**introduction**

Prediction has become a very crucial part in the modern business intelligence [22]. Various statistical methods have been applied. However, it can be quite complex when there are uncertainties or data is missing [22]. Accuracy is key when it comes to forecasting because it may involve high risks in an investment. Several studies have shown that machine learning techniques have the same classification and prediction power as statistical methods [23]. A few time series models like ARIMA, SARIMA, GARCH have been developed to solve the prediction problem [22]. Normally, several time series models of different algorithms are developed, then you compare which model has the best accuracy result. These time-series models assume there is a linear relationship between input and output [23].

Regression techniques such as linear regression, random forest, Support Vector Machine (SVM) , Decision Trees can also be applied to solve prediction problems [22]. The main concept behind regression is that the algorithms to find a relationship between independent variable (commonly called X) and dependent variable (Y). This technique checks what is the outcome after a causal effect is applied on the independent variables. Regression assumes patterns shown in the past will be repeated in the future [22]. Regression analysis is mainly applied in prediction problems.

Recent studies have also shown that neural network has also been applied as predictive models. Neural networks have shown that they can approximate a continuous function [23]. The neural network architecture is composed of a series of nodes and connections forming artificial neurons. The neurons can be joined to form layers and the layers can also be joined to form a network structure with at least one hidden layer (a layer that does not produce final output) [25]. Neural networks are models that try to mimic the human brain [24]. They try to “think and reason” when solving a problem. To do this, neural network uses feed forward and backward propagation techniques to facilitate learning. Neural network use weight and biases in order to obtain an output from the input. In feedforward the input is given then random weights and biases are generated to be able to generate the output and then during backward propagation is when we adjust the weights to improve accuracy of the model. Therefore, the adjustment of the weights while minimizing loss or improving accuracy in classification models is what we call learning in neural networks [25]. The neural networks have the ability to reconstruct fuzzy rules and obtain hidden relationships in the data [24]. This makes them very good at solving non-linear problems even without initial information [23]. Various neural networks, for instance, Feed Forward Neural Network, Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), have been applied in prediction problems. Neural network algorithms like CNN and LSTM have been able to achieve incredible results in very complex tasks like image and video classification as well as natural language and speech processing [25].

All the techniques including the statistical methods aim at improving the accuracy of forecasting. They do this by minimizing a loss function which usually is the sum of the squared errors [25]. Even though neural networks always seem faster and accurate than statistical methods, both have shown to have closely related performance and neural networks do not guarantee that the forecasting accuracy will automatically increase [25]. Just because machine learning is AI (artificial intelligence) doesn’t infer that it has any practical advantage over other alternative algorithms [25]. Therefore, there is need to always evaluate models using various benchmark models because different algorithms perform the best in different problems.

.

**Problem statement**

Ideally, with times series data, the data is usually volatile therefore, there is need to analyze the data in order to forecast and obtain future values and trends. In order to do this, the patterns or trend of the data are established using regression techniques. Linear regression is the most popular and has been widely researched and adopted in various areas like stock market, meteorology and many others. However, with highly volatile data, linear regression models cannot detect non-linear relationship or any interactions between the dependent and independent variable.in the data. Mathematically, this means we cannot predict the values from a straight line. In this research project, I am proposing implementation of neural networks to perform regression analysis on the time series data and for this case bitcoin prices against US dollars.

**Aims and objective**

The main aim of this project is to develop a model that predicts the prices of bitcoin (BTC) against US dollar (USD).

**Specific Objectives**

1. To develop a model that predicts bitcoin prices against US dollar.
2. To evaluate the accuracy of neural network models as prediction models.

**Justification**

This project will be beneficial for the me and other scholars to be able to gain more insight on time series analysis and neural networks.

In order to understand analysis and regression of data using neural networks, the project will look at the architecture, variants, advantages and challenges of recurrent neural network (RNN). It will also involve developing a RNN model and a few benchmark models like support vector machine to predict time-series data, whereby in our case bitcoin prices.

Moreover, it is expected that the findings of this research will be able to encourage implementation of neural network and machine learning techniques in analyzing data.

**Literature review**

**Data mining**

Data mining is a technique of acquiring information from a large dataset. We can describe data mining as the process of discovering interesting patterns and knowledge from large amounts of data. They extract hidden and previously unknown patterns, relationships and knowledge that are difficult to detect with traditional statistical methods [18]. The data sources include databases, data warehouses, web and other information repositories [4]. The various data mining techniques are classification analysis, clustering analysis, prediction/regression analysis, association and time series analysis [5].

**Time series Analysis**

Time series analysis involves analyzing data in a sequence form collected over a range of time. In almost every field there is time series data therefore we can say time series is ubiquitous [20]. Time-series analysis is quite challenging especially when the data is huge because it is not easy generate patterns that describe relationships between various data points. However, there may be internal correlations within the data that can be used identify the periodic movements and thus making it possible to forecast. This is what we call autocorrelations between the current and the previous value [20].

Time-series data have some few characteristics that make it possible to analyze and generate various patterns. The features include seasonality whereby if data is observed at monthly or quarterly intervals, we identify seasonality patterns [20]. Like on the instance of bookshop sales, during early January they experience more sales than mid-year and same trend occurs almost every year. They also exhibit characteristics like stationarity and non-stationarity whereby data fluctuates around a certain mean constant [20]. Time-series data is also associated with trends which is the upward or downward movements of data [20] when drawn on a graph and with data like Bitcoin prices it means appreciation and depreciation of the currency. A common characteristic among this kind of data is volatility which is the variance of the data from the mean constant [20]. It is usually in percentage form and the bigger the percentage the more the variance. The technique of analyzing data to extract patterns is called regression analysis.

***Regression analysis.***

Regression is a machine learning technique of using a set of predictors to predict an outcome. Regression analysis finds a relationship between independent variable (commonly called X) and dependent variable (Y) [26]. Various regression techniques like linear regression, logistic regression, support vector machine (SVM) have been applied in different problems but the choice of the algorithm is dependent on various constraints such as the variables used, or the number of parameters, or the number of independent variables [26]. In machine learning the input is the independent variable and it used to compute the output (dependent variable). Each value of the independent is mapped to the output [26].

***Recurrent Neural Network (RNN)***

RNN network is artificial neural network whereby the connection between nodes forms a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior [6]. Basically, RNNs are standard recurrent cells such as sigma cells or tanh cells [11]. They use their feedback connections to store representations of recent input events in form of activations as described by Sepp and Jurgen [7]. RNN predicts its next input based on the previous input. The output from neurons is used as feedback to neurons of the previous layer that is the current output is the next input [8]. This means that RNN has memories in order to remember the previous output.

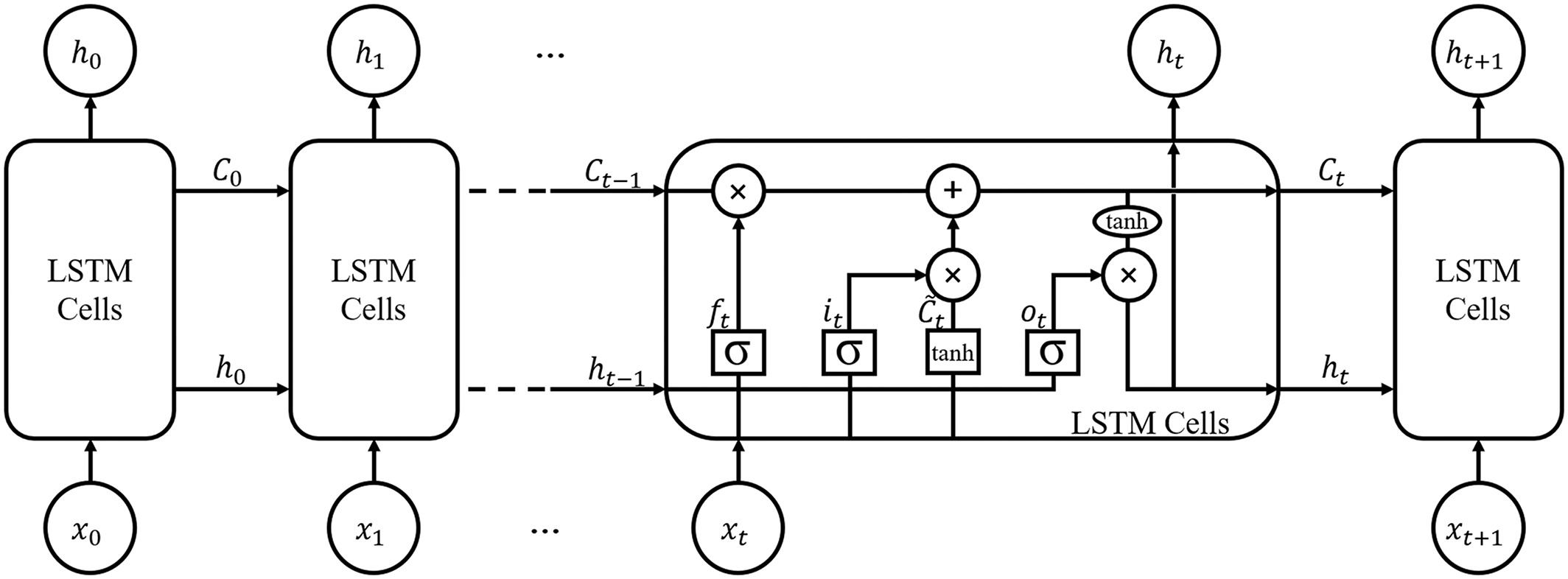
RNNs are powerful enough to make use of the information in a relatively long sequence, since they perform the same tasks for every single element in the sequence, with output dependent on all previous computations [9]. RNN is mainly used to process sequential data. Due to this, RNN is used in speech recognition, text processing and prediction like stock market prices prediction. Most common type of RNN is Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM).

With RNNs, when the gap between related inputs grows, the standard recurrent cells (RNNs) are not capable of handling long-term dependencies thus making learning difficult [11]. During training the model using back propagation, the error signals tend to either blow up or vanish [7]. This is called the vanishing gradient problem whereby the gradient tends to zero. RNNs using LSTM units partially solves this problem, because LSTM units allow gradients to flow unchanged [10].

**Long Short-Term Memory (LSTM)**

As described by Hochreiter and Jurgen, LSTM is a recurrent network architecture with a gradient based learning algorithm designed to overcome error back-flow problems. This means it can learn to bridge time intervals in excess of 1000 steps even in case of noisy, incompressible input sequences, without loss of short time lag capabilities [7]. Since the RNN could not handle long term dependencies, LSTM solved this issue by introducing “gates” into the cell [11]. According to Hochreiter and Jurgen, the importance of gates in LSTM is that they control the error flow to memory cell input connections to avoid input weight conflicts [8].

A LSTM unit is made up of a cell, input gate, output gate and forget gate whereby the cell remembers values over time intervals and the gates regulate flow of information through the cell [10]. In each LSTM cell, six parameters are updated in each step [13]. Below figure represents how LSTM cells looks like.



***Fig 1:*** *LSTM cell architecture*

***Forget gate***

Forget gate is the first layer in a LSTM cell and it decides whether to forget previous data or to push it to the next step [12] and also decides when to access memory cell and when to prevent other units from being disturbed [8]. Forget gate uses a sigmoid function with a linear calculation on the current input and the previous result to decide what to forget. The linear equation has different weights in different steps or cells [13]. The result of this gate is in the range of 0 to 1 and the closer it is to 0 it is forgotten and when it is closer to one it is moved to next step [12]. The forget gate layer is as follows.

𝑓𝑡 = 𝜎 ∗ ( 𝑊𝑓 ∗ [ℎ𝑡−1 , 𝑥𝑡 ] + 𝑏𝑓 )

Where ft, Wf, ht-1, Xt, bf represent forget gate in every cell (first layer), Weights, previous cell output, inputs and biases respectively.

***Input Gate***

The input gate is the second layer that determines which part of data will be stored or remembered. Input data is filtered by the sigmoid function with relation to the inputs, Xt and previous cell results ht-1. A new possible data, 𝐶̃ t is calculated by a Tanh function in relation to Xt and ht-1 [13]. The sigmoid results are multiplied to the tanh layer results and added to the current cell state and the result is the new cell state [12].

All the input gate processes are as follows:

𝑖t = 𝜎 ∗ (𝑊i ∗ [ℎt−1, 𝑥t ] + 𝑏i )

𝐶̃ t = tanh ∗ (𝑊c ∗ [ℎt-1, 𝑥t ] + 𝑏c )

𝐶t = 𝑓t ∗ 𝐶t-1 + 𝑖t ∗ 𝐶̃t

***Output gate***

In the output gate, the input is filtered using the sigmoid function, Ot , and the cell state is filtered with the tanh function. Both results are then multiplied to get the output. The processes can be represented as follows:

𝑜t = 𝜎 \* (𝑊o \* [ℎt-1 ,𝑥t-1 ] + 𝑏o )

ℎt = 𝑜t \* tanh(𝐶t )

**GATED RECURRENT UNIT (GRU)**

GRU is a new RNN model similar to LSTM proposed by Cho Et Al (2014) but has a much simpler architecture by getting rid of the cell state. It uses hidden state to transfer information [14]. It has only two gates, reset, **rt**,and update gate, **Zt** , whereby the reset gate determines how much past information should be forgotten, similar to input and forget gate in LSTM while update gate determines what information to throw away and what to be pushed to the next stage, similar to the output gate [14]. Since GRU does not have output gate, it exposes all its memory contents to other cells as compared to LSTM which has output layer to control which memory content should be exposed. The gate can be described as follows:

**Reset gate:**

rt = 𝜎 (Wxrxt +Whrht-1 +br)

**Update gate:**

zt = 𝜎 (Wxzxt + Whzht-1 +bz)

**Output layer:**

ht’’= tanh(Wxhxt +Whh(rt ⊙ht−1) +bh)

ht = zt(ht−1)+ (1−zt)ht’’

**related studies**

***Amin Azari (2019). Bitcoin Price Prediction: An Arima approach.***

Amin Azari uses an Autoregressive Integrative Moving Average (Arima) based model to predict the close prices of bitcoin. Because Arima model cannot well fit to the whole time period, Amin implement two approaches for identifying (p, g, d), which include one which minimizes the residual sum of squares (RSS) in the model fitting phase, and the second one which minimizes the mean square error (MSE) in the prediction phase [17]. She uses Mean Squared Error to evaluate the performance of the model. From the study, an Arima model with a time window size of 9 was the best with the lowest MSE of 16000. The study also shows that the MSE increases (16000 – 118000) with decrease in the number of the time window (9 - 2). According to Amin, the bitcoin price prediction using close prices could result in large MSE because of the high volatility (high jumps and fall-downs) in the bitcoin price [17]. She also adds that ARIMA model are more suitable in predicting bitcoin prices in sub-periods of the timespan.

***Bitcoin Price Prediction and Analysis Using Deep Learning Models by Muniye, Temesgen & Rout, Minakhi & Mohanty, Lipika & Satapathy, Suresh. (2020)***

In this research Muniye and his team used deep learning techniques by applying Long short-term memory and Gated recurrent units (GRU) to predict bitcoin prices. They used Mean Absolute Percentage Error (MAPE) and Root mean squared error (RMSE) to evaluate their system [14]. The research used 1, 5, 7, 12, 15 window size and the results showed that GRU based models performed better than LSTM models in every window size except 7 with the lowest MAPE and RMSE. GRU also took the lowest compilation time 5 milliseconds against 53 milliseconds with LSTM models [14]. The research focused on predicting bitcoin prices within a short period of time because of the volatility of bitcoin prices.

***Jiang, Xiangxi. (2020). Bitcoin Price Prediction Based on Deep Learning Methods. Journal of Mathematical Finance.***

Jiang also applied deep learning techniques to predict hour-based [14] bitcoin weight prices based on data from the past 24 hour by applying Multiple Layer Perceptron (MLP), LSTM and GRU. The research uses Root mean squared error to evaluate the models. In this research, the 2-layer GRU model was the best [14] with a mean RMSE of 19.020 followed by 2-layer LSM model with a mean RMSE of 19.121 and MLP had mean RMSE of 20.093. Even though GRU was the best, the RMSE was close showing that different model can be applied in different contests.

***Rina Sriwiji, Handini Primandari (2020): An Empirical Study in Forecasting Bitcoin Price Using Bayesian Regularization Neural Network***

Rina and Handini performed bitcoin Bayesian Regularization Neural Network (BRNN) networks. The research used 25 different variables containing blockchain information, macroeconomic factors, and global exchange ratio and the variable selection used linear regression with stepwise method which resulted to fourteen significant variables [16]. They also used trial and error to determine the number of neurons and epochs and the Mean Absolute Percentage Error (MAPE) is used to evaluate the models. The use of 2 and 5 neurons produced the smaller MSE, 1055285 and 2948530 respectively, than the others. Based on the testing data the BRNN model with 5 neurons was the best with a MAPE of 3.7%. and MSE of 23916.

***Emmanuel P, Livieris I.E., Stavroyiannis S., Kotsilieris T., Pintelas P. (2020): Investigating the problem of cryptocurrency price prediction: A deep learning approach.***

Emmanuel and his team conducted a research to compare the most popular deep learning machine learning models, that is CNN (Convolutional Neural Network), LSTM, BILSTM and dense layers (CNN-LSTM and CNN-BILSTM), to traditional ML models (Support Vector Regressor, 3-Nearest Neighbors and Decision Tree Regressor) in predicting the prices of three cryptocurrencies (BTC, ETH, XRP). CNN-LSTM and CNN-BILSTM exhibited the best overall performance with CNN-LSTM exhibiting the best RMSE and CNN-BILSTM exhibiting best accuracy and F1 score [21]. The deep learning models slightly outperformed the other ML models although the experiment showed there is still room for improvement in terms of the model performance. However, Emmanuel and the team describe the models they used as inefficient and unreliable cryptocurrency prices predictors because cryptocurrencies are almost a “random walk” process which makes it difficult for the models to extract the hidden patterns [21].

**CHAPTER 3: RESEARCH METHODOLOGY**

In this project, the preferred methodology follows CRISP-DM (cross-industry process for data mining) methodology which is a very popular and well-proven methodology for implementing machine learning models. Since the methodology is not a blue print for any model, then similar methodology can be applied to implement various models [26].

The CRISP-DM can be described using the following diagram.

**Data**

**Pre-Processing**

**Modelling**

**Alternative Models**

**Model Evaluation**

**Deploy**

The population data in this project will all be about bitcoin. The bitcoin data contains information such as the timestamp, open and closing prices, high and low prices, the volume of bitcoin and finally the weighted price of the bitcoin prices. In this project we will be predicting day by day close price of the bitcoin prices. Therefore, our independent variable will be the close price of bitcoin of the past few days and then the predicted value is the close price of the next day.

According to [26] and an initial prototype model I developed, building the model consists of the following steps:

1. Data collection.
2. Data cleaning.
3. Data preparation.
4. Modelling.
5. Evaluating the model
6. Deployment of the model

For every predictive model in machine learning, the development of the model will always follow the above steps. However, in different models, different techniques are applied in order to achieve the above steps.

***1.Data collection***

In this step data can be collected from various sources like internet. For this project we will collect data from yahoo finances website (<https://finance.yahoo.com/quote/BTC-USD/history?period1=1410912000&period2=1630800000&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true>) which has bitcoin data from the year 2014 to the current date. The data will then be downloaded and hosted so as it can be easily accessible by the notebook via the URL.

***2.Data cleaning***

In this step we remove noisy or unnecessary data. Irrelevant columns and rows will be deleted. To prevent the algorithms from failing, we also need to fill the missing values. To solve the missing value problem, various techniques such as removing the rows with missing data or filling with a random value selected from the dataset or using the mean or median for the column may be used. For this project we will use the previous and the next value adjacent to the missing value to compute the mean and use the mean to fill the missing value. That is if the missing value is at row *k* we take value at row *k-1* and *k+1* and find their average. Since the time column is normally collected in timestamp format, we also format the time to the normal readable format day, month and year (dd-mm-yyyy). We can then visualize the data by plotting the close prices against time. The visualization is used to study the trends and relationship between data in order to decide the type of algorithm to use.

***3. data preparation***

First, we normalize our data. This step involves scaling data to fit to a certain scale while ensuring the differences between dataset values is not distorted. Since data may contain values with very high scale like 20000 and other data with low scale like 1000, this might lead to very poor results while performing data mining. This is because the model will think that because the value is high then it means it has more effect on the outcome as compared to the value with the least value which may not be the case. The models are very sensitive to this kind of change in scale. However, tree-based models are insensitive to scale. Usually, we transform the data to fit between range of 0 and 1. Min-max scaler, standardization techniques and Z-score normalization can be used to achieve this. Minmax scaler will be used in this project. It uses the maximum and the minimum value of the features to scale and the resulting scale values range between 0 to 1 or -1 to 1.

We will then divide the data to training and testing set. We divide in the ratio 80 % to 20% respectively. As the name suggest, the training set will be used to train the model and the testing set will be used to validate the performance of the model. We will then rearrange the training set in the required format. For the X and Y training values, if Y is today then we will take the values of the previous 30 days and set them as X. What I mean by this is that the model will be taking the values of the last 30 days then it will predict the value of the 31st day. So, if we want the Y value of day 100, we consider the last 30 days which are day 69 to day 99. RNN works with sequences, therefore reordering the data to a certain sequence will improve the model performance.

For the final step, the model requires the data to be of a certain shape, so we have to format the data to the required format. For example, in the prototype I developed, the Keras LSTM model requires the input to be in 3D shape, so we have to reshape the data to 3D. The 3D shape can be described as size of the dataset, the number of steps, the number of features in the data. This whole step is applied to both training and testing data.

***4.Modelling.***

In this step, a model of the chosen algorithm is developed and trained. Here the chosen algorithms will be LSTM and GRU models. However, I will also develop several bench-mark models like support vector machine SVM, linear regression and ARIMA model in order to evaluate the performance of the chosen models. Since LSTM and GRU are neural networks then it means we have to determine the number of hidden layers and neurons of the model. Trial and error technique will be used to determine the number of neurons and the layers since there is no rule on how to determine the number of layers nor the number of neurons. This technique involves setting a certain value of neurons and layers and increasing or decreasing the number of neurons or layers appropriately as you evaluate the performance of the model.

To train our model we have to define the number of epochs. Epoch is the number of times the model will iterate the training dataset. We want our model to learn from the data provided, therefore we have to make the model iterate over the training data so as it learns the various patterns in the data. We will also use trial and error method, however, to avoid unnecessary time wastage, we can set a halt when it reaches a certain epoch. This means that if it reaches a certain epoch and there is no improvement in the metrics then it should stop training. This will be helpful to determine the number if training epochs. The metrics we will be minimizing in this project will be value loss or commonly written as “val\_loss”.

Since neural networks are affected by overfitting, then we have to deal with overfitting. Overfitting, is whereby the model learns to much about the training data including the noise until is unable to generalize when fed new data. We will therefore add a Dropout layer so as to avoid overfitting. Dropout is a technique where a certain percentage of neurons of the next layer are dropped. For example, if the dropout is 0.3 then it means 30% of the neurons in the next layer will not be used.

We also need to define the activation function type for the model. Activation is used to determine what neuron will be activated. It introduces the issue of non-linearity between the input and output of a neuron. Various activation such as Relu, sigmoid, Tanh can be used. We will implement the default activation, sigmoid, but we will also use also use Relu so as to compare between Relu and sigmoid which produces smoother results.

Finally, we also need to define the optimizer, the batch size and the validation split percentage. The optimizer is a technique that controls the learning rate of the model while minimizing the loss function. In this project we will use Adam technique as our optimizer. It is very popular and efficient with recurrent neural networks.

The batch size is the number of training dataset to consume per epoch before updating the weights. Large batch sizes lead to more gradient steps which can affect the accuracy of the model. In addition, with larger batch size comes with more time consumed during the learning phase. We will implement trial and error to determine the batch size from either 16, 32, 64 or 128 batch sizes.

Validation split is similar to splitting data to train and test but now we will be dividing the data into training and validation set. If you define the validation split as 0.3 then 30 % of the data is used as the validation set. The model does not train using this data but it uses this data to evaluate its performance after every epoch. This involving using the validation set to calculate validation loss after every epoch.

***5. Model evaluation***

This is a very important step as we have to determine how good was our model. Since the problem is a regression problem, the popular evaluation metrics are Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE). We will use the mean squared error metrics to evaluate the model. The lower the MSE, the closer the predictions are to the actual values.

To evaluate the model, we will also plot the number of epochs against loss and validation loss. This is an important step to determine if the model is overfitting or underfitting. The expected results are that the validation loss should merge with the loss as the number of epochs increase. This means the model is learning. If both of them are diverging then there is overfitting. Also, if the loss is increasing then the model will perform poorly. Ideally, the loss is supposed to decrease continuously then it should remain constant, more like a L-like curve.

Thirdly, remember we split the data to train and test data, therefore we will use the test data to evaluate the model. We will use the model to predict results from the starting date of the test data to the last day and plot the predict results against the actual test data. Since the model did not learn from the test data, visualizing the predicted results against the actual test data is very important in evaluating the performance of the model.

In addition to the above steps, we will also evaluate the models against the benchmark models. This is to determine the accuracy of the model against other algorithms used in various models. We will collect the MSE values from the models and compare the model to determine the model with the lowest MSE.

***6.deploying***

This step involves making the model readily available and user friendly for end users. In this project we will deploy the model as a trading bot. The user interface will be a web-based page. We will also build a web API service, to offer the model prediction services to the interface. Then Interface can then display the predicted price and also display a message on whether to buy or sell your stock prices. We also have to link the model to a data source. The data source is used to retrieve the current bitcoin prices. Therefore, it means we have to link the model to an external API, such as google finance API, that offers historical data on bitcoin prices.

All the model development steps can be summarized using the following flowchart.

inputs

conclusion

Determine number of layers

Train model

predict

Split data (train and test)

modelling

Data preparation

Data cleaning

**Tools**

***Python***

Various python libraries have been developed to implement machine learning algorithms. The libraries reduce the hustle of building the model from scratch and also act an abstraction to the core code of the algorithm. They are also very easy and simple to implement. We will therefore use python programming language to develop our model. We will use several libraries in order to achieve every step of developing the model as described in the previous chapter. The libraries are described below.

1. **Keras** – it is a library that acts as an interface to the TensorFlow library. It provides an interface to access the neural networks. It enables fast implementation of the nueral networks.
2. **Pandas –** it is a python-based tool that is used for data analysis and data manipulation. Pandas we be used to retrieve the training data from files as well as manipulate data such as data cleaning.
3. **Numpy -** This tool is also used for data analysis. The great advantage it has is that it has large, multidimensional arrays and matrices and it facilitates complex mathematical operations on a large dataset efficiently.
4. **Matplotlib –** This tool is used for visualization of data. We will use it to plot graphs to visualize the actual bitcoin prices against the predicted prices.
5. **Scikit-Learn –** This tool is also used for data analysis and it can be used to offer operations such as the normalization operation (such as MinMaxScaler ) and the metrics functions (such as MSE ) used to evaluate the model.

For the notebook editor we will use Google Colab, since they offer GPU and TPU services. We will build our user interface using Javascript, the API services we will develop using FLASK which is a python library.

**CONCLUSION**

Neural networks, same as other machine learning models, can be used in regression analysis. Because RNN are used with data in a sequence, it therefore means they can be used for time series analysis. The RNN model, through the process of learning, can find the relationship between inputs and outputs and then it modifies its weights in order to capture the pattern. The main aim of learning to is minimize the function error between the predicted outcome and the actual outcome. The lower the error, the more accurate the predictive model is. However, to the model cannot be 100% accurate. This is because of issues like the gradient descent or overfitting, but the issues can be mitigated during training by using trial and error techniques to develop a model with the lowest loss possible. We will alter the layers and neurons in order while evaluating the model.

References

1. Michael Crosby, Nachiappan, Pradan Pattanayak, Sanjeev Verma , Vignesh Kalyanaraman. (2016). BlockChain Technology: Beyond Bitcoin.
2. Rina Sriwiji, Arum Handini Primandari. (2019). An Empirical Study in Forecasting Bitcoin Price Using Bayesian Regularization Neural Network. ICSA 2019, August 02-03, Bogor, Indonesia. DOI: 10.4108/eai.2-8-2019.2290515.
3. Christian Morquecho. (ND). What is Bitcoin? Why the digital currency became popular? Retrieved July 27, 2021 from <https://www.entrepreneur.com/article/368694>.
4. Jiawei Han, Micheline Kamber, Jian Pei. (2012). Data mining: Concepts and Techniques. Morgan Kaufmann Publishers. Waltham, Ma 02451, USA.
5. M. Nikhil Kumar, K. V. S. Koushik, K. Deepak (2018). Prediction of Heart Diseases Using Data Mining and Machine Learning Algorithms and Tools. 3(3), 1-12.
6. Wikipedia. Recurrent Neural Network Retrieved July 27, 2021 from <https://en.wikipedia.org/wiki/Recurrent_neural_network>.
7. Sepp Hochreiter, Jurgen Schmidhuber. (1997). Long Short Term Memory. Neural Computation 9(8): 1735 – 1780. [doi](https://en.wikipedia.org/wiki/Doi_(identifier)):[10.1162/neco.1997.9.8.1735](https://doi.org/10.1162%2Fneco.1997.9.8.1735). [PMID](https://en.wikipedia.org/wiki/PMID_(identifier)) [9377276](https://pubmed.ncbi.nlm.nih.gov/9377276). [S2CID](https://en.wikipedia.org/wiki/S2CID_(identifier)) [1915014](https://api.semanticscholar.org/CorpusID:1915014).
8. Sepp Hochreiter, Jurgen Schmidhuber. (2019). Optimization of Ann Architecture: A Review on Natire-Insipred Technique. Doi: 10.1016/B978-0-12-816086-2.00007-2
9. Runjie Zhu, Xinhui Tu, Jimmy Xiangji Huang. (2020). Deep learning on information retrieval and its applications. Doi: [10.1016/B978-0-12-819764-6.00008-9](https://doi.org/10.1016/B978-0-12-819764-6.00008-9)
10. Wikipedia. Recurrent Neural Network Retrieved July 27, 2021 from <https://en.wikipedia.org/wiki/Long_short-term_memory>.
11. Yong Yu, Xiaosheng Si, Changhua Hu, Jianxun Zhang; A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Comput 2019; 31 (7): 1235–1270. doi: <https://doi.org/10.1162/neco_a_01199>.
12. Akın, Cihan & Kacar, Umit & Kirci, Murvet. (2019). Twins Recognition Using Hierarchical Score Level Fusion.
13. Zhongrun Xiang, Jun Yan, Ibrahim Demir (202): A Rainfall-Runoff Model With LSTM-Based Sequence-to-Sequence Learning: Water resources Research 56 (1): doi: https://doi.org/10.1029/2019WR025326
14. Muniye, Temesgen & Rout, Minakhi & Mohanty, Lipika & Satapathy, Suresh. (2020). Bitcoin Price Prediction and Analysis Using Deep Learning Models. 10.1007/978-981-15-5397-4\_63.
15. Jiang, Xiangxi. (2020). Bitcoin Price Prediction Based on Deep Learning Methods. Journal of Mathematical Finance. 10. 132-139. 10.4236/jmf.2020.101009.
16. Rina Sriwiji, Handini Primandari (2020):An Empirical Study in Forecasting Bitcoin Price Using Bayesian Regularization Neural Network: ICSA: EAI: DOI: 10.4108/eai.2-8-2019.2290515
17. Amin Azari (2019). Bitcoin Price Prediction: An Arima approach. [arXiv:1904.05315](https://arxiv.org/abs/1904.05315)**[cs.SI].** Retrieved from [https://arxiv.org/abs/1904.05315 on August 20](https://arxiv.org/abs/1904.05315%20on%20August%2020), 2021.
18. Mutyala, Nikhil Kumar & Koushik, K.V.s & Krishna, K. (2018). Prediction of Heart Diseases Using Data Mining and Machine Learning Algorithms and Tools. 10.13140/RG.2.2.28488.83203.
19. Apeksha Shewalkar, Deepika Nyavanandi, Simone A. Ludwig. (2019). performance evaluation of deep neural networks applied to speech recognition: rnn, lstm and gru. JAISCR, 2019, Vol. 9, No. 4, pp. 235 – 245. Doi: 10.2478/jaiscr-2019-0006
20. Terence C. Mills. (2019). Applied Time Series Analysis. Loughborough, United Kingdom: Candice Janco.
21. Emmanuel P, Livieris I.E., Stavroyiannis S., Kotsilieris T., Pintelas P. (2020): Investigating the problem of cyptocurrency price prediction: A deep learning approach. In: Maglogiannis I., iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innnovations. AIAI 2020. IFIP Advances in Information and communication technology, vol 584. Springer, Cham. <https://doi.org/10.1007/978-3-030-49186-4_9>
22. Pavlyshenko, B.M. Machine learning Models for sales Time Series Forecasting. Data 2019, 4, 15.https://doi.org/10.3390/data4010015
23. Bogdan O, Stephan C.C. (2014): Time series forecasting using neural networks : <arXiv:1401.>1333**[cs.NE].** Retrieved from [https://arxiv.org/abs/1401.1333 on September](https://arxiv.org/abs/1401.1333%20on%20September%20) 04, 2021.
24. Grossi, Enzo & Buscema, Massimo. (2008). Introduction to artificial neural networks. European journal of gastroenterology & hepatology. 19. 1046-54. 19.1097/MEG.0b013e3282f198a0.
25. Maxim Naumov. (2017): Feed forward and recurrent neural networks backward propagation and hessian in matrix form: <arXiv:1709.>06080**[cs.LG].** Retrieved from [https://arxiv.org/abs/1709. 06080 on September](https://arxiv.org/abs/1709.%2006080%20on%20September%20) 04, 2021.
26. Anshey.S, Narandra.M. (2020).Prediction of Crypto Currency by mulitiple linear regressions using anaconda software. European journal of Molecular & clinical medicine, 7(4), 76-81.