Machine Learning Engineer NanoDegree Capstone Project Proposal

Predicting cryptocurrency prices with Machine Learning.

Domain Background

I was always interested in algorithmic trading and wanted to write my own automated trading strategy. However, I think that stock/bond and FX markets are oversaturated with institutional traders with far greater access to better technology, much larger trading capital and better equipped talent pool for me to stay competitive. The cryptocurrency space which has experienced a significant growth in the last few years with over 2000+ new coins being created since 2013 are considered too volatile and risky for institutional traders might be a perfect opportunity to exploit the trading opportunities that are no longer available in more conventional markets. Elendner et al. (2017) proved that cryptocurrencies are uncorrelated with each other and with traditional assets.

Problem Statement

The purpose of this project is to create a deep learning model that is able to predict the Price of the Cryptocurrency Coin with the use of historical price data available online. The idea is to build a coin price predictor which would be able to use historical market data such as Opening price, Daily Highs and Lows and Trading volume over some period of time to predict the closing price of a coin at the end of the day. The output of the model would be used to make a trading decision for buying or selling of one unit of coin for one day. I will assume that there is no market spread and no trading fees associated with the transaction and that short selling (betting that the market will fall) is allowed with no additional fees.

Datasets and Inputs

For the dataset I will use the daily coin price from https://coinmarketcap.com (https://coinmarketcap.com), which I will obtain via their public API with the help of Pandas "read_html" function and convert it to a csv format. I will make the data loader abstract enough use with any of the 2300 coins available on the website, but we will use the bitcoin as our currency of choice, simply because it has the most historical data available.

1) The original dataset contains seven variables.

- Date: the trading date
- Open: the opening price
- High: the highest price for the day
- Low: the lowest price for the day
- Close: the latest price in the date range
- Volume: number of coins trader in that day
- Market Cap: number of coins * value

2) I am going to use all of the data available on the website from 2013-04-28 to 2019-09-28 which gives us 2345 observations of 7 variables. I will use Close price as the target variable and the remaining as my independent variables.

3) The data has to be preprocessed before we can use it for modelling. First thing I will have to do is to make the data stationary and remove any possible trends and seasonality. This can be achieved by taking the lag of data and using a natural log to remove smooth the data a bit. Since the coins price changed quite

significantly over the years from the min of 65.53 to 20089.0 USD it will be a good idea to normalise the data from 0 to 1.

4) I want to do some feature engineering and adding the volatility of the times series and add 10, 30, 60 day moving averages of (Open, High, Low) as my independent variables.

5) I will convert the data into time series sequences to before feeding to the model. I will consider sequence length from 1 to 30 days.

6) I will use the last 90 days as my testing set and will split the remaining 2255 observations into my training/validation with the ratio of 80/20.

Solution Statement

I will use a recurrent neural network for this project, specifically a Long Short Term Memory Network as the model. The Closing Price of Bitcoin as the target variable and the other variables as prediction variables. Based on the prediction, the algorithm will decide to either buy or sell one unit of bitcoin f at the opening price and results of the trade will be evaluated based on the actual closing price of the coin at the end of the day. I am assuming no additional fees apart from the coin price itself. I will do this for the latest 90 days of the available data and compare the results to the simple buy and hold strategy.

Benchmark Model

For this project I decide to compare the results of the daily trading of one bitcoin to the simple buy and hold strategy. I select the last 90 days as our validation sample and will purchase one coin at the beginning. The difference in price at the end of the period will be the profit/loss of the buy and hold strategy.

Evaluation Metrics

In order to evaluate the performance of the model we will use Mean Squared Error. MSE, which shows the average deviation from the mean true value is considered a go-to approach when it comes to evaluation of model performance for regression problems. I will use ADAM as our optimizer. I picked ADAM because it achieves good results fast and slightly outperforms other algorithms on most problems. In order to compare the performance of the two trading strategies we will use the rate of return which is the (current price - starting price) / starting price.

Project Design

1) Getting the data.

I will use python's pandas library to download the data from https://coinmarketcap.com (https://coinmarketcap.com) public API and convert it to CSV format. I will use all of the available data from 2013 to 2019, leaving the last available 90 days as the validation dataset for our trading strategies. The dataset contains seven variables Open, close, High, Low, Volume and Date. I am planning to use the daily closing price (Close) as the target variable.

2) Prepressing the data.

Before training a neural network we will have to pre-process the data. The financial time series are usually non stationary, so I am planning take the natural logarithm of the price difference ln(Price t / Price t-1), this will help in eliminating any trend and seasonality.

Finally, because neural networks uses activation functions that squash the input variables to their output interval, I would like to scale the input data so the neurons learn faster, see LeCun et al. (1998). I am planning to use the Min/Max scaler to normalise the input variables from 0 to 1.

3) Generate Sequences.

Since we are dealing with LTSM I would have to convert the data into time series sequences. The optimal length of the sequence will have to be established empirically through tuning. I am thinking of trying out the values from 1 to 30 (one trading day to one month). The procedure for sequence generation would be as following: Let's assume we picked the length of 5 days. Set the rolling window of 5 days long. Move the 5 day window along the timeseries one day at the time and record the values as the input features for the next days target variable.

4) Training the model.

I want to use the LSTM-RNN model. I am planning to use 3 layers of LSTM with 512 neurons per layer followed by 0.25 Dropout after each LSTM layer to prevent over-fitting and finally a Dense layer to produce our outputs. I am thinking of using 'tanh' for my activation function and Mean Squared Error for my loss and 'adam' as my optimizer.

5) Building a trading strategy.

I will have to incorporate the results of the model in to a trading strategy. I am choosing the simplest approach. I will treat the this as a classification problem: If predicted Close price Ln(Price t / Price t-1) > 0, buy one coin and hold until the end of day. If predicted Close price Ln(Price t / Price t-1) < 0 sell one coin and hold until the end of day. The difference between the predicted and the actual Close price would be the Profit/Loss for the day. I will do this for the last 90 days of the available data and compare the result to the