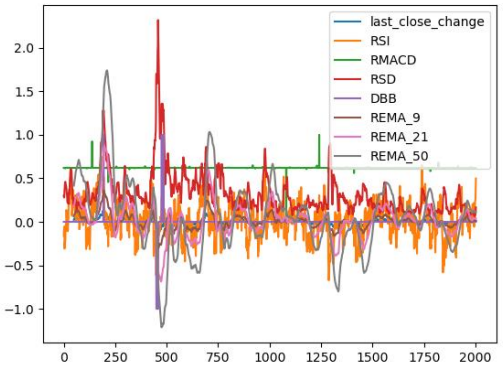
Cryptocurrency profit prediction and bot integration

TradeBot

TradeBot is a cryptocurrency trading bot built on the Binance API. The TradeBot codebase is what enabled us to collect all of the data for this project and is where we planned to use the trained model. It was our spring 2020 OOP course project.

Data

We collected 2 GB of raw aggregate trade data for the BTCUSDT pair from 2019.01.01 to 2020.01.01. This was then processed to a 10 GB CSV containing time, price, and calculated values for 5min change, RSI, RMACD, RSD, REMA 9, REMA 21, REMA 50



Goals

We wanted to create simple learners that are able to predict some aspect of the timeseries data we had collected. We decided to try to predict some aspects of this data. The final goal was to earn more profit with a bot using these models than the market baseline.

Development and Training

We used the python libraries *scikit-learn* and *keras* to implement the learners. For the transformer model we used the custom keras layers from the article 'Stock predictions with state-of-the-art Transformer and Time Embeddings'. We used CUDA (cuDNN) to train our tensorflow models. To monitor training progress and check performance we used Weights & Biases. For hyperparameter tuning in *scikit-learn* we used *sklearn RandomizedSearchCV*.

Models

XGBoost

Extreme Gradient Boost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable . It implements machine learning algorithms under the Gradient Boosting framework. We used it to predict the minimum price relative to the current price in the next 100 timesteps.

Random Forest Regression

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap Aggregation, commonly known as bagging. We used it to predict the maximum price relative to the current price in the next 100 timesteps.

Transformer + TimeEmbedding

The 2017 paper 'Attention Is All You Need' introduced a novel sequence-to-sequence machine learning architecture called Transformer. As the title indicates, it uses the attention-mechanism. Typically, Transformers consist of an encoder and decoder, but we only used the encoder to create a single prediction (sequence-to-vector). This is supplemented by embedding the notion of time with a layer based on the paper 'Time2Vec'. We used this model to predict the change in 5 min candle closing price.

Results

As we expected, and as many who have tried to conquer this problem before have found – this problem was simply too difficult for our simple learners. Even the state-of-the-art Transformer + TimeEmbedding model was not able to predict 5 min candle closing values (1 candle ahead). Our models tended to get stuck in local minimums of the loss function, which resulted in behaviors such as predicting a constant or predicting an input multiplied by a constant.

The reason for this is likely that any more complex or varied predictions would result in too much increase in loss to be attempted. Many of the articles and papers we read discussed that this could be due to the complexity of commodity market interactions resulting in seemingly random changes in price. Only when applying smoothing (such as rolling mean) would the model be able to make seemingly original predictions. This is demonstrated in RSI prediction model. RSI has an exponential component which is smoothed over N candles (in our case 14). The larger the model, the smaller this smoothing window needed to be.

Another observation that we confirmed was that this problem could be solved with just more compute resources and larger models (as seen in the improvement from GPT-2 to GPT-3 in NLP). Our useful Transformer + TimeEmbedding models started to run into the limits of our hardware (11 GB of VRAM) during training. It might be possible to predict features with an even smaller smoothing window if we trained a larger model using something like Lambda GPU Cloud.