**Project Final Report**

**Miao Zheng**

1. **Introduction**

There is an increasing number of people entering the cryptocurrency market in the recent years as they learned some of people made a lot of money in this new market. However, the price of cryptocurrency is highly unstable and there’s a relationship between high risk and high return. So, if we would like to enter the market and gain some profits, we should do some research for the market economic trend and also use some technologies to predict price/return that help us to make a better decision.

In this final paper, I’m going to call API to extract the price of Binance as an example (one of the most popular cryptocurrencies in the market). My objective is to investigate and analyze the overall trend of change in Binance close price from 2018-12-25 to 2023-02-02 and predict the Binance coin price after 2023-02-02 (the latest 60 days). I would like to learn and try neural network we learned in class and see how it improves the traditional statistical model in predicting stock/cryptocurrencies price.

As we have a time-series dataset, which is continuous, I’ll use the traditional ARIMA model as the reference model and test how LSTM performs with different loss functions and optimizers that can improve the price prediction. Here I choose MSE and RMSE as our evaluation metrics.

After building several models, we found ML does a better job than ARIMA model and for LSTM - loss function MSE will perform better than MAE and based on the models we build, LSTM(100) with MSE and ADAM optimizer shows the best performance with lowest MSE and RMSE.

**Note:** Binance is restricted in Ontario, so no matter how prediction goes, it’s just a project for the purpose of practice

1. **Methods**

Let’s take a look of our dataset first – This is a continuous time-series model, close price depends on time. Although we have 7 columns in total, here we will focus on close price and only interested in how close price changes as time goes by.Chart

Description automatically generated

* 1. ***Traditional Time Series Model - ARIMA***

From the plot above, we’ll see the growth is not stable, changing dramatically without constant variance. Then we’ll consider the traditional statistical ARIMA model, which predicts well for non-stationary function.

**For traditional time-series model, the most important concept is stationarity as we assume each point(day) is independent of one another when we predict time-series model**. In this case, we will check stationarity of the dataset and transform the data to stationary ones if it’s non-stationary, to predict with statistical time-series model. The common transformation methods are differencing and logarithmic transform.

***ARIMA(p,d,q) equation works as below: d is the differences needed for stationarity (see pic below)***

Text, letter

Description automatically generated

Here’s how ARIMA model works – after we transform the dataset to stationary data, we’ll check ACF and PACF plots to determine the parameters p and q, or we can use autoarima() function directly to choose the best model fit. After checking model diagnostic, we can predict the price with selected ARIMA model.

Diagram

Description automatically generated

**Limitation:** there are some limitations when we use ARIMA model. 1. ARIMA model is not suitable for multivariate time-series data, that means here we only consider the relationship between close price and time and make prediction for future close price. 2. ARIMA model is a statistical model, where we assume it’s normal distribution. However, for some cases, it may not be normal distribution, i.e. stock & cryptocurrencies dataset. 3. It works well for short-term prediction, but for the long-term prediction, it does not perform well.

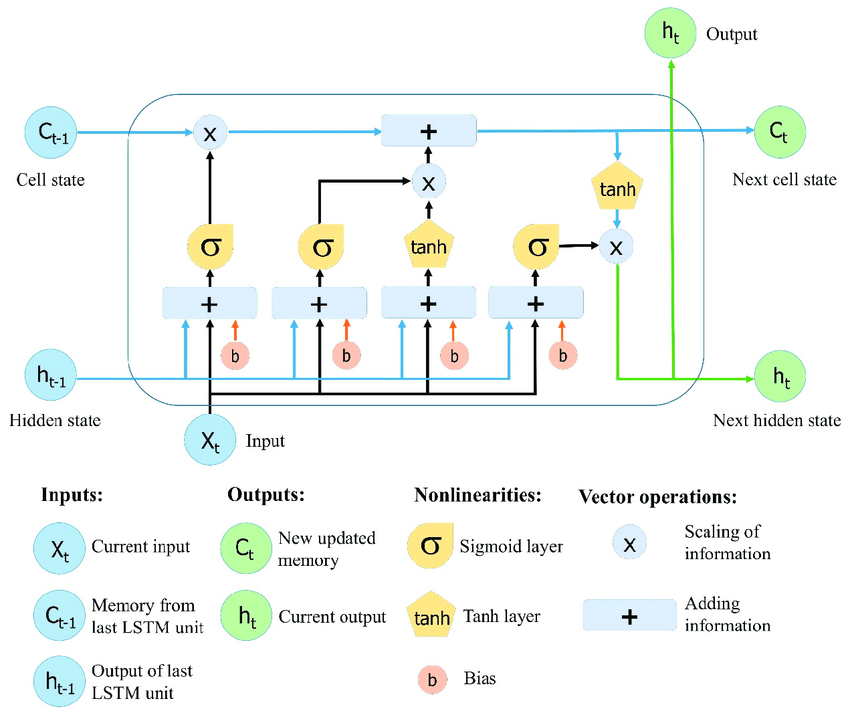
* 1. ***Neural Network - LSTM model***

Based on the limitation of ARIMA model mentioned before, I would like to find a method which performs better in the long term.

Compared with traditional time-series models, neural network learns features and dynamics directly from the data, which speeds up the data preparation. Unlike ARIMA model, which predicts well in the short term, but less accurate in the long term, a special type of recurrent neural network model – **Long Short-term Memory (LSTM)** can address this problem well.

**There are 4 main components in LSTM model:** **the forget gate** controls the information from the previous step to the current step, **the input gate** controls how much the new information from the current step will be added to the cell state, **cell state** here means ‘memory’, containing information from both previous and current time steps, and finally **output gate** controls the information from cell state used to produce output at the current time step.

***Here’s how LSTM works:***

(Picture from the website - referemce8)

***LSTM equations:***

Chart, box and whisker chart

Description automatically generated

(LSTM equations from website – reference 9)

Unlike traditional feedforward neural network models, processing all input at one-time step and producing a single output as the next step, with the help of these 4 components, LSTM will take input at one-time step and use it to influence their output at a future time step – so that means **it will learn from sequences of data without treating each point independently but retaining the previous useful information in the previous points.**

**fixed the vanishing gradient problem**

LSTM was designed to overcome vanishing gradient problem, which is difficult to learn the long-term dependency problem in traditional RNN, by introducing constant error carousel, as it maintains the internal activation with a recurrent connection of fixed weight 1.0, which may be reset by the forget gate and scaled by input and output gate.

**Limitations of LSTM**. First, it’s complicated than other models and hard to remember dependencies that are many steps removed from the current input with forget gate. Second, limited context window size. But in both cases, we can resolve by using more data to train a larger LSTM with more cells.

**What to do in our project**

In our project, I will choose 60 days as our window period, i.e. we’ll learn the first 60 real days value and predict the 61th day. The previous prediction will not influence the current prediction. The next step before model building is filing the missing values, rescaling, and doing normalization to help optimization algorithm converge faster and get a better performance.

After that, I’ll try different optimizers, loss function and parameters, and with the help from evaluation metrices: MSE and RMSE and plot, I’ll choose the best model fit.

1. **Results**

I’m going to try LSTM with different parameters and compare with ARIMA model to show the difference and improvements for each model.

* 1. ***Statistical Model – ARIMA***

ARIMA model, as we mentioned above, after we did log difference (close price), we got somehow constant variance. Although there’re still some outliers, the general trend is around 0 with constant variance which is stationary.

***Chart

Description automatically generated***

Here I use autoarima function to choose the best model – here we get ARIMA(1,1,1) as our best model.

**Evaluation metrics: model diagnostic & 95% CI**

And we use AIC/BIC and check model diagnostic (QQ plot, residuals,etc) to check the model fit. Chart, line chart

Description automatically generated

Although the normal QQ-plot has outlier at the end of both tails, we can still get some sense of the model fit, since it makes sense in the reality. After that, we’ll plot the prediction of log(close price) for test dataset. Here we predict the whole test set (300 days) , which is a long-term prediction and we can see the log(price) is still in 95% CI, but as time goes by, the gap between the real value and prediction increases, which implies that ARIMA model does a good job for short-term, but not too good for the long-term prediction.

Chart

Description automatically generated

* 1. ***LSTM with different parameters***

Then we’ll start the core concept in this project – how LSTM works. In this project, we’ll keep **the activation function same and let activation function = tanh(x)** and want to figure out how each model performs with different parameters.

***Evaluation Metrics:***  MSE and RMSE for all models below – the lower MSE/RMSE, the better the model fit.

***Note:***  1. Here, we use MinMaxScaler() to do the normalization and rescale for the features.

2. since we choose random points to train the model, the evaluation metrics (MSE & RMSE) may change in each run.

* + 1. **Optimizations**

1. ***Adam Optimizers – default optimizer***

Adam Optimization is an extension to stochastic gradient descent procedure to update network weights iterative based in the training dataset.

***Advantages of Adam Optimization:***

* Efficient Computation
* Little memory requirements
* Good for non-stationary data

**Result:** If we use LSTM(100) with Adam Optimizer and MSE loss function, we get MSE is 209.6093 and rmse is 3.3606. From the prediction plot below, we see there’s a pretty good prediction for a whole test dataset (300 days).

Graphical user interface, chart

Description automatically generated

**When we zoom in the test part, we can see the actual value and predictions are very close.Chart, line chart, histogram

Description automatically generated(figure 1)**

1. ***Gradient Descent with Momentum Optimizer***

Gradient Descent with Momentum is an extension to the gradient descent that accelerate the optimization process and allows the search to build inertia in a direction in the search space and overcome the oscillations of noisy gradients.

**Result:** After changing optimizer to momentum and keep other conditions same as before, we’ll get mse = 908.244 and rmse is 6.4133, which is a bit higher than the same conditions under adam optimization. Chart, line chart, histogram

Description automatically generated

**(figure 2)**

Based on the plot above, we can get a clear sense that adam optimizer may be better than momentum in LSTM for time-series prediction.

* + 1. **Loss Function**

Now, we’ll check the difference between loss function and keep other conditions same. Here we’ll use LSTM(100) and Adam Optimizer, as adam does a better job shown before.

***A loss function is a measure of how accurate the ML prediction is.***

1. ***Mean Squared Error (MSE)*** Diagram, schematic

   Description automatically generated with medium confidence

***Advantages of MSE:*** there’s no outlier prediction if we use MSE.

**Result:** Just as figure 1 shown above, model with Adam & MSE has mse = 209.6093 and rmse = 3.3606.

1. ***Mean Absolute Error (MAE)***

***A picture containing text, clock

Description automatically generatedMAE takes the average of absolute value od errors***

**Result**: So under the model with Adam optimizer & MAE loss function, we get MSE is 276.23074 and rmse is 9.18 which is larger than Adam&MSE. Then let’s plot and see the difference between actual value and predictions. Chart, histogram

Description automatically generated

From the plot above, we see the prediction is just okay, which matches the result we get from MSE and RMSE above.

1. **Discussion/Conclusion**

In summary, we showed LSTM did a better performance than ARIMA model in the long term. Although we use log(close price) in ARIMA model, we can still find the overall trend that prediction gap will become larger as time goes by.

We also showed how model performs after changing only 1 condition above for LSTM model. Although there’re a lot of things we can test, i.e. different activation function (sigmoid, softmax,etc), different parameters inside the model (number of neurons, dense output,etc), here we haven’t shown all models in this report.

**Based on the models we build in 4. Results, we found LSTM(100) with Adam optimization and MSE loss function (model1) performs well among all models.**

**New data:** Then I extract the latest 60 days (data from 2022-02-03 to 2023-04-01) and fit model 1 to see how well our prediction is. Chart, line chart

Description automatically generated

(Example of actual price and predictions)

Table

Description automatically generated

We get MSE = 226.8 and from the plot, we can see the gap between true value and prediction is okay, around $10, which shows the accuracy makes sense and performance follows the train dataset. But overall, our prediction is higher than the actual price.

**Conclusions & Further Steps:**

Although it’s not suitable to mention in this report, I realized that predicting price is a bit meaningless for our project. We want to make a better decision and gain more profits in cryptocurrencies market based on our prediction, but as results shown above, there’s around $10 - $30+ difference between the close price and prediction per day, which is very large in the real life. Although our model shows good performance in our train/validation dataset, when we introduce new dataset (the latest 60 days data here), the performance drops and we can see the short-term trend clearly.

In this case, for my further step, I would like to try to predict the return daily, which may need to use all features instead of just close price only. I still need to do some research and see how to calculate the return, but I believe we can still use neural networks and hope we could make a better decision based on the predictions of returns instead of price.

1. **Reference**

**1.** <https://www.cryptocompare.com/coins/guides/how-to-use-our-api/> (the website I’m going to call api)

2. what is LSTM: [https://www.knowledgehut.com/blog/web-development/long-short-term-memory](https://www.knowledgehut.com/blog/web-development/long-short-term-memory%20)

3. ARIMA model equation (pic): <https://people.duke.edu/~rnau/411arim.htm#pdq>

4. LSTM explanation <https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9>

5. <https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/>

6. how to solve gradient vanishing: <https://www.codingninjas.com/codestudio/library/solving-the-vanishing-gradient-problem-with-lstm#:~:text=Backpropagation%20Through%20Time%20in%20LSTM,-On%20the%20k&text=The%20gradient%20is%20computed%20across,total%20error%20gradient%20to%20vanish>.

[7. https://www.researchgate.net/figure/Illustration-of-LSTM-memory-block-with-one-cell-Constant-Error-Carousel-CEC-maintains\_fig1\_301404520](https://www.researchgate.net/figure/Illustration-of-LSTM-memory-block-with-one-cell-Constant-Error-Carousel-CEC-maintains_fig1_301404520)

8. how LSTM works – pic <https://www.researchgate.net/figure/The-structure-of-the-Long-Short-Term-Memory-LSTM-neural-network-Reproduced-from-Yan_fig8_334268507>)

9. LSTM equations pic- <https://towardsdatascience.com/tutorial-on-lstm-a-computational-perspective-f3417442c2cd#:~:text=LSTM%20equations,-The%20figure%20below&text=The%20LSTM%20has%20an%20input,the%20next%20time%20step%20LSTM>.

10. adam optimizer introduction: <https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/>

11. momentum introduction: <https://machinelearningmastery.com/gradient-descent-with-momentum-from-scratch/#:~:text=Momentum%20is%20an%20extension%20to,spots%20of%20the%20search%20space>.