. The developed models were tested extensively using specifically designed experiments that were used to compare performances of different models, and their hyperparameter configurations using quantitative and qualitative metrics such as Mean Squared Error, R Squared Score and Mean Absolute Percentage Error.

This paper will start by discussing the background research pertaining to different types of deep learning architectures studied and their uses in existing solutions relating to price prediction in financial markets. Moving on the report will focus on the methodology used for the implementation of our prediction models, starting with data collection and processing criteria, where we employ exploratory data analysis and discuss reasoning behind feature selection. Further on, the paper will discuss the more technical aspects of the implementation of the different models aimed at price prediction with and without sentiment data. The results section details the evaluation criteria for our models and the experiments developed to test the model performance giving an in-depth review of the model results. Finally, concluding with a review of the paper in form of objectives achieved and the scope for improvements and future work the area. A list of references can be found at the end of the paper pertaining to the existing works studied in order to develop our models.

2 Background

The Background Chapter will outline and explain the various concepts used in this study and will discuss on how they relate to the objective of this study. This chapter contains sections aimed to improve understanding of Deep Learning, a subcategory of Machine Learning. Machine learning refers to the variety of algorithms that allow systems to learn without being programmed explicitly. Deep Learning uses multiple layers of neurons, more commonly known as a perceptrons. These neurons are capable of receiving inputs and applying weights to the inputs, before using an activation function to produce an output. The neurons are layered together in stacks to create Artificial Neural Networks (ANNs).

2.1 Artificial Neural Networks (ANNs)

One of the most widely used prediction models due to their data driven and self adaptive approaches. Input, hidden, and output layers are the three different types of layers that make up an artificial neural network's architecture. The input layer is created using a number of perceptrons, which corresponds to the number of features in the training data. The data is received into the input layer before being passed on to the hidden layer. The hidden layers allow the models to learn the trends and patterns in the data and are able to extract the relevant features from the input. Each hidden layer is made up of multiple perceptrons, the number is chosen in order to create a balance between performance on training data and generalisation of the model, as too many neurons can lead to a model that is overfitted. The output layer or the prediction layer is the last layer where the number of perceptrons depend on the task at hand, for regression tasks an output layer will have only 1 perceptron as it aims to predict a single continuous numerical value, while for classification tasks the number of perceptrons in the final layer reflect the number of distinct classes that are in the prediction space. ANNs can include a number of hidden layers that enable them to recognise complex hierarchical patterns in data, capable of solving challenging tasks in the field of natural language processing, speech recognition and image recognition. ANNs are capable of using feature representation learning, whereby the deep neural networks is able to extract hierarchical patterns from the data. The performance of artificial neural networks depends on a number of factors;

- Data Quantity and Quality
- Training Parameters
- Network Architecture
- Optimisation Algorithms

2.1.1 ANN Training

The fundamental idea behind ANNs is that they are capable of learning from data through training. In order to learn to recognise and generalise patterns outside of the

training set, ANNs iteratively update their internal parameters based on patterns found in the input data. Their effectiveness in tasks such as pattern recognition, classification, regression, and sequence modelling has been boosted by their capacity to learn and adapt to new information. The data is first split into training and test data, and fed into our initial model. The data is in the form of a set of features which are the metrics used to predict price in our example, and a target value, which is the value our model is trying to predict. The training consists of a process known as Feedforward pass where, the data flows through each layer to the output layer. Initial weights are added to the data to build a weighted sum at each connection from the input layer to the hidden layer neuron. The weighted sum then passes through activation functions which are used to add non-linearity to our network 1. Non-linearity is crucial as it allows our network to learn non-linear relationships, which makes the network capable of solving complex tasks without being restricted by linear boundaries. Without the activation functions the result would simply be the linear combination of inputs which is a function of polynomial degrees one. Real world data normally include complex patterns which cannot be modelled using a linear relationship between inputs and outputs.

Figure 1 shows the training for a single neuron where the weighted sum $z = \sum_i wi.xi+b$

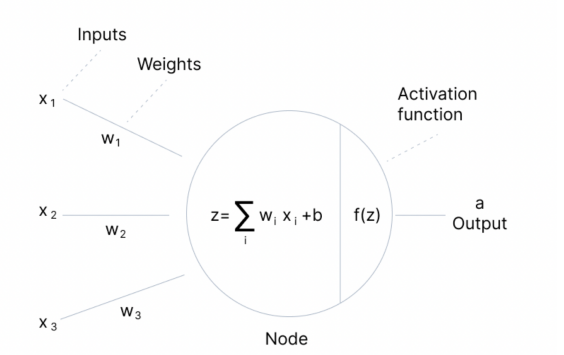

Figure 1: Neural Network Training with Activation Function

is calculated and passed through an activation layer $f(z)$, to get an output. With enough data and an appropriate architecture, neural networks have the ability to approximate any continuous function.

2.1.1.1 Activation Functions There are a variety of activation functions which are used across different use cases and tasks. The following is a list of some of the most widely used functions in ML:

ReLU: $f(x) = max(0, x)$ the Rectified Linear Unit is a non-linear activation function which is known to work well with non negative features

Leaky ReLU: $f(x) = max(ax, x)$ Leaky ReLU is a version of ReLU which instead of have a flat 0 function for $x < 0$ it applies a small linear weight to the x value thereby adding a small slope for negative x values.

Tanh: $f(x) = (2/(1 + e^{-2x})) - 1$ Hyperbolic Tangent function adds non-linearity to the cells, by compressing the values between -1 and 1.

Softmax Softmax function is commonly used for the output layer for multi-class classification tasks, as it maps raw output to class probabilities.

Sigmoid Function $\sigma f = 1/(1 + e^{-x})$ similar to the Tanh function, maps the output between 0 and 1 and is useful when the output needs to be limited to a specific range

At the end of the day there is no perfect activation function which can fit every model. The selection of the activation function is based on the characteristics of the data set and task at hand. A combination of activation functions can also be used with different function used in the hidden layer and output layers. Our study will make use of Tanh and ReLU functions dominantly as they proved to be the best performing ones after trial and error.

2.1.2 Network Architectures

Neural network architectures include a wide range of sophisticated approaches designed to address particular problems and extract useful insights from various data kinds. With their input, hidden, and output layers, Feedforward Neural Networks (FNN), also known as Multi-Layer Perceptrons (MLPs), set the standard among these architectures and are effective at performing regression, classification, and pattern recognition tasks. The use of convolutional layers to automatically extract hierarchical features and layer pooling, which refers to the downsampling of the dimensions of our input data while preserving crucial data, also known as dimensionality reduction makes Convolutional Neural Networks (CNNs) suitable for processing video and picture data.
Similarly, Recurrent Neural Networks (RNNs) excel when dealing with sequential data containing temporal dependencies, by utilizing loops to retain crucial information from previous stages or time steps, allowing them to use a window of previous training samples to make predictions. This makes them ideal for tasks like Natural Language Processing (NLP), time series predictions like financial market price prediction, and speech recognition. A specialized variant of the Recurrent Neural Network, the Long Short-Term Memory (LSTM) network, excels at adapting to long-range dependencies and addresses the vanishing/exploding gradient issues, proving useful for language translation and sentiment analysis, as well as time series forecasting. Gated Recurrent Unit (GRU), another variant of the RNN , offers similar performance using fewer number of parameters. Traditionally, RNN attempts to remember all previous information which can lead to problems managing the long term dependencies, the use of GRU can help improve that as it contains components which are able to discard information that may not be useful and only retain crucial bits, similar to the CNN as discussed above. Additionally, the Transformer architecture, which was first developed for natural language processing, includes attention mechanisms for non-sequential feature extraction, making it ideally suited for jobs like text summarization and machine translation. Last but not least,

Capsule Networks (CapsNets) offers a unique method for handling hierarchical connections in data and are now being used for image recognition applications.
These diverse neural network architectures cater to distinct data types and tasks, allowing researchers and practitioners to discerningly choose the best appropriate model for their particular problem domain, including price prediction in financial markets. The use cases of the above network architectures was studied and by relating to our task the choice of network architecture was made.
As our area of research is data analysis in cryptoeconomics, this paper will focus mostly on the use of LSTM, CNN and GRU networks in order to leverage their power over times series analysis to build a robust Bitcoin price prediction model.

2.1.3 Model Optimisation

During the training process described in the ANN section, the ANN model's biases and weights are adjusted to minimize the discrepancy between the predictions and the actual target values which are provided by the training data. Through this optimization process we aim to discover the optimal set of parameters that best fit the provided training data and generalize well to our unseen test data.
The process is driven by different optimization algorithms, which include the popular Gradient Descent algorithm. Gradient Descent updates the model's parameters aiming to reduce loss by travelling in the opposite direction of the gradient of the loss function. Using variations like Stochastic Gradient Descent (SGD) and Mini-batch Gradient Descent, we may accelerate convergence and avoid local minima. Additional developments include Momentum, Adagrad, RMSprop, and Adam, which adaptively modify the learning rates for each parameter to improve convergence and robustness. The models featured in this study will be dominantly focused on the Adam optimiser. According to the historical first-order moments (mean) and second-order moments (variance) of the gradients, the Adam algorithm adjusts the learning rate for each parameter.
The choice of optimization algorithm depends on the problem's nature, neural network architecture, and dataset characteristics. The use of trial and error with different techniques is common to find the most suitable optimization strategy. In order to increase the model's performance on unseen test data, the generalisation of the neural network model can be improved by applying regularisation techniques as L1 and L2 regularisation, dropout, and batch normalisation. The dropout layer essentially works to generalise the model, by dropping out a percentage of the perceptrons in the model. By reducing the reliability on a single perceptron, the model avoids over specialising in patterns in our training data which can result into overfitting, thus increasing the robustness and versatility of the model.

3 Related Work

This section will look at the existing work done in the field of price prediction. We will discuss the architectures and data used in these papers and how they relate to our approach. There has been a number of papers written regarding price prediction using machine learning ranging from simple linear regression models to more advanced deep learning models. Different approaches were used past researches in terms of choice of data for prediction and choice of machine learning models, all of which can result in different results. Each configuration choice has a different result on the performance of models and essentially depends on the type of questions you want answered. The following are some of the papers reviewed in order to gain some insight and set an evaluation benchmark.

3.1 Literature Review

Junwei Chen's paper 'Analysis of Bitcoin Price Prediction Using Machine Learning'[2], proposes the idea of using a random forest regressor to predict bitcoin prices. In the research he claimed that the use of random forest regression resulted in similar accuracy to different works using LSTM models, even higher in some stages. However, the model was unable to predict prices, that were unseen in bitcoin history as quoted in his paper; "Although random forest regression has the disadvantage of being unable to predict the results that did not appear in the training samples. For example, when the price of Bitcoin broke the record high, random forest regression could not provide a higher price result than the previous historical high."[2]
Gurupradeep, Harishvaran and Amsavalli's paper 'Cryptocurrency Price Prediction using Machine Learning'[3] used a similar random forest linear regression model using grid search as an optimising tool. They used the random forest classifier to fit their training set which consisted of price features of bitcoin which included high, low, opening and closing prices to train the data.
'Bitcoin price prediction using Deep Learning Algorithm' by Rizwan, Javed and Narejo [4], implemented a 4 Layer GRU deep learning model to predict bitcoin price trends, and resulted in excellent results of 90% accuracy. However they did conclude, as quoted from the paper 'Selected features: Low, High, Close and Open can't be enough to predict the Bitcoin value, as various factors, including social media responses', which does make sense. Many of the research works done in the field have used similar features to predict price which got us to focus on more technical blockchain features in our study.
Ping Feng's paper, 'A hybrid ARIMA and support vector machines model in stock price forecasting'[5] proposed an innovative hybrid model where the combination of support vector machines and ARIMA models was used for predicting stock prices. ARIMA, (AutoRegressive Integrated Moving Average), time-series forecasting method that allows to understand the relationship of each variables to its past values. It is able to understand some of the non-linearity of data but is much better suited for linear relationships. Ping used this model to predict stock prices which generally have more stability unlike the

bitcoin market. Our study although did not explicitly use ARIMA due to our data being non-linear, it did take inspiration in the form of engineering features in the form of simple and exponential moving averages.
Rcik Dey, Saurabh Shukla, Sarthak Jasani and Hezal Lopes's paper 'Bitcoin Price Prediction Using LSTM'[6], works with LSTM models where they used a similar dataset to [3] which focused on just the price indicators. They produced a base LSTM model which performed well on unseen data, the interesting this we see is the use of a look back or time step which we adapted into our model as well, this allows the model to consider a number of training variables for each prediction. For example when predicting the price for 10th of November, using a time step value of 3, the model will use the data from 7, 8 and 9th of November for its prediction. It allows to leverage the sequential nature of the data. We will discuss this in a lot more detail in the 'Approach' section of this paper.
A paper that used a similar approach as proposed in our study was 'Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach'[7] by Seabe, Moutsinga and Pindza where they proposed the use of Bi-Directional LSTM, Gated Recurrent Unit and LSTM models for predicting prices of Bitcoin, Ethereum and Litecoin. The data used for the models developed was historical prices for the the cryptocurrencies which included price features of high, low, open and close prices. The paper concluded that the performance of all the models was very good, with Bi-LSTM proving the best out of the three. Our study will be using the results from this paper as a benchmark as we look to use similar models, however our models will focus on different features in the data, more focused on technical indicators.
Yan Li and Wei Dai proposed the use of Convolution Neural Network combined with a Long Short-Term Memory network for predicting the daily closing price of Bitcoin, in their 2019 paper 'Bitcoin price forecasting method based on CNN-LSTM hybrid neural network model'[8]. An innovative approach where the CNN model is used for extracting features and passing it on to the LSTM for training and forecasting. This research paper concluded with some very promising results where the CNN-LSTM model performance improved as compared to standalone CNN and LSTM models, the performance was evaluated using mean absolute error (MAE) and mean absolute percentage error (MAPE).
The use of sentiment in price prediction has been a topic of investigation in the last few years, Franco Valencia's paper 'Price Movement Prediction of Cryptocurrencies Using Sentiment Analysis and Machine Learning', combined Support Vector Machines, Multi-Layer Perceptrons and Random Forests models with twitter data set in order to add sentiment. The study conclude that in order to be able to harness the power of sentiment analysis for ML models, the data collected must be of high quality without any noise or bias which is common due to promotional and advertisement campaign regarding cryptocurrency on Twitter. The models in this study were not specialised in terms of adapting to temporal data which goes to show that models such as LSTM may be more suitable for this task.
Sara Rosenthal from IBM Research's paper on 'Sentiment Analysis in Twitter', was used to better undeerstand how twitter data can be harnessed in order understand the concepts of Sentiment analysis and polarity subjectivity. Allowing competitive experiments

on using NLP techniques for sentiment prediction, the research concluded that the participants had most difficulty with twitter data which consisted of sarcastic context.
Another paper 'Bitcoin price change and trend prediction through twitter sentiment and data volume' by Jacques Vella Criterien, used similar deep learning models to Seabe [7], in order to classify the Bitcoin price trends using a voting classifier. The use of sentiment analysis with a time lag was quite interesting as it pointed out that changes in sentiment may not always reflect changes on price instantly and can take time to impact if it does at all. The study concluded with impressive classification results of over 70%.
Another interesting approach was to test other factors that may influence price. Kristoufek's paper 'BitCoin meets Google Trends and Wikipedia: Quantifying the relationship between phenomena of the Internet era' [9] where he attempted to model the relationship between Bitcoin prices with search queries on google and wikipedia. The effects trend changes was compared to check if there might be any understandable correlation between them.

4 Approach

This section will delve into the technical aspect and detail of our study, starting with the chosen models and objectives, moving on towards data collection and processing. Further on the paper will look at the model architecture implementation.

4.1 Objectives, Specification and Design

As mentioned in the Introduction section, the main objectives of this study were as follows;

- Application of LSTM, GRU, Bi-LSTM and CNN techniques for time series Bitcoin price predictions
- Model trained on technical Blockchain indicators instead of just price features for the year 2021.
- Evaluate the effects of adding Sentiment to Price Prediction models using news article data sets.
- Evaluating performance using MAPE and R2 Score metrics

After thorough research on different model architectures, our study was focused on developing the LSTM, Bi-LSTM, CNN and GRU networks. In order to test our model, we gathered a data set that was away from the trivial data features used by many past researches. Instead of focusing more on price related factors, our study focuses on the use of diverse set of technical indicators. We were able to derive these indicators from various market metrics as there are many other factors that can affect the price of crytocurrencies. As the majority of Bitcoin trading value is speculation based, it is important to account for features that can affect that, social media has been known to be a major factor in influencing speculation based trading trends. Our study explores the use of social media data in order to predict the prices. The following sections will contain snippets of code that are useful to demonstrate the implementation, the complete code can be found in the appendix section.

4.2 Data Collection

Data collection and pre-processing has in the past not been given much attention in terms of importance in creating machine learning models as models are only as good as the data they are trained on. A major time-consuming part of model developing relates to acquiring accurate and comprehensive data sets and processing it in a way that is suitable for chosen models. Its significance cannot be overstated because the correctness, validity, and reliability of any conclusions, insights, or forecasts produced from the data directly depend on the quality, quantity, and relevance of the data that were gathered. Our data set consisted of a whopping 19 features for regular price prediction and 20 features for price prediction with sentiment. The features used in the study can be

found in Table 1, along with a description of each feature.

Data Features		
N	Feature	Description
1	blocks-size	Size of blocks in the Bitcoin Blockchain
2	avg-block-size	Average size of blocks in the Bitcoin Blockchain
3	n-transactions	Number of Bitcoin transactions on the day
4	n-transactions-excluding-popular	Same as n-transactions, excluding transactions considered as popular
5	n-transactions-total	Total number of transactions to date
6	hash-rate	Computing Power of the Bitcoin network for the day
7	difficulty	Measure of difficulty to find a new block
8	transaction-fees-usd	Fees in USD to include transactions in a block
9	n-unique-addresses	Number of unique addresses that interacted with Bitcoin Blockchain for the day
10	my-wallet-n-users	Number of users on the Bitcoin Blockchain to date
11	utxo-count	Count of unspent transaction outputs (UTXOs) in the Bitcoin Blockchain.
12	estimated-transaction-volume-usd	Total value of Bitcoins transactions in USD
13	trade-volume	Volume of Bitcoin traded
14	total-bitcoins	Total number of Bitcoins in circulation
15	Sentiment Polarity	Average Sentiment Polarity for the day
16	EMA10	Exponential Moving Average Price Average for 10-day period
17	SMA10	Simple Moving Average Price Average for 10-day period
18	EMA50	Exponential Moving Average Price Average for 50-day period
19	SMA50	Simple Moving Average Price Average for 50-day period
20	Lagged Price	Closing Price of previous day

Table 1: Features used for Bitcoin Price Prediction

The data set above for collected for the year 2021 from January 1st to 31st December, as this window contained a lot of trends in regards to price falling and rising and seemed to be an interesting window to analyse. The following sections contains the data extraction and collection process for our price, technical and sentiment features.

4.2.1 Bitcoin Price and Technical Metrics

Data Source: Blockchain.com
This section explains the code used for data extraction of features 1 to 14 mentioned in table 1. The price and technical historical data set was extracted from Blockchain.com using a customised python script, adapted from online sources in order to request and fetch data with the help of the Blockchain.info API. The code below can provide a bit more insight in the collection process;

```
def loadPriceData(metrics , years : int):
df_all = pd.DataFrame()
for m in metrics:
    append_data = []
    for y in years:
        ts = datetime.datetime(
            y, 12, 31, tzinfo=datetime.timezone.utc).timestamp()
        print('https://api.blockchain.info/charts/'+m +
            '?timespan=1year&rollingAverage=24hours&
            format=csv&
            start='+str(int(ts)))
        df = pd.read_csv('https://api.blockchain.info/
        charts/'+m+'?
        timespan=1year&rollingAverage=24hours&
        format=csv&start='+str(
            int(ts)), names=['date', m], parse_dates=[0], index_col=[0])
        append_data.append(df)
    df_m = pd.concat(append_data)
    df_m.index = df_m.index.normalize()
    df_m = df_m.groupby([pd.Grouper(freq='D')]).mean()
```

Listing 1: Price and Technical Feature Extraction Script

Code above was adapted from the link below.
`https://github.com/pahurlocker/bitcoin-price-prediction/blob/main/data_producers/bitcoin_price_dataproducer.py`
The above code was adapted from the source mentioned at the end of this section. The above code takes in 2 parameters, a list of required metrics and a list of the years for which the data is requested. It then uses a nested loop to access data for each metric and for each year through the API, the API provides a csv file containing the requested data which is combined into a single list. The lists of features/metrics data are then concatenated into a single Pandas Dataframe, using the outer join merge function with date as a reference point. Figure 2 gives a small example from our data set which shows the structure. It should be notes that the figure is for reference only and does not contain all the features as the number of features were too many to be fit in one figure.

date	blocks-size	avg-block-size	n-transactions-total	hash-rate	difficulty	transaction-fees-usd	n-unique-addresses	n-transactions	my-wallet-n-users	utxo-count
2021-01-01	319114.0485	1.352232664429530	601577750.0	137764027.0294170	18599593048299.0	1455398.207853410	609741.0	258080.0	63467248.166666700	69524615.25
2021-01-02	319315.580633	1.3956372450331100	601835959.0	139613208.60028200	18599593048299.0	2504284.3156627500	711719.0	297111.0	63544347.197916700	69314224.67619050
2021-01-03	319526.405445	1.2886939240506300	602133929.0	146085344.09830800	18599593048299.0	2893753.6352049900	812749.0	359116.0	63636098.16666670	69291381.15873020
2021-01-04	319729.832857	1.3188061907514400	602493027.0	159954205.87979300	18599593048299.0	3516375.453386650	851667.0	373734.0	63738746.864583300	69288106.71111110
2021-01-05	319957.982033	1.2927130828025500	602865747.0	145160753.31287600	18599593048299.0	3677640.8004324100	845343.0	354091.0	63846852.59375	69333357.2031746
2021-01-06	320160.890674	1.293667920903960	603220075.0	163652569.02152200	18599593048299.0	4477049.4851396100	924851.0	397384.0	63949725.78125	69408890.88888890
2021-01-07	320390.014476	1.269546125	603618745.0	155331251.95263100	18599593048299.0	4701567.105017460	940647.0	401744.0	64055086.09375	69511478.17857140
2021-01-08	320603.341523	1.295011067567570	604020433.0	136839436.24398500	18599593048299.0	5435433.863222620	883788.0	358526.0	64158977.666666700	69705712.35555560
2021-01-09	320794.958819	1.2609308633093500	604378746.0	133808024.52210500	19365166707094.80	4571160.4769916700	760179.0	321389.0	64252059.54166670	69891240.34523810
2021-01-10	320970.191524	1.282691371069180	604699941.0	162879650.9247730	20607418304386.0	3643194.912175910	768365.0	331865.0	64339124.78125	69953307.8015873
2021-01-11	321174.048974	1.3372515000000000	605031340.0	157757649.32336500	20607418304386.0	4563809.325475650	801285.0	333958.0	64428763.97916670	69944315.57142860
2021-01-12	321380.113784	1.3242120472973000	605365473.0	151611247.4016760	20607418304386.0	5843153.716853880	796457.0	336627.0	64519249.739583300	69994049.77777780
2021-01-13	321576.046068	1.2648656644736800	605701559.0	155708848.68280200	20607418304386.0	4725847.947130550	772596.0	319167.0	64602336.91666670	70037409.28968250
2021-01-14	321768.18095	1.2964830204081600	606020714.0	150586847.08139400	20607418304386.0	5760122.5729426	802110.0	338809.0	64688608.552083300	70078992.11904760
2021-01-15	321958.923378	1.3152658064516 10	606359799.0	158782049.6436470	20607418304386.0	5500740.880559790	808283.0	344002.0	64770134.79166670	70193177.66349210
2021-01-16	322162.948202	1.3197683197278900	606704771.0	150586847.08139400	20607418304386.0	3324821.305439300	733398.0	308461.0	64848253.57291670	70242617.8968254
2021-01-17	322356.7137	1.3245906842105300	607012869.0	136245242.59745200	20607418304386.0	2359708.0570458600	642730.0	271874.0	64921582.25	70261836.50793650
2021-01-18	322532.980588	1.3425268551724100	607284053.0	148538046.440831	20607418304386.0	3085563.262248470	720228.0	313816.0	65001504.30208330	70157773.66666670
2021-01-19	322727.574032	1.278255315436240	607597959.0	152635647.72195700	20607418304386.0	3635331.922466610	757920.0	325083.0	65092489.78125	70189629.36904760
2021-01-20	322917.077201	1.2771157234042600	607923111.0	144440445.1507050	20607418304386.0	3281456.6810408200	755823.0	314890.0	65180877.66666670	70261601.26587300

Figure 2: Base Price and Technical Indicators Data set Example

4.2.2 Sentiment Data

Data Source: Kaggle, CryptoNews.api The data collected for sentiment analysis was acquired from multiple sources for experiments. We were able to obtain 2 separate data sets from Kaggle containing tweets referencing bitcoin and cryptocurreny for different time frames. The twitter data set initially consisted of features such as date of tweet, tweet text, username and hashtags. Figure 3 shows an example of the twitter data set layout using a subset of our data. It is to be noted that the figure does not reflect all the features in our dataset, due to a large number of feature variables.

A second data set was obtained which contain articles regarding cryptocurrencies, this was obtained via a Github repository, that obtained the data using CryptoNews API. CryptoNews.com is well known for their coverage of global Blockchain news. It contains all articles regarding crytocurrencies from various reputable sources such as Forbes, Bloomberg Markets and Finance, Fox Business and Coindesk. The news article data set consisted of features such as source, date of article, article title, summary and article text. The original data set obtained through CryptoNews originally consisted of just the article summaries, however the data had been expanded to include the whole article text using Python's Newspaper library. The data set consists of articles from the start of January 2021 to November 2021. Figure 4 shows an example of the articles data set layout using a subset of our data. It is to be noted that the figure does not reflect all the features in our data set, due to a large number of feature variables.

Link for articles Data set:

`https://github.com/pahurlocker/bitcoin-price-prediction/tree/main/data`

For our Sentiment Analysis we chose to use our news articles data set as during further processing, the twitter data set contained too much noise as it is an open platform for people to express themselves, which sometimes may not be completely accurate representation of the facts as the language used may not always be context-dependent and

user_description	user_created	user_followers	user_friends	user_favourites	user_verified	date	text	hashtags
Biz Consultant, real estate, fintech, startups...	2009-04-26 20:05:09	8534.0	7605	4838	False	2021-02-10 23:59:04	Blue Ridge Bank shares halted by NYSE after #b...	['bitcoin']
😎 BITCOINLIVE is a Dutch platform aimed at inf...	2019-10-17 20:12:10	6769.0	1532	25483	False	2021-02-10 23:58:48	😎 Today, that's this #Thursday, we will do a "...	['Thursday', 'Btc', 'wallet', 'security']
IM Academy : The best #forex, #SelfEducation, ...	2014-11-10 10:50:37	128.0	332	924	False	2021-02-10 23:54:48	Guys evening, I have read this article about B...	NaN
I will post a lot of buying signals for BTC tr...	2019-09-28 16:48:12	625.0	129	14	False	2021-02-10 23:54:33	$BTC A big chance in a billion! Price: \487264...	['Bitcoin', 'FX', 'BTC', 'crypto']
Co-founder @RENJERJerky | Forbes 30Under30 | I...	2016-02-03 13:15:55	1249.0	1472	10482	False	2021-02-10 23:54:06	This network is secured by 9 508 nodes as of t...	['BTC']
...	...	...	...	...	...	...	...	...
#Bitcoin | Forever Alone Crypto Trader 🐱 | Ana...	2022-11-04 10:18:11	674	1789	27466	False	2023-01-06 17:46:35	Bitcoin squeeze is SUPER TIGHT, which way will...	['BTC', 'bitcoin', 'Crypto', 'cryptocurrency',...
My backpack has jets. Reformed Necromonger	2008-04-02 21:48:47	79	454	125	False	2023-01-06 17:46:29	Closed #BTC short at 16725. Missed my long pla...	['BTC']
UP or DOWN...\n.\n.\n.\n.\nPrice matters NOT.	2022-07-24 04:50:18	532	1	0	False	2023-01-06 17:46:22	#Ethereum price update: \n\n#ETH $1263.59 USD\...	['Ethereum', 'ETH', 'Bitcoin', 'BTC', 'altcoin...
Tweets the current price of #bitcoin every 5 m...	2022-10-20 07:10:38	83	7	9	False	2023-01-06 17:46:20	1฿ = $16814.7 -0.07%🔻\n\nDetails:\nChange: 🔻-1...	['bitcoin', 'btc']
💰faucetpay 👉 \nhttps://t.co/eHqMw9QdFx\n\n	2022-10-13 05:41:53	14	10	0	False	2023-01-06	Earn crypto by playing fun games online.\nGet ...	['faucet', 'cointiply', 'BTC']

Figure 3: Tweets Data set Example

contains sarcasm which led to inaccurate classification of sentiments, this was emphasized d in the paper by Rosenthal [10], where she claimed the sarcastic language in tweets can skew sentiment predictions. Even though the Bitcoin market is said to be speculative, the twitter community is much highly volatile as there are attempts to manipulate the market, which may distort sentiment and our data set may not be based on genuine information. An example of this is a rise in Non-Fungible Token (NFT) giveaways, our twitter data set, consisting of up to 4 million tweets, consisted a high level of promotional content for giveaways and advertisements, leaving very few tweets which are actually relevant for our analysis. In contrast the articles collected are all from reputable sources and contain expert opinions backed up by hard facts which are more reliable. The use of articles data set was thus chosen to be the right option going forwards.

4.2.3 Exploratory Data Analysis

After collection of data, we performed the meticulous process of cleaning our data in order to avoid any inconsistencies and missing value problems, as we prepared the data to be fit into our models. Using Python's powerful Pandas library the data was cleaned of any and all missing values by removing the rows itself. Next step involved exploring the data to gain insights on our data features and understand the relationship to our target variable, price of bitcoin. This section will shed some light into the reasons behind

date	datetime	news_url	title	text	source_name	sentiment	type	article_title	article_text
2021-01-02	Sat, 02 Jan 2021 13:00:15 -0500	https://dailyhodl.com/2021/01/02/crypto-analys...	Crypto Analyst Michaël van de Poppe Unveils Bi...	Widely-followed trader and crypto strategist M...	The Daily Hodl	Neutral	Article	Crypto Analyst Michaël van de Poppe Unveils Bi...	Widely-followed trader and crypto strategist M...
2021-01-02	Sat, 02 Jan 2021 10:04:44 -0500	https://cryptodaily.co.uk/2021/01/btc-usd-test...	BTC/USD Tests 31609 Technical Resistance: Sall...	Bitcoin (BTC/USD) rampaged higher early in tod...	Crypto Daily	Positive	Article	BTC/USD Tests 31609 Technical Resistance: Sall...	Bitcoin (BTC/USD) rampaged higher early in tod...
2021-01-02	Sat, 02 Jan 2021 11:00:38 -0500	https://www.coindesk.com/more-aussies-back-bit...	More Aussies Back Bitcoin, the Underdog	Aussie resilience (and bitcoin) binds us toget...	Coindesk	Positive	Article	More Aussies Back Bitcoin, the Underdog	To me, this is a classic example of that iconi...
2021-01-02	Sat, 02 Jan 2021 11:32:11 -0500	https://cointelegraph.com/news/how-massive-bit...	How massive Bitcoin buyer activity on Coinbase...	The price of Bitcoin surged past $32,000 as bu...	Cointelegraph	Positive	Article	How massive Bitcoin buyer activity on Coinbase...	Coinbase has seen a large spike in buyer activ...
2021-01-02	Sat, 02 Jan 2021 11:47:14 -0500	https://bitcoinist.com/3-key-reasons-why-bitco...	3 Key Reasons Why Bitcoin Price Just Exploded ...	Bitcoin moves past $23,500 for the first time ...	Bitcoinist	Positive	Article	3 Key Reasons Why Bitcoin Price Just Exploded ...	Bitcoin moves past $33,500 for the first time ...
...	...	...	...	...	...	...	...	...	...
2021-07-19	Mon, 19 Jul 2021 06:32:03 -0400	https://cryptopotato.com/nearly-70-of-would-ho...	Nearly 70% of Would HODL Bitcoin if it Drops B...	More than 65% of survey participants said they...	CryptoPotato	Negative	Article	Nearly 70% Would HODL Bitcoin if it Drops Belo...	Nearly 70% of current bitcoin holders are prep...
2021-07-19	Mon, 19 Jul 2021 06:36:00 -0400	https://coinidol.com/bitcoin-31000-downtrend/	Bitcoin Finds Support Above $31,000; Can It Re...	Bitcoin (BTC) price has been trading above $31...	Coin Idol	Neutral	Article	Bitcoin Finds Support Above $31,000; Can It Re...	Jul 19, 2021 at 10:36 // News\n\nCoin Idol Aut...

Figure 4: News Articles Dataset Example

feature selection of specific features. Using Python's **Matplotlib** library we started by creating a visualisation of the original time series data set of all the features collected. Figure 5 shows a simple visualisation of the original time series data set.

Next up we created visualisations of our features as compared to price in order to observe any trend similarities in temporal patterns. In order to explore temporal similarities, we first needed a way to normalise our data set, as without normalisation it can be difficult to observe trends as we have a variety of features, all of which have different scales. For example if we compare the trends for Bitcoin Price and Transaction Fees in USD, the Bitcoin chart will completely dominate our visualisation as it can range up to 50000 USD while transaction may have a max of 1000 USD. Using a MinMax Scaler we were able to normalise the data which essentially maps the entire data features to a certain range. The formula for the MinMax Scaler is as followed;

$$m = \frac{(x - xmin)}{(xmax - xmin)}$$

x denotes the original value of our feature, while xmin and xmax denote the minimum and maximum values for that feature respectively, in our data set. Now that the data has been normalised, we can easily plot our features against the price of bitcoin for the time-frame. The following are some of the plots displaying the relationship of some of the features to Bitcoin price;

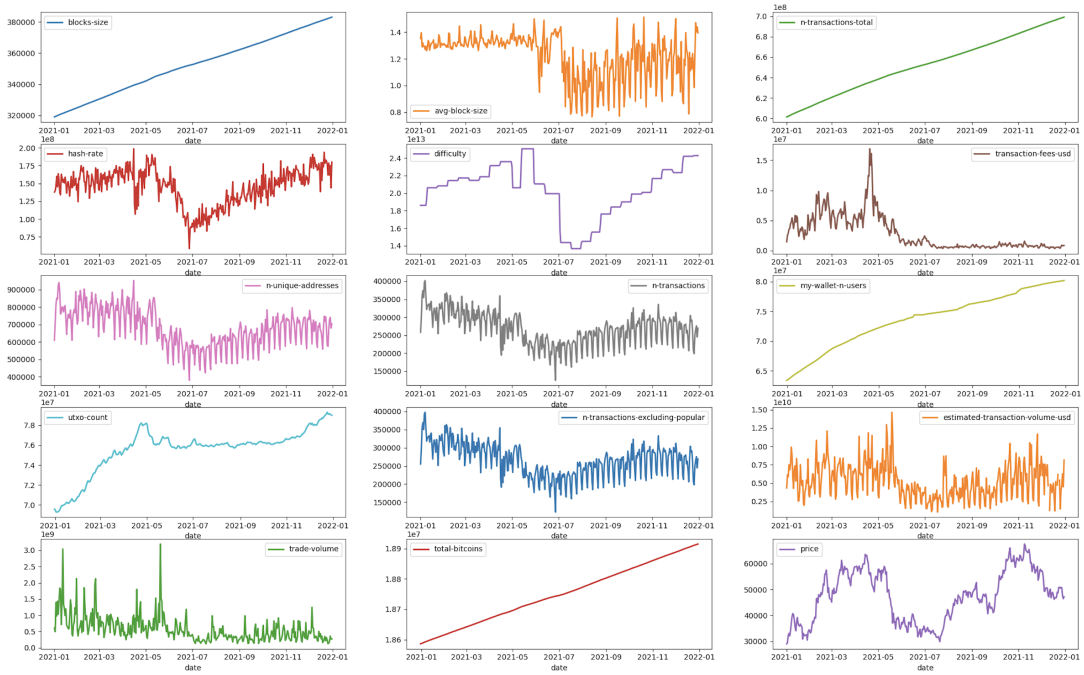

Figure 5: Visualisation of Original Time Series Data

In figure 6, we can see a vague correlation between the changes in daily transactions

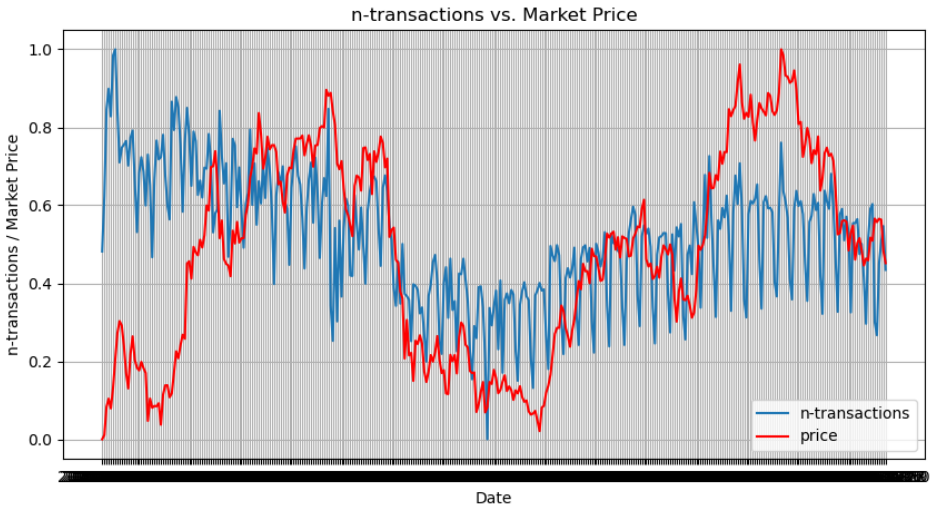

Figure 6: N-Transactions vs Price

and changes in price. The plot for transactions, has loads of spikes and falls but stays in a specific range. It seems to be vaguely following the trend movement of price especially in the middle section where price fell. It shows a level of correlation and potential impact on the target value. Logic dictates, an increase in transactions is a sign of increased demand, which based on economic principles will lead to an increase in the price. This may be directly or indirectly in the form of impacting other related factors such as market sentiment. As Bitcoin is based on Blockchain which employs the transparent ledger system where all transactions on the network can be viewed, a sudden increase in transactions can lead to speculative traders to buying into the market in hopes of high return.

Figure 7, seems to show a very strong correlation between hash rate and price. Hash rate

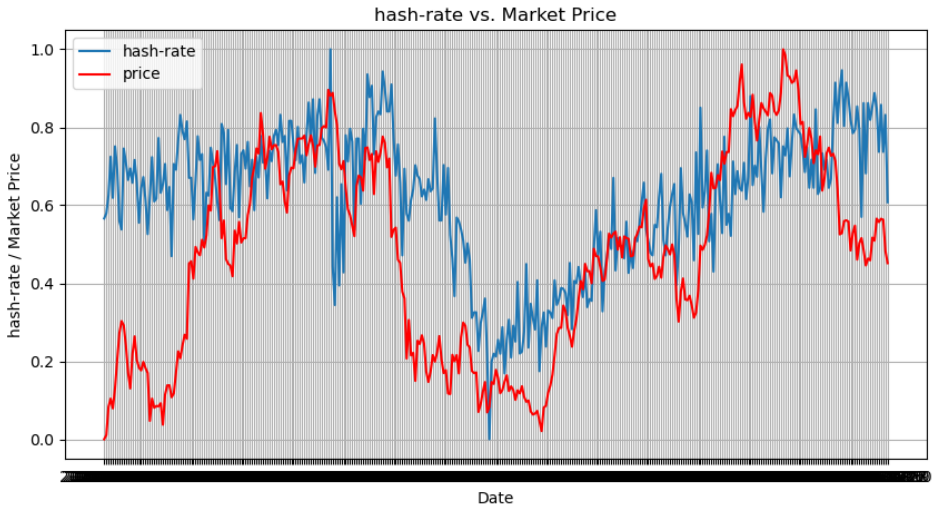

Figure 7: Hash Rate vs Price

refers to the computational strength of the network, the speed at which bitcoin transactions are being validated. Transaction validation is based on the solving of mathematical problems by individuals or groups known as miners. These miners are given a certain rewards for solving these problems which are more commonly known in cryptocurrency circles, as proof-of-work algorithms, this acts as an incentive for miners. Higher hash rate refers to high mining speed which can be speculatory indicator. In simpler terms, if an individual is being paid using a certain item, if the value of the item is expected to rise, the individual will want to increase their effort as the incentive is now higher, this can be observed in the plot where at the start, the hash rate went up which reflect more mining starting to work and price increased following it. This can further be seen when the drop of price led to the fall in hash rate as miners lost incentive. Its direct impact on price is still complex but it can definitely affect the sentiment in the market which

is known to be a strong explanatory variable in a speculation based market. The plot seems to confirm our theory and seems to have a delayed response to price which can be used in an LSTM model which looks at training data from both directions which can be said about Bi-Directional LSTM models.

In figure 8, we observe the plot of difficulty against the price over time. Difficulty

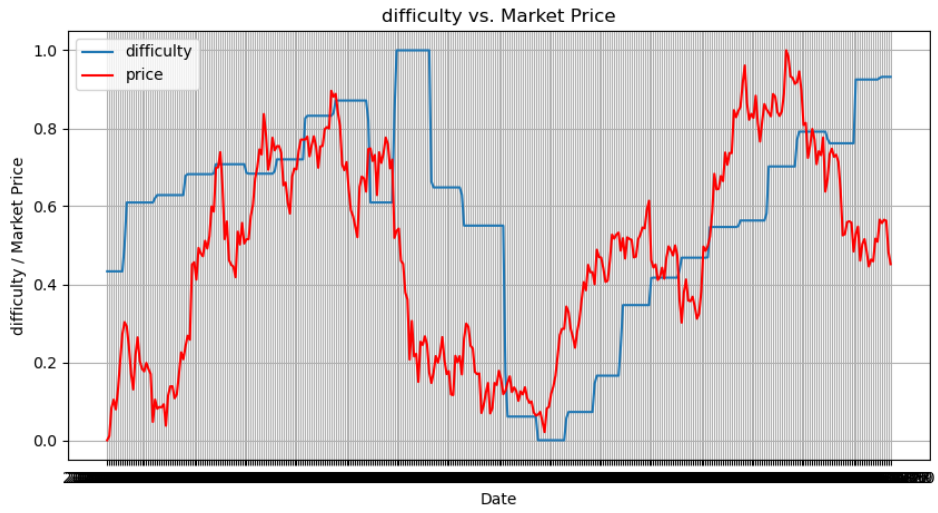

Figure 8: Difficulty vs Price

refers to, the level of toughness in mining new blocks. The level of difficulty is automatically adjusted every 2016 blocks according to our research on Blockchain. Transactions are all stored in blocks, where new blocks are added every 10 minutes. To ensure that this timetable in maintained, the difficulty level is adjusted in response to competition between miners, for example if there are a huge number of miners contributing computational power into solving the mathematical problems for validating transactions, the difficulty of these problems will be increased to ensure stability and vice versa. We can clearly relate this feature to hash rate as discussed above. Change in mining power can have a direct relation to difficulty. So by law of transitivity we can conclude that the difficulty will have an impact on price, which is plausible given the clear correlation on figure 8. However as the change of difficulty and hash rate seems to be following the same trend and have similar trends, dimensionality reduction concepts would suggest removing one of these variable in order to avoid overloading the model. We chose to retain both of these features in our model as even though they follow similar trends, the chart shows a difference of movement in the sense where the plot line for difficulty seems to move in the form similar to a step function and it was decided that this may add some interesting generalisation and relevance to our model.

Finally, looking at figure 9 we can see the number of unique addresses for each day

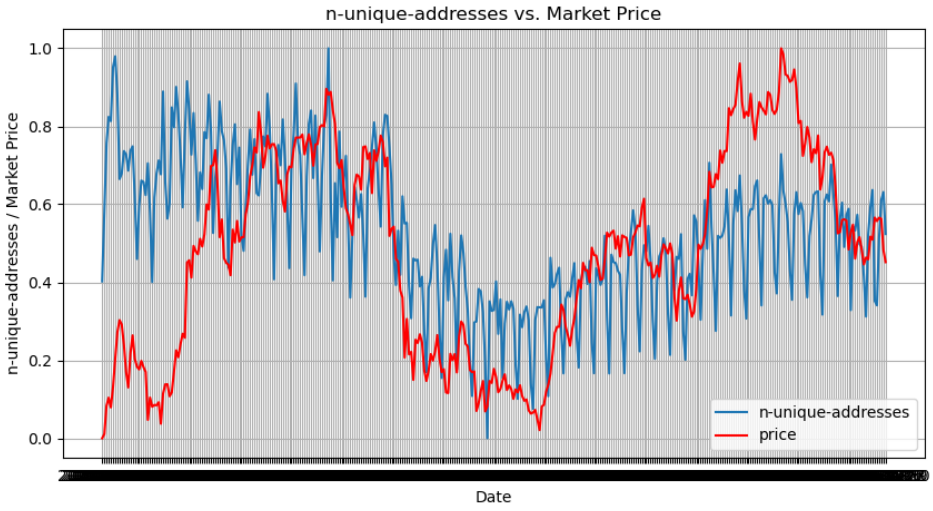

Figure 9: N-Unique Addresses vs Price

against price. This is an interesting feature, as it indicates the change in users adopting cryptocurrency. An increase shows us that the investors are confident as more individuals are currently participating in Bitcoin trade which indicates confidence in the market and can be a signal for speculative trading and should affect the price of Bitcoin as its use case increases.
Our next step for Exploratory Data Analysis was to create a Correlation matrix to increase our understanding of inter feature correlation and confirm our choice of features. Our correlation matrix was calculated for our data set features using the Spearman's correlation as it is suitable for non-linear data and is not affected by different scales of data as it is less sensitive to outliers. Using a combination of a correlation matrix and a heat map we were able to develop an aesthetic visualisation in figure 10, which gives us a different perspective to the hidden relationship in our data set.

Looking at the bottom row 'Price', we can observe the correlation of price with each of our features. On a cursory observation we can tell that all features have a positive correlation which indicates that an increase in the feature value tends to translate onto price as well. Hash rate and difficulty are the figures that are leading in correlation, as they have the highest correlation of 0.53 and 0.44 respectively, with price, the reasons for which we discussed in the section paragraphs above. Next, looking at UTXO count we can see a decent correlation of 0.4. To understand how that is possible requires us to understand what UTXO (Unspent Transaction Count) count is and how it works, a simple technical definition would be, UTXO is the part of bitcoin which is unspent after a transaction. This is basically similar to the change we receive when making cash transactions. Another way to describe this would be, in order to make payments or

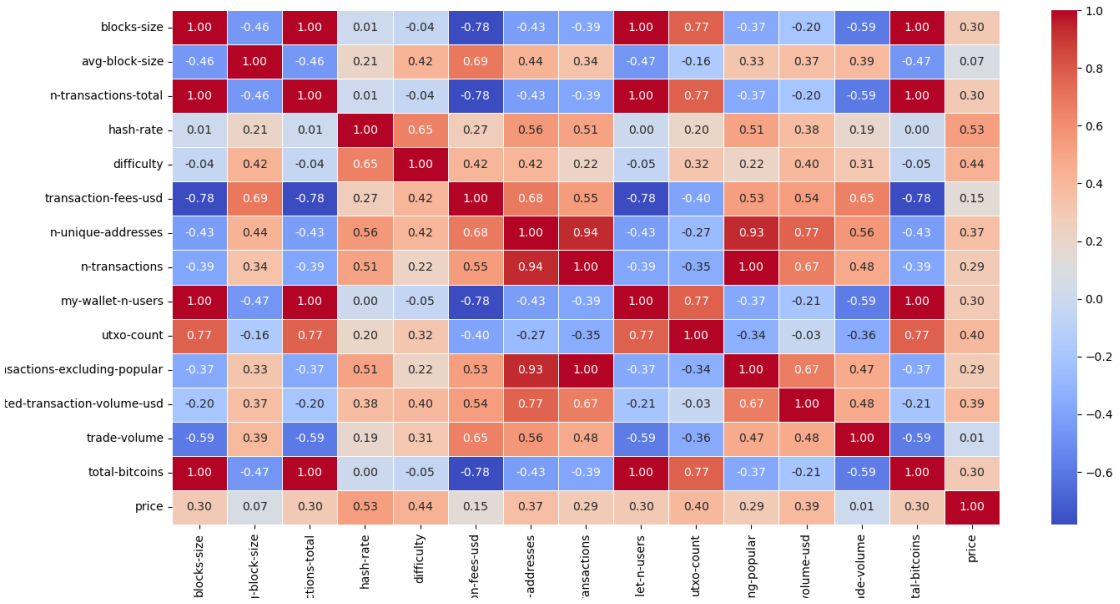

Figure 10: Correlation Matrix

purchases we need a specific amount of Bitcoin to pay, in order to ensure we pay the exact amount we create smaller denominations of Bitcoin which is basically spendable change, this creates a UTXO for the value of change. This means that an increase in transactions over the Bitcoin Blockchain means that the need for exact change is needed thus the generation of more UTXO. Although there may not be a direct relationship between UTXO and price, it does affect the market speculation and the demand and supply factors, in turn leading to an indirect affect on price. Now looking at the other correlations, it goes to show that we have a number of features have weak positive correlation to our price. This effectively means that there is a weak linear relationship between the features in this study. Another important thing to note is that correlation does not always mean causation and it is possible that there may be features and variables outside this study that may have a more direct affect on price and other variables. This also goes to validate our choice of model architecture as we chose deep learning models which are known to be a good fit for non-linear models.

4.2.4 Feature Engineering

In order to provide a machine learning model more information, feature engineering generally involves the development of new variables using an existing data set. It helps to relate underlying trends and explore more hidden patterns which may not always be obvious given current information. In order to provide more tools to our model we

decided to engineer some price based features that may help provide better understanding of pricing trends to our model along with our existing technical indicators. As our problem is based on time-series data, it is essential to capture the temporal patterns hidden in the data, in order to detect these trends we created the simple moving average (SMA) features which are number 17 and 19 in table 1 about our features. The simple moving average basically creates a smooth average price model based on a specific number of days, for every variable we calculated a 10 day and 50 day SMA. These SMA variables are used to provide information about long term price trends. The formula for calculating simple moving average is as follows;

$$SMA = \frac{\sum_{i=1}^{n} p(i)}{n}$$

Where n is the number of days we calculate average for and p(i) is the price for day i. One thing we must note when engineering features such as moving averages, data leakage can become a possibility. Data leakage essentially refers to the accidental flow of information to the test data. For example if we provide information from the future to our test data, the performance of the model will surely be very high and overly optimistic and may not be an accurate evaluation of the model. In order to avoid the loss of integrity of our model, we created shifted MA streams which did not take into account the current price when calculating the moving averages. This meant that the information received by our model was all from past figures and avoided data leakage ensuring the integrity of our model. Figures 11 and 12 show a visualisation of the

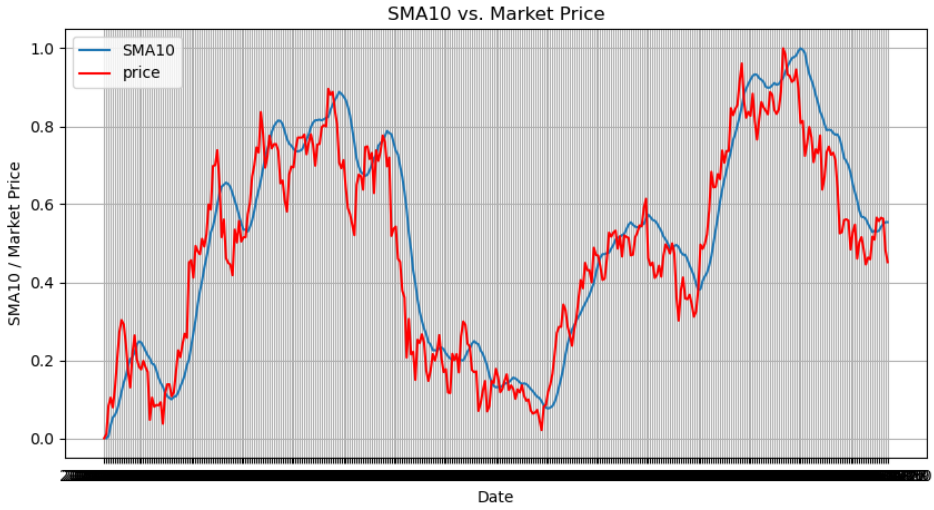

Figure 11: 10-Day Simple Moving Average vs Price

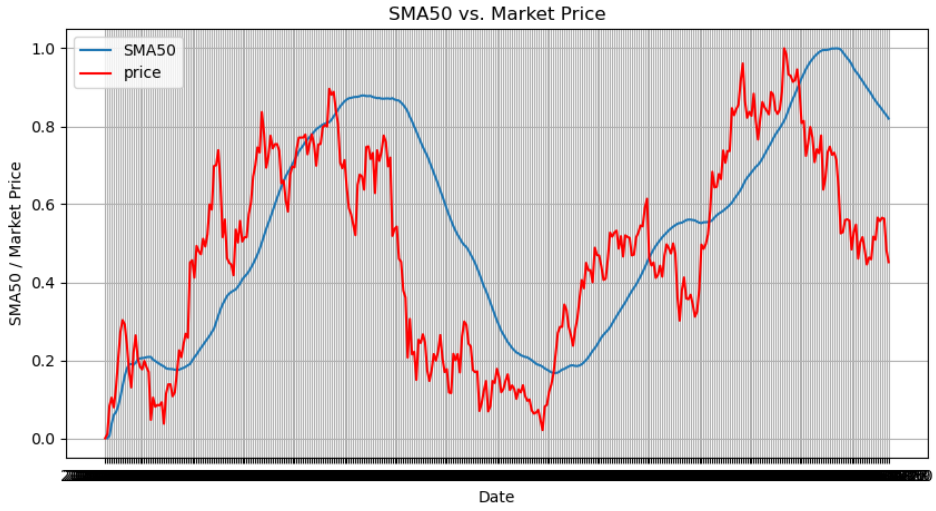

Figure 12: 50-Day Simple Moving Average vs Price

10 day and 50 day moving average plotted against the price. The visualisation clearly shows the smooth trend of price over time, with 50 day moving average focusing more on the long term trends and the 10 day average is more responsive to shorter trends. The visualisation shows the plots for our moving average to be slightly shifted ahead of the price plot which is a direct result of our technique to avoid data leakage.
Simple moving average although provides a good smooth average may not be fully responsive to short term peaks and falls, which given the nature of our problem, is very natural in cryptocurrency markets. So in order to capture short term trends better we added a second type of moving average variable, the Exponential Moving Average (EMA). EMA is another popular time series analysis feature which using weightage to give importance to certain days. It gives a higher weightage to more recent data in order to ensure higher response to sharper change in trends as compared to SMA. The formula for calculating EMA is as follows;

$$EMA(t) = \alpha * p(t) + (1 - \alpha) * EMA(t - 1)$$

Where p(t) is the price of Bitcoin at time t and alpha is the rate of responsiveness to data, higher the alpha the more it responds to recent changes. It is a recursive formula which uses the EMA calculated at previous time stages to calculate current EMA.

Our 10-day EMA and 50-day EMA can be observed in the figures 13 and 14. We can see the difference between the SMA and EMA by comparing the plots where EMA seems to be more responsive to the change in price, especially in the middle of the plot where price rises, EMA plot line is able to capture the peaks swiftly. Both EMA and

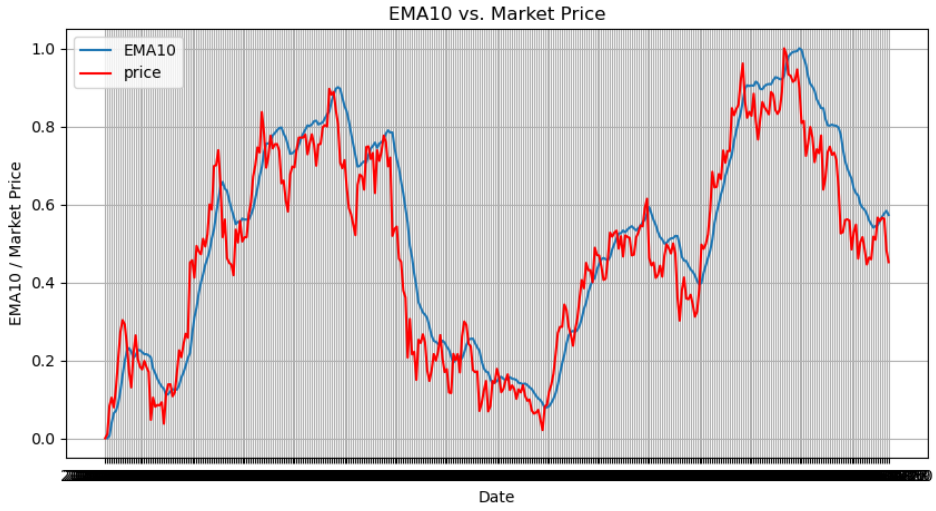

Figure 13: 10-Day Exponential Moving Average vs Price

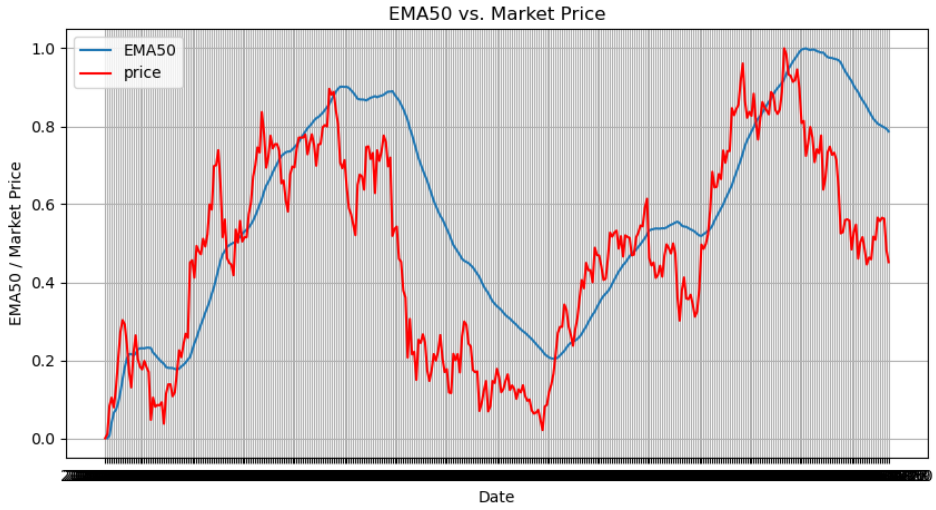

Figure 14: 50-Day Exponential Moving Average vs Price

SMA are used in our model as EMA helps capture sharp trends while SMA will provide our model with a much needed stability,
Our final type of variable was a lagged variable, this variable was added later in development in an attempt to improve the performance of our model. We used a lagged

predictor which essentially incorporates the target value from previous stages as additional inputs allowing to capture auto correlation in the data. So basically by using the price of Bitcoin from 1, 2 and 3 days ago we were able to account for seasonality trends and improve our model performance and as we are using the data from previous time steps and not feeding information about the future, we avoid the data leakage problem.

4.2.5 Sentiment Analysis

Sentiment can be a very powerful motivator for price especially in speculative markets, the affects of sentiment is a widely studied phenomenon. It may not provide the complete overview of the market sentiment but it does have a certain influential capability. Essentially, sentiment can be seen as the public perception and opinion. There are many mediums for gathering sentiments, social media and news articles are powerful mediums for sentiment. Market perception can be determined by the level of sentiment regarding the asset. An article about future development regarding the Blockchain can lead to good market sentiment which can influence investors to buy into the market, same way social media allows a good medium for promoting and reviewing Bitcoin performance. The ability to translate sentiment into price prediction is sought after. In order to be able to do that, we need to be able to obtain the sentiment from raw data collected. This can be done using Natural Language Processing libraries such as NLTK (Natural Language Toolkit). Using NLTK's pre-trained sentiment polarity predictor we were able to predict sentiment for each article text. One thing to note, the data set for article we had obtained was limited as our data source CryptoNews API is a paid service so we able to obtain a sample data set from public online source which were originally gained through CryptoNews. The sentiment for each article was generated ranged from -1 to 1 denoting negative and positive sentiment text. This was further normalised and mapped to a range of 0-1 in order to match it to our existing scaling criteria.

4.3 Methodology and Implementation

Our study focuses on 4 types of deep learning models as mentioned before, the following sections will discuss more detail about the architecture of each of our model and the technical code that was used to implement each model. In order to create a better visualisation of our created models, we used the Keras's plot model feature allowing us to observe the structure of model layers. The figure for each model is displayed in their respective sections.

4.3.1 Training Data

Before starting to build models, the data set needs to be processed in an appropriate manner to be fed into our deep learning models. In order to do that we need to follow some principles. After importing the data additional variables can be added that may provide more information regarding the time series data, which in our case includes the

Simple Moving Average (SMA), Exponential Moving Average (EMA) and Lagged variables for 3 days prior. Our data was processed in a consistent manner to avoid any data leakage. After all feature engineering is completed, the data set needs to be cleaned of all missing values, this must be done in an appropriate manner to avoid any resulting noise or bias. This study used the concept of removing any rows with missing values. As our dataset was collected in a structured and complete manner, this allowed us to avoid any significant loss of data, with the only **NAN** values existing in the first and last row dues to the calculation of our Lagged Variables.
The next step involves feature scaling, this is done in order to ensure that the data is normalized, i.e all our data is in the same range. This is a crucial step when dealing with time series data, as normalized data can be processed faster and the underlying trends can be captured effectively. This project focuses on the use of the MinMax Scaler to scale the features and the target values separately, the reason behind scaling separately is to ensure that we are able to inverse scale the target value data and predictions after they have been made.
GRU and LSTM models rely on sequenced data to be able to learn the different patterns in the data set. In order to organize our data into sequences the data was split into overlapping windows. Each window consists of a number of data points known as the time-step and each window is updated according to step-size. A time-step of 3 means that our data set is split into sequences of 3 data points in each window, and a step-size refers to how often the window is updated. For example when predicting the price for Monday, the data will use a window consisting of data from Friday, Saturday and Sunday to make predictions. For predicting the price for Tuesday, the sequences used will depend on the step-size, using a step-size of 2, the window will be updated after 2 time stages which means that the data sequence used for predicting the price for Tuesday will be the same as the one for Monday, however if we change the step-size to 1, the window will be updated at every time stage and the window used to predict the price on Tuesday will be the data points, Saturday, Sunday and Monday. The code listing shows the code for creating the sequences, where **X-train-seq** is the list of sequences for x values or the feature values. The **y-train** variable contains the corresponding target sequences for our feature data.

Finally the data needs to be split into training and testing data, as we need to retain a set of unseen data in order to evaluate the model performance and its ability to generalise. The data is split according to the 80/20 rule, with 80% of the data used as training data and the remaining 20% used for testing the model. As seen in line 7 of the code listing , **y-train** and **y-test** gives us the testing data and the actual testing target values which will be used for evaluation of the models. The training data is then fed into our respective models. The models are then fit and trained with the specified hyper parameters. And finally we are able to test our models using the y-train and y-test data. We must remember that, before the data was split into train and test data, we normalised our data set using the MinMax scaler, so our predictions will be in the form of scaled values. In order to return unscaled values, the predictions will be inverse scaled

using the same scaler used to scale them in the first place. This allows us to observe our predictions in the actual bitcoin price scale and makes it better when visualising our predictions on the price chart.

4.3.2 Long Short-Term Memory (LSTM) Price Models

LSTM networks have the capability of dealing with short term and long term memory using a feedback links. This allows LSTM to treat data as complete sequences instead of independent data points. This allows it to retain patterns and temporal changes.The figure 15 shows the architecture for a base LSTM model and will help us analyse the different components present.

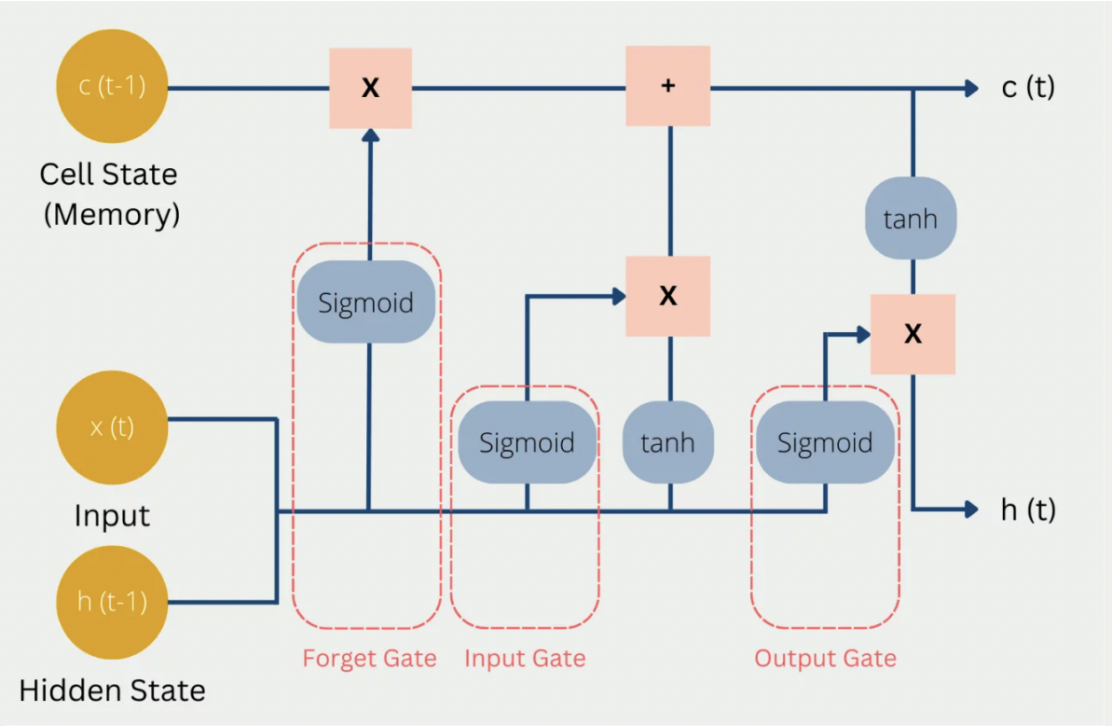

Figure 15: Architecture of LSTM Layer (source: Towards Data Science

Cell state $c(t-1)$ in figure 15, the memory cell of the LSTM model, it serves as a storage state for all the information learnt for the time step sequence. The cell state is used to storing and updating the information used for predictions if the cell is in the final layer or used to pass the information as input to another LSTM layer. The cell is modified over time with information updates regarding the patterns in the data. **Hidden state** $h(t-)$, collects all the information the model has learnt overtime up to the current stage in time and is used for making predictions and can be considered as

the output cell or state. **Forget gate**, uses a sigmoid function to determine which of the previous information is still relevant, judging the information in hidden state and the current input, it preserves information that is deemed valuable. **Input Gate**, is the date responsible to adding new information into the model, it determines the value of the new input and adds it to the memory cell after applying weights and add the hidden state values through multiplication, the new information and the information from the forget gate is then used to create the new cell state $c(t)$. **Output Gate** finally outputs the prediction calculated in the hidden states, and multiplied with the cell state to determine the output.

Our LSTM model was created using Python's **Tensorflow** and **Keras** libraries to create

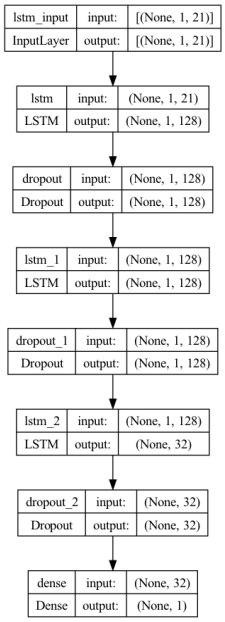

Figure 16: Architecture of LSTM Model

a sequential layered model 16. Starting with an LSTM layer consisting of 128 neurons, this layers outputs a time sequences to a **Dropout** layer, this is a technique used to avoid overfitting as it randomly drops out some of the neurons while training, allowing a generalised learning process. The network will not be over specialised in specific pattern and will have a level of robustness about it. The model then passes through another LSTM layer made up of 128 neurons which further extracts deeper features from the data, the resulting sequences are then passed on to another dropout layer in order to regularise the model even more. The resulting output is then passed through to a final LSTM layer used to summarise all the patterns learnt overtime, this layer consists of 32 neurons. The result is passed though yet another dropout layer before entering a Dense layer consisting of a single neuron responsible to making a final price prediction.

The model was trained using the Mean Squared Error (MSE) loss function. The training minimise the errors by adjusting weights over time, the model then does a backward

pass also known as backpropagation to update the weights based on the calculated error gradients. The following is a short snippet of the code used to create the model.

```
model = Sequential()
model.add(LSTM(128, return_sequences=True,
input_shape=(X_train_seq.shape[1],
X_train_seq.shape[2])))
model.add(Dropout(0.2))
model.add(LSTM(128, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(32))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mse')
```

The above code creates the LSTM layers using the shape of the data. The model is compiled by specifying the loss function used for training and add the optimizer used for training which in our case is the Adam optimizer. This is a sophisticated model, the number of layers creates a complex framework which is able to model non linear relationships effectively and make accurate predictions based on our time series data. The model was then tested using appropriate metrics such as MSE, MAPE, R2 score and MAE. The results will be discussed in the conclusion section of this paper. The full code can be found in the **LSTMBuilder.py** file, in the **baseLSTM** function which takes in 3 hyperparameter arguments which are used to configure the model, with time steps, epoch and batch size.

4.3.3 Bi-Directional Long Short-Term Memory (LSTM) Price Models

Bi-Directional LSTM are a specialised version of LSTM which allows the data to be be processed in both directions when training also known as the Dual Context. Basically allows the consideration of past and future values when training the model, allowing the model to develop better understanding of the non linearity of the data set which is very useful when dealing with our data set where the relation between our technical indicators in not linear and has a weak relationship. Further on this helps with the vanishing gradient problem which normally affects LSTM networks, Vanishing gradient problems hinders the learning ability of RNN models, as the during backprobagation the gradient is flowed through multiple layers, the value of updates gets smaller and smaller. Bi-LSTM works with 2 passes, the forward and backward, calculating and combining the gradient for both passes, creating a stronger gradient signal and allowing a stabilized learning process where the model adapts to the complex patterns better. The architecture of a Bi-Directional LSTM has 2 hidden states, one for each of the backward and forward passes. The combination of the states is used to make predictions. The following figure shows a visualisation of the forward and backward pass of the BiLSTM model. The model starts by taking in a the input data and is processed using the forward

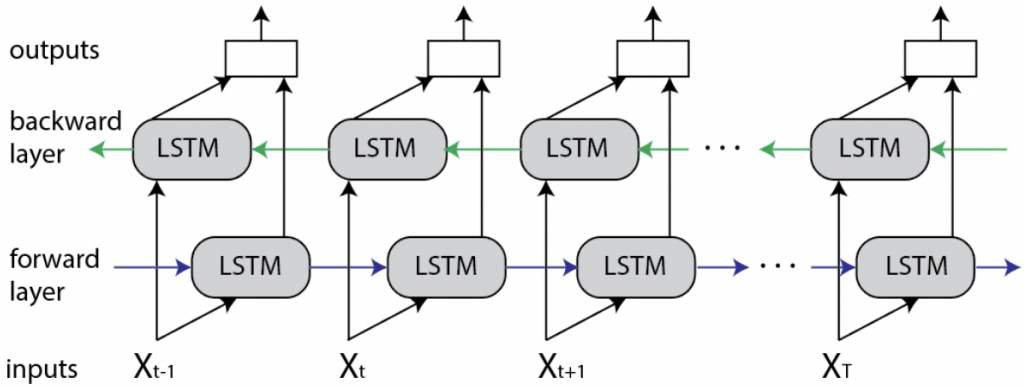

Figure 17: Architecture of Bi-Directional LSTM Model

LSTM layer updating the cell states and hidden states as we go along, the working of the cell and hidden states are the same as in a regular LSTM model. Simultaneously, the same data is passed through our backward LSTM layers updating the hidden and cell states in the opposite direction. The model then combines the output from both passes, this output essentially has calculated the the future and past trends. This is done for every time step. Next, up we perform backpropagation, calculating the loss using our loss function (MSE), which is the mean squared error or difference between the target and actual values, the formula for which is as follows; $MSE = \frac{1}{n}\sum_{i=1}^{n}(y - yi)^2$
Where, n is the total number of predictions, y is the actual predictions and yi is the values predicted by our model. MSE is used in the calculation of the loss gradient which is the backpropagated through both the forward and backward LSTM layers, updating the parameters using optimisers, which in our case is the Adam optimiser, which uses each parameter's past gradients to update the learning rates.

The figure 18 shows the architecture of our mode. The model was built using Tensorflow and Keras libraries. The model is built using stacked layers similar to the LSTM model, consisting of 3 Bi-Directional LSTM layers, with each of the first 2 layers made up of 128 neurons each and the last one made u of 32 neurons. Dropout layers are added after each BiLSTM layer to make the model more robust. The model finishes off with a Dense layer, which aggregates all the outputs from the past layers and generates the final prediction. The code listing below provides the exact code to create and compile our BiLSTM model.

```
model = Sequential()
model.add(Bidirectional(LSTM(128, return_sequences=True,
activation='relu'), input_shape=(X_train_seq.shape[1],
X_train_seq.shape[2])))
```

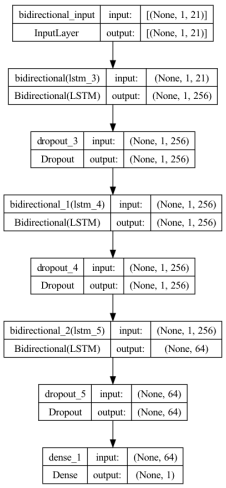

Figure 18: Architecture of our Bi-Directional LSTM Model

```
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(128,
return_sequences=True, activation='relu')))
model.add(Dropout(0.2))
model.add(Bidirectional(LSTM(32,
activation='relu')))
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mse')
```

The full code can be found in the **LSTMBuilder.py** file, in the **BiLSTM** function which also takes in 3 hyperparameter arguments.

4.3.4 Gated Recurrent Unit (GRU) Price Models

Next we worked with the Gated Recurrent Unit, which like the LSTM and Bi-LSTM is a special type of Recurrent Neural Network, and is another model designed to solve the vanishing gradient problem. GRU models were designed to be a more computationally efficient LSTM.

Consisting of 2 gates, the **reset gate** and the **update gate**, the GRU model is comparatively easier to train due to it having fewer parameters as compared to traditional LSTM models. GRU is able to replicate the performance of LSTM when dealing with short term dependency, with a better training time and can easily be fine tune to our specificity. The model starts the by combining the weighted input x(t) and h(t-1) and applying a sigmoid function in order to map the values to a range between 0 and 1. The update gate, similar to LSTM's input gate decides the amount of information that us

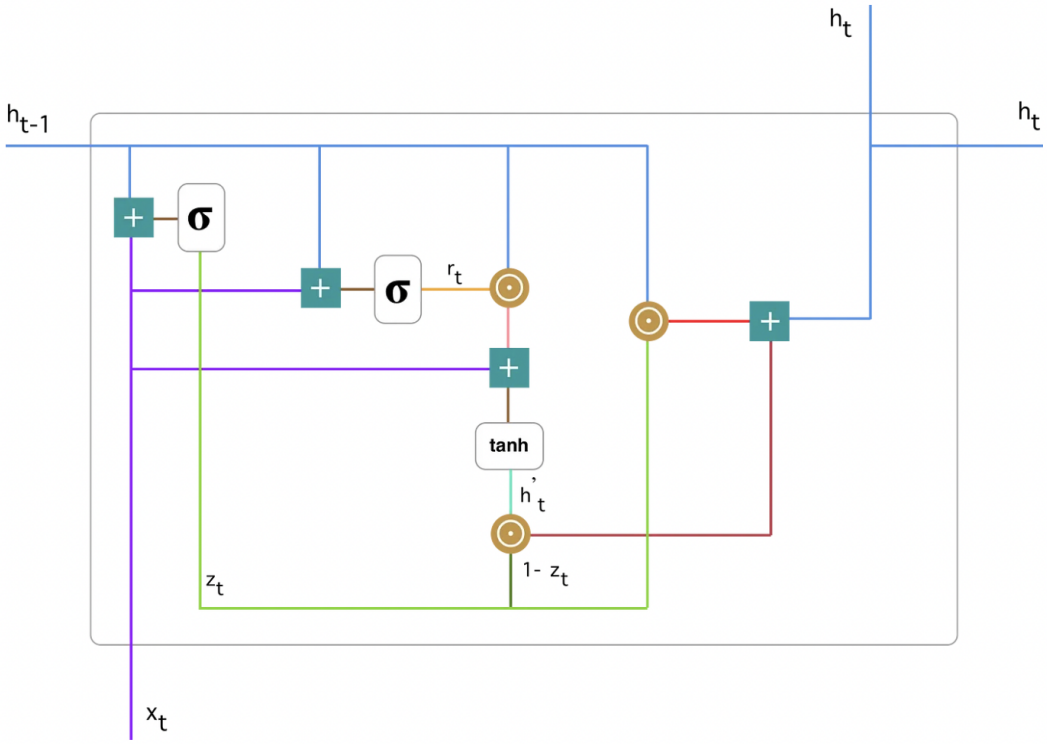

Figure 19: Architecture of GRU Model

proceeded. The reset gate performs similarly to an LSTM's forget gate, where it decides what information to drop moving forwards.
Figure 20, shows the model layers of our GRU model. The code listing below shows the code used to create our GRU mode, we first imported the required library components in the form of the GRU, Dropout and Dense layers from Keras. We followed a similar criteria for layers as we did for both LSTM models. The model is made up of 3 GRU layers each with 128, 128 and 32 layers respectively, with a dropout layer after each GRU layer. The model finalises the output received from the layers into the Dense layer which is used to make a final prediction. The same optimiser is used in order to compare the model performances better.

```
model = Sequential()
model.add(GRU(128, return_sequences=True,
input_shape=(X_train_seq.shape[1], X_train_seq.shape[2])))
model.add(Dropout(0.2))
model.add(GRU(128, return_sequences=True))
model.add(Dropout(0.2))
model.add(GRU(32))
```

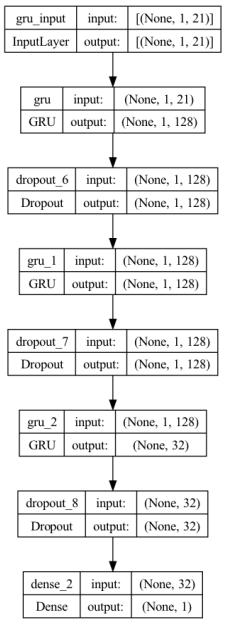

Figure 20: Architecture of our GRU Model

```
model.add(Dropout(0.2))
model.add(Dense(1))
model.compile(optimizer=Adam(learning_rate=0.001),
loss='mse')
```

The full code can be found in the **LSTMBuilder.py** file and the appendix, in the **GRUModel** function which also takes in 3 hyperparameter arguments.

4.3.5 Convolutional Neural Network (CNN) Price Models

Our final model is the Convolution Neural Network which is a specialised neural network, normally a powerful network used for grid based tasks such as image analysis, however this unconventional approach can prove useful as CNN's ability to learn hierarchical features from data automatically may help identify local dependencies in our price data. CNN may be quite useful in detecting smaller trends and patterns withing larger trend sequences which may account for adapting to longer price movement trends while also learning the short term price volatility. This network is also less vulnerable to overfitting as it relies on dimensionality reduction. We will be studying on how well it performs in our task as compared to our LSTM and GRU network which are known to be more suited to time series data.

The figure 21 shows us the visualisation for our Convolutional Neural Network model, where the model is made up of a single **Conv1D** layer. Unlike the above LSTM and GRU models, the CNN model takes in an additional parameters in the form of **Kernel size**, which basically controls the size of our convolutional window. This convolutional

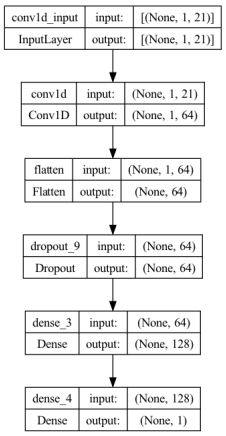

Figure 21: Architecture of CNN Model

window considers the size of data points considered, we set the kernel size for our model to be equal to the assigned time step which is the number of data points used to predict the price next day. **Filters**, is the number of neurons used in the layer which is set to 68 and a ReLU activation is added to the Conv1D layer. The purpose of the Flatten layer after the Conv1D layer is to ensure the data passed on is in a 1 dimensional vector. The model further features a Dropout layer to avoid overfitting and a Dense layer to aggregate the output and make predictions. This can further be seen in the code listing below.

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=timestep,
activation='relu', input_shape
(X_train_seq.shape[1], X_train_seq.shape[2])))
model.add(Flatten())
model.add(Dropout(0.2))
model.add(Dense(128, activation='relu'))
model.add(Dense(1))
```

After a series of experiments the model performed well with the lowest MAPE of 0.05. In order to improve the model performance we created a hybrid CNN-LSTM model, taking inspiration from [8] where Yan Lei and Wei Dai, used a similar approach to make sure that the model was able to capture the temporal dependencies accurately. By adding an additional layer in the form of an LSTM with 32 neurons, and another set of dropout layers we were able to improve the model performance. The detailed results will be discussed in the next chapter along with the experiment development and more clarification about the quantitative and qualitative metrics used to evaluate the models. The full code can be found in the **LSTMBuilder.py** file and the appendix, in the **CNNModel** and **CNN_LSTMmodel** functions respectively which also take in 3 hyperparameter arguments and set the kernel size of the model equal to the chosen time

step.

4.3.6 Long Short-Term Memory Price Models with Sentiment

In order to gauge out better a more holistic approach using our price prediction models, the sentiment polarity for articles was added to our model. The sentiment polarity from January to November was added to the price data from 2021 by merging the data using Pandas Dataframe merge function, this resulted in the reduction of our data set in order to train the model based on both sentiment and technical features. Due to a smaller set of data, we increased the set of training data to 90%, with the remaining 10% as test data. By adding sentiment values to the feature set of each of our models; LSTM, Bi-LSTM, GRU, CNN and CNN-LSTM hybrid, we would be able to observe if our sentiment data is able to improve the prediction accuracy of the models. The complete code can for our models with sentiment can be located in the **LSTMBuilder.py** file and the appendix section, with the name of the model with a 'sent' suffix, to denote that model inclusive of sentiment.

5 Results, Analysis and Evaluation

5.1 Evaluation Metrics

After training of the model, the next step is to evaluate the model performances, and fine tuning to improve performance. These metrics help track the improvement of the models during experimentation and are a way to allow people with n on-technical background to understand the model performance. Further on we need to ensure the robustness of the model, by testing the generalisation ability of our model on the unseen test data. We evaluated the model performances based on a number of qualitative and quantitative metrics;

- R2 Score
- MAE: Mean Absolute Error
- MSE: Mean Squared Error
- MAPE: Mean Absolute Percentage Error

R2 score quantifies the percentage of the dependent variable's variation that the independent variables in a regression model can account for. It ranges from negative infinity to 1, the value closer to one signifies that the majority of the variance is explained. **MAE** is the measure of absolute difference, providing the average error in prediction. It ranges from 0 to infinity, with the value closest to zero signifying low error and well trained model. **MSE** is the measure of average squared differences, ranging from 0 to infinity. Due to the square function's tendency to give more weights to higher values, this metric is susceptible to significant outliers. Finally, **MAPE**, is the percentage difference measure between our expected and predicted values. The value ranges from 0 to 100.

5.2 Experiments

In order to test the model performance, each model was tested using the same data set and evaluated with the metrics above.The **Epochs**, **Batch Size** and **Time Steps** hyper parameters we used for model optimisation. The learning algorithm's "epochs" hyperparameter controls how many times it will go over the whole training data set.One time during each epoch, each sample in the training data set has the opportunity to modify the internal model parameters. Epochs are composed of one or more batches. The batch size is the number of training examples used by before updating the model, The size of a batch must be one at the least and less than or equal to the number of samples in the training data set at the maximum. It affects how the deep learning model converges and the speed at which it does so. The time steps are the number of data points considered for each prediction. The models were tested with every configuration of the above hyper parameters, with time steps 1, 2, 3, 5, 7, epochs 50, 100 and batch size 32, 64, 128. Each model was tested more than 40 times using the configurations., in order to find the best combination of hyper parameter for each model. After the each

model was trained, they were evaluated using the test set and the evaluation metrics were stored in a csv file **modelTest.csv**, which was further formatted to contain values with 4 significant figures, called **model_results_formatted.csv**. The next section provide the results for some of the best performing models, along with an analysis of the models. The models can easily be tested by running the file **modelTest.py**, from the source code, where the the **LSTMBuilder** has already been imported and all data preprocessing has been done. Any model configuration can easily be tested by calling its respective function along with a choice of hyper parameters.

5.3 Results: Price Prediction Models

Below is a table of the top 3 performing configurations for each price prediction model; Table 2 shows the results of the top runs for each of our models. Note that the MAPE

Model	Time Step	Batch Size	MSE	R2	MAE	MAPE	
LSTM	1	32	100	0.0041	0.8484	1829.068	0.0326
LSTM	1	64	100	0.0038	0.8563	1847.4281	0.0328
LSTM	1	64	100	0.0039	0.8559	1846.4902	0.0329
Bi-LSTM	1	32	50	0.0039	0.8544	1818.2355	0.0319
Bi-LSTM	1	64	100	0.004	0.8505	1818.1294	0.0322
Bi-LSTM	1	32	50	0.0038	0.8569	1818.5729	0.0322
CNN	2	32	100	0.0053	0.805	2176.2315	0.0388
CNN	3	32	50	0.0066	0.7574	2343.2402	0.0406
CNN	3	32	100	0.009	0.6677	2872.4324	0.051
CNN-LSTM	1	32	50	0.0036	0.8652	1870.7309	0.0328
CNN-LSTM	1	32	50	0.0041	0.848	1868.4641	0.0333
CNN-LSTM	2	32	50	0.0045	0.8324	1999.1691	0.0349
GRU	1	64	50	0.004	0.8509	1863.4976	0.0327
GRU	1	64	50	0.0039	0.8546	1854.7873	0.0329
GRU	1	64	50	0.0042	0.845	1904.2952	0.0332

Table 2: Top 3 Results for Each Model

values are not in percentage form. Lets analyse the performances for our models;
Bi-LSTM Model

- The Bi LSTM model consistently produces results, with a variety of batch sizes and epochs
- The top 3 models have a MAPE value ranging from 0.0319 to 0.0322, showing high accuracy.
- High R2 of 0.85 shows the model's ability to explain 85% of the variance in the data.

- Overall, the Bi-LSTM model seems to perform well across different hyperparameter configurations.

CNN Model

- MSE values are higher compared to the Bi-LSTM model, indicating slightly worse predictive accuracy.
- R2 of 0.805 is impressive considering we used an unconventional architecture.
- R2 and MAE seems to decrease as the time step and kernel size increases
- The best MAPE of 0.038 is still quite interesting, showing the percentage of error in predictions is quite low.

CNN-LSTM Model

- Similar performance to the Bi-LSTM model in terms of MSE and R2.
- MAPE values range from 0.0328 to 0.0349 which is lower as compared to the CNN model which shows model improved after addition of LSTM layer.
- MAE and MAPE values are marginally higher than Bi-LSTM, and LSTM so the model is not as effective compared to its counterparts
- Considering the combination of CNN and LSTM layers, this model provide reasonable predictive performance.

GRU Model

- GRU's performance is similar to Bi-LSTM in terms of MSE and R2.
- Model shows low relative error percentage, as seen by the low MAPE values of 0.0327 to 0.0332
- MAE and MAPE values are slightly higher than Bi-LSTM, but still in an acceptable range.
- Generally consistent results across different configurations.
- The training time for the model was faster than the training for Bi-LSTM showing effective computations.
- The GRU model offers an alternative to LSTM-based architectures.

LSTM Model

- The base model fits the data well showing an R2 value of 0.8484 to 0.8563 for the best 3 test runs.
- MSE and R2 performance is very close to Bi-LSTM.

- Slightly higher MAE and MAPE values compared to Bi-LSTM.
- Results remain consistent across different configurations.
- The LSTM model demonstrates competitive performance.

The table shows that the Bi-LSTM model had the best performance when trained using time step = 1, epochs = 50 and a low batch size of 32. This model outperformed all other models in the when considering the metrics MSE, MAE and MAPE. In a surprising result the CNN-LSTM hybrid model with time step = 1, batch size = 32 and epochs =50, led the charts in R2 score with 0.8652. These results are quite insightful when comparing the abilities of different models to capture the trends of the Bitcoin price models.

5.4 Results: Models with Sentiment

The following are the results for our price prediction models with sentiment. We will evaluate them separately due to the use of a different data set.

Model	Time Step	Batch Size	Epochs	MSE	R2	MAE	MAPE
Bi-LSTM Sent	1	32	50	0.0062	0.864	2372.5026	0.045
Bi-LSTM Sent	1	64	50	0.0072	0.8421	2549.3159	0.0481
Bi-LSTM Sent	1	32	100	0.0085	0.8122	2718.877	0.0503
CNN Sent	1	64	50	0.013	0.7127	3419.3494	0.0625
CNN Sent	1	64	50	0.0137	0.6966	3482.7358	0.0636
CNN Sent	1	32	50	0.0197	0.5649	4117.5252	0.0741
CNN-LSTM Sent	1	32	50	0.0126	0.7223	3315.4044	0.0607
CNN-LSTM Sent	1	64	50	0.0124	0.7257	3387.69	0.0617
CNN-LSTM Sent	2	32	50	0.0124	0.7318	3335.4028	0.0619
GRU Sent	2	64	100	0.0054	0.8822	2174.7442	0.0418
GRU Sent	2	32	100	0.006	0.8695	2324.8101	0.0441
GRU Sent	1	64	100	0.0069	0.8486	2479.8663	0.0464
LSTM Sent	1	64	100	0.0057	0.8741	2268.8151	0.0433
LSTM Sent	1	32	100	0.0058	0.8709	2299.939	0.0436
LSTM Sent	1	32	50	0.0061	0.8658	2326.1816	0.0444

Table 3: Top 3 Results for Each Model with Sentiment

Table 3 shows the results of the top performances for each of our models after the addition of sentiment data. Lets analyse the performances for our models;
Bi-LSTM Sentiment Model

- The model performed well in terms of the metrics however it had higher MAPE and MAE as compared to the model without the sentiment data.

- It had an impressive R2 score of 0.864 which was higher than the model without sentiment which gives us an interesting outlook on the sentiment model.
- The model performed the best with the parameters of time step = 1, batch size = 32 and 50 epochs.

CNN Sentiment Model

- The model performance was adequate but can be much better in terms of MAPE and R2.
- The performance was worse as compared to the model without sentiment data.

CNN-LSTM Sentiment Model

- The model performance decent with R2 and MAPE at 0.63 and 0.061 respectively
- The hybrid model performed better as compared to the sole CNN model, however was not able to improve on the performance of the model without sentiment.

GRU Sentiment Model

- The GRU model seems to have improved with the inclusion of sentiment.
- It resulted in a high R2 of 0.882 which is the highest recorded in all models, with and without sentiment data.
- The MSE is relatively low which is a positive.
- A MAPE value of 0.0418 is promising however is higher than the model without sentiment

LSTM Sentiment Model

- The model performed similar to the non sentiment model with not much significant changes.
- The MAPE seems to have risen which is a negative.
- The R2 score however has shown a strong value of 0.865 which is an improvement over the non sentiment model

Some models such as GRU has definitely seen an improvement with the sentiment data set. Overall, we can see a trend, the models with the addition of sentiment data seem to be performing better in terms of the R2 score however seem to be under-performing in the case of MAPE. This can be taken as a sign that the sentiment model is able to detect much of the variance in our data set and perform well when detecting long term trends however is not that effective against specific short term trends leading to a higher percentage error.

Overall we can see that some of the models have indeed seen an improvement in performance after the addition of sentiment data according to some evaluations but have under-performed in percentage error. The reasons for this can be, shortage of training data, considering we used a shortened data set in order to match our sentiment data set. High level of noise and bias in the article can also be a reason for under performance of the model.

5.5 Results Comparison to Existing Works

Using a benchmark set by Seabe in the paper, 'Forecasting Cryptocurrency Prices Using LSTM, GRU, and Bi-Directional LSTM: A Deep Learning Approach'. Seabe quotes in his paper 'the Bi-LSTM model has the best performance in forecasting BTC prices, with the lowest RMSE and MAPE values of 83.9531 and 0.1243', our BiLSTM model was able to beat the this benchmark with MAPE of 0.0322 and 0.045 in models with and without sentiments respectively.

5.6 Visualisation Charts

In order to observe the performance of the best performing models, it is important to see the side by side comparison of our models predictive capabilities with the actual prices. This can be done by visualising our predictions using MatplotLib. The predictions of the best performing model; Bi-LSTM model with the hyper parameters, time step = 1, batch size = 32 and 50 epochs, were visualised on chart along with the actual prices. The following figure 22 shows the the values predicted by the Bi-LSTM model (in pink) vs the actual price values (in golden);

in 2021, the price of Bitcoin was significantly volatile, with multiple peaks and significant drops over the year making it a appropriate data to train and test the models with. When we compare our overall trend lines, both seem to be moving in the same direction almost simultaneously. We can clear see that our model was able to capture the price trends, with accurate predictions, matching all the short term peaks and sudden drops. The model was able to adapt to the price volatility and seems to have generalised pretty well. We will further analyse a more detailed version of the chart which focuses only on the testing set, allowing us a closer observation to the trend line comparison in figure 23.

On a closer look we can further conclude that the model has been effectively matching the trends, as it is responsive to every peak and drop. The only point of discrepancy occurs between index 40 and index 50 where our model predicted a drop instead of a rise in price, however was able to recover well. Overall we can say that the model has done well in capturing both, long and short term trends effectively.

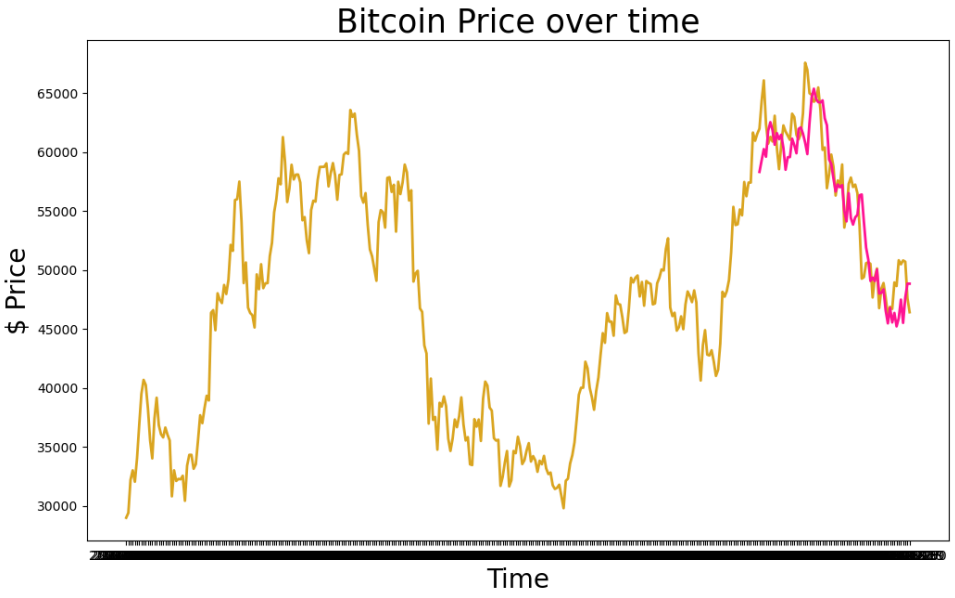

Figure 22: Predictions vs Actual Price Chart Overtime

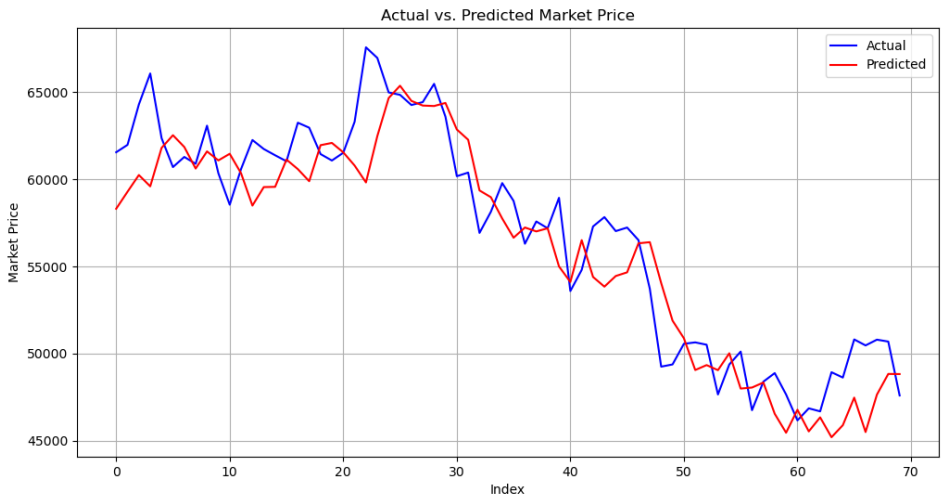

Figure 23: Predictions vs Actual Price Chart

6 Legal, Social, Ethical and Professional Issues

Throughout the creation of the current work, the four BCS Code of Conduct principles have been adhered to and applied, and the project's integrity has been protected. The Rules of Conduct of the Institution of Engineering and Technology (IET) have also been carefully stated, as have all instances of open-source libraries used in this project.
This study suggests using deep learning models to predict the price of bitcoin; it is an experimental research work and should not be construed as giving investment advice.

7 Conclusion and Future Work

In conclusion, the objective of this study was achieved, as we were able to explore the cryptoeconomics data sets in order to enhance Bitcoin price prediction, using a new approach with the use of technical indicators. We can conclude that the use of deep learning models, helped us understand the non-linear data boundaries using our price and technical data set. Furthermore we gained valuable insights into the complex relationship that exists between the price dynamics and market sentiment, by comparing performances of our LSTM, BiLSTM, CNN, CNN-LSTM, and GRU models.
The resulting analysis of our study has displayed the potential of the using deep learning models for Bitcoin price prediction. Our BiLSTM model has showcased excellent performance in terms of MAE and MAPE which has shown its superior accuracy when predicting prices. Our unconventional CNN and CNN-LSTM models have shown competitive results. The GRU and LSTM have showcased a stable and consistent performance with their predictions. These results highlight the potential of using several deep learning architectures to estimate bitcoin prices accurately.
Moreover, adding sentiment polarity data to our price prediction models adds an additional layer of complexity, which although did not consistently beat our non-sentiment models, it offered an insightful look at how market sentiment and speculation may affect Bitcoin prices. This highlights the need of taking both technological and emotional factors into account when predicting bitcoin prices. The use of transactional Blockchain data did prove to be suitable for predicting prices, with excellent metric results.
This study's conclusions point to the potential for strategic decision-making in cryptocurrency trading and investing by utilising the predictive capability of these models. Given the tremendous volatility and vulnerability to outside influences that characterise the cryptocurrency market, it is crucial to recognise the limits that such predictions inevitably carry. The features used to predict prices in this study may not always be the correct explanatory variables as there may be some other factors that affect price and the features as a whole.
Future study might examine the introduction of more sophisticated sentiment analysis techniques, incorporation of external economic data, and refinement of model architectures as the bitcoin landscape continues to change. The use of sentiment analysis could be a key for future models, however a consistent and accurate source of data will be required in order to be able to harness this key strength. This dissertation emphasises the potential of deep learning models to deliver beneficial insights for investors and enthusiasts in the dynamic world of digital assets, adding to the increasing body of knowledge in cryptocurrency research and prediction.

References

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Cryptography Mailing list at https://metzdowd.com*, 03 2009.

[2] J. Chen, "Analysis of bitcoin price prediction using machine learning," *Journal of Risk and Financial Management*, vol. 16, no. 1, 2023.

[3] G. G, H. M, and A. K, "Cryptocurrency price prediction using machine learning," *IJARCCE*, vol. 12, 04 2023.

[4] M. Rizwan, S. Narejo, and M. Javed, "Bitcoin price prediction using deep learning algorithm," in *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, pp. 1–7, 2019.

[5] P.-F. Pai and C.-S. Lin, "A hybrid arima and support vector machines model in stock price forecasting," *Omega*, vol. 33, no. 6, pp. 497–505, 2005.

[6] S. J. Rcik Dey, Saurabh Shukla and H. Lopes, "Bitcoin price prediction using lstm," *International Research Journal of Engineering and Technology (IRJET)*, 2022.

[7] P. L. Seabe, C. R. B. Moutsinga, and E. Pindza, "Forecasting cryptocurrency prices using lstm, gru, and bi-directional lstm: A deep learning approach," *Fractal and Fractional*, vol. 7, no. 2, 2023.

[8] Y. Li and W. Dai, "Bitcoin price forecasting method based on cnn-lstm hybrid neural network model," *The Journal of Engineering*, vol. 2020, no. 13, pp. 344–347, 2020.

[9] L. Kristoufek, "Bitcoin meets google trends and wikipedia: Quantifying the relationship between phenomena of the internet era," *Scientific reports*, vol. 3, p. 3415, 12 2013.

[10] S. Rosenthal, A. Ritter, P. Nakov, and V. Stoyanov, "Semeval-2014 task 9: Sentiment analysis in twitter," pp. 73–80, 01 2014.

[11] M. Khashei and M. Bijari, "An artificial neural network (p,d,q) model for timeseries forecasting," *Expert Systems with Applications*, vol. 37, no. 1, pp. 479–489, 2010.

[12] P. L. Seabe, C. R. B. Moutsinga, and E. Pindza, "Forecasting cryptocurrency prices using lstm, gru, and bi-directional lstm: A deep learning approach," *Fractal and Fractional*, vol. 7, no. 2, 2023.

[13] H. Jang and J. Lee, "An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information," *IEEE Access*, vol. 6, pp. 5427–5437, 2018.

[14] H. S. Hota, R. Handa, and A. K. Shrivas, "Time series data prediction using sliding window based rbf neural network," 2017.

[15] F. Valencia, A. Gómez-Espinosa, and B. Valdés-Aguirre, "Price movement prediction of cryptocurrencies using sentiment analysis and machine learning," *Entropy*, vol. 21, no. 6, 2019.

[16] J. Vella Critien, A. Gatt, and J. Ellul, "Bitcoin price change and trend prediction through twitter sentiment and data volume," *Journal of Financial Innovation*, vol. 8, 05 2022.

[17] T. Phaladisailoed and T. Numnonda, "Machine learning models comparison for bitcoin price prediction," in *2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE)*, pp. 506–511, 2018.

[18] L. Catania, S. Grassi, and F. Ravazzolo, "Forecasting cryptocurrencies under model and parameter instability," *International Journal of Forecasting*, vol. 35, no. 2, pp. 485–501, 2019.