Crypto Trading Bot

Jacob Jalomo, Joshua Bond, TJ Azodeh

Motivation & Summary

For our project, we wanted to apply Deep Learning and Algorithmic Trading to create an automated Trading Bot that we could use in real-life.

We hypothesized that we could use a RNN to accurately predict buy and sell signals on various Cryptocurrencies.

Project Map

Data Collection / Processing

Kraken API for data collection.

Cleaned historical data.

Engineering Variables

Computing predictive variables.

Defining Y.

Principal component analysis.

Scaling variables.

Model Analysis / Backtesting

Building Neural Network.

Fitting multiple models with different PCs.

Comparing Accuracies.

Testing model on unseen data.

Data Collection / Processing

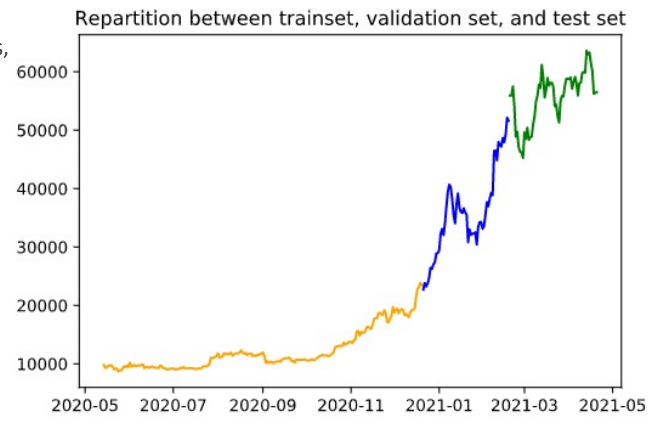
Kraken API

- Our model utilizes Kraken API to pull historical data for Cryptocurrencies (BTC)
- Minimal data cleaning required because Kraken pulls clean data.
- Open, high, low, close all used as features in deep learning model.

Variable Engineering

- Predictive features computed for the model:
  - EMA, RSI, JWWMA, ATR

  (Also included candle shadow and body size, min, max, volume, modulos at differing levels, differences, percentage change, week day, and day)
- Results calculated based on stoploss of -5% and take profit calculated based on +12%.
- Variables created using multiple window sizes: (8, 13, 21, 34, 55, 89, 144, 377)
- Train, test, and validation sets created after variables computed.
- Variables scaled for use in model.

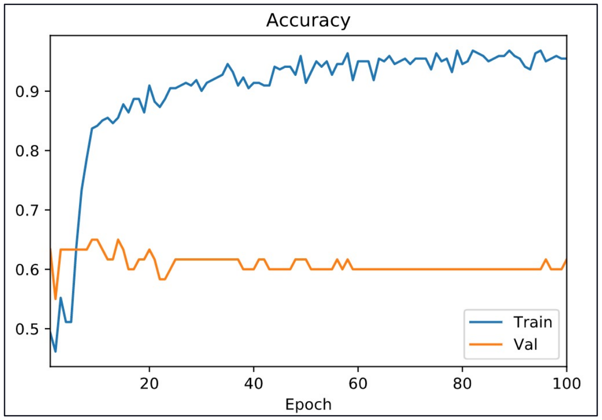


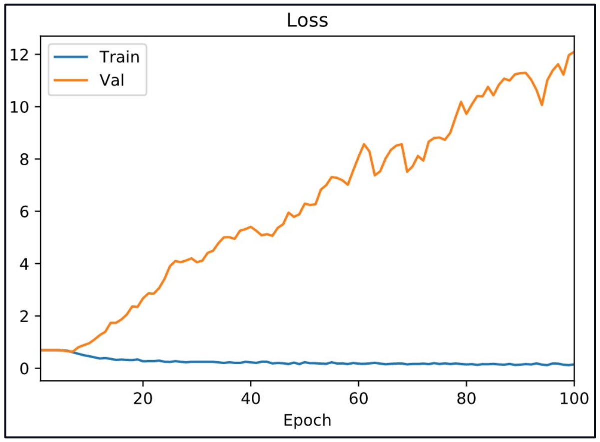
Model Architecture

- Built a function to create multiple models at once using different numbers of principal components (7 models created).
- We used a Sequential model with Dense layers and the relu activation function to allow for greater adaptability in the model.
- Four hidden layers in each model
- Output layer used Sigmoid for activation
- Neural Network was compiled using the Adam optimizer and binary cross-entropy for loss.

Model Analysis

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_30 (Dense)             (None, 32)                1312
_________________________________________________________________
dropout_24 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_31 (Dense)             (None, 32)                1056
_________________________________________________________________
dropout_25 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_32 (Dense)             (None, 16)                528
_________________________________________________________________
dropout_26 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_33 (Dense)             (None, 16)                272
_________________________________________________________________
dropout_27 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_34 (Dense)             (None, 1)                 17
=================================================================
Total params: 3,185
Trainable params: 3,185
Non-trainable params: 0
_________________________________________________________________
None
```





Model Results

Accuracy Report

The accuracy report shows how accurate our model was at predicting a take profit scenario (1 vs 0 in our result column). When predicting only 1, this is the accuracy.

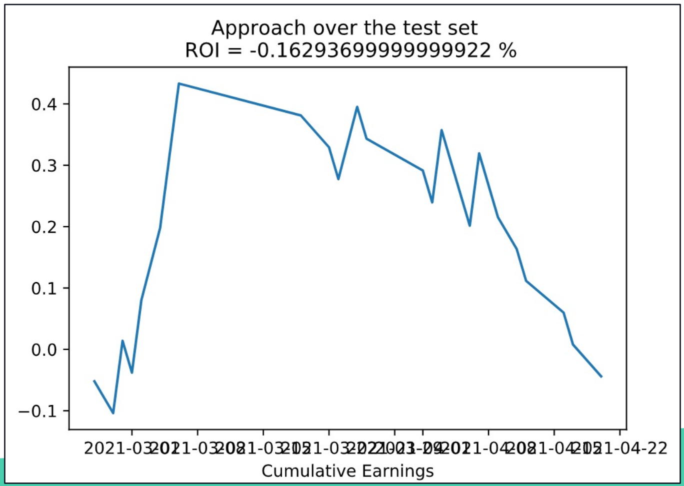
nPCs	Accuracy
10	0.6244444444444450
15	0.6155555555555560
20	0.5133333333333330
25	0.5
30	0.5533333333333330
35	0.5577777777777780
40	0.4866666666666670

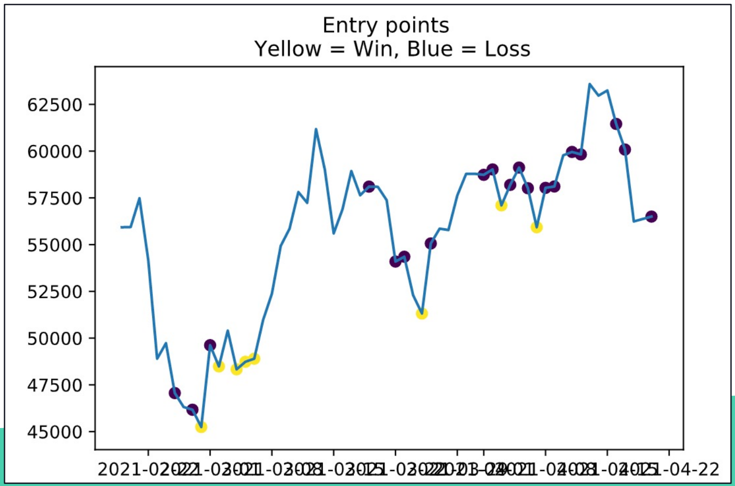
Backtesting

- As a final step of our process, we backtested the models on data that was never before seen in the model.
- This allowed us to test our model and strategy to see if it is accurately predicting on new data.
- This was used on the last 17.5% of the original data.

Model Results

The result of our models was a negative ROI of 0.16%. This leads us to believe that there is room for growth in the model and that more data and testing is needed in order to increase the accuracy of the model.





Postmortem / Q&A

Some unexpected issues we experienced included:

- Finding a method to run, save, and load multiple models
- We used the pickle package in Python but had to seek out a fix online to enable pickling of keras models as it is not supported by default
- The models are not accurate, given more time we would adjust the model architecture more to see if that makes an impact on the accuracy of the model.
- With more time, we would also deploy our model across a larger range of Cryptocurrencies other than just BTC (ie. Etherum, Ripple, etc.)

Questions?