Andrew Marks

Advanced Big Data Final

Math 389L

Quick recap of project:

Following the paper “Algorithmic Financial Trading with Deep Convolutional Neural Networks: Time Series to Image Conversion Approach”, I used Kraken the Cryptocurrency exchanges API to get minutes data from January of 2014 through March of 2020. The data set is comprised of the timestamp, open, close, high, and low prices along with the volume for each minute.

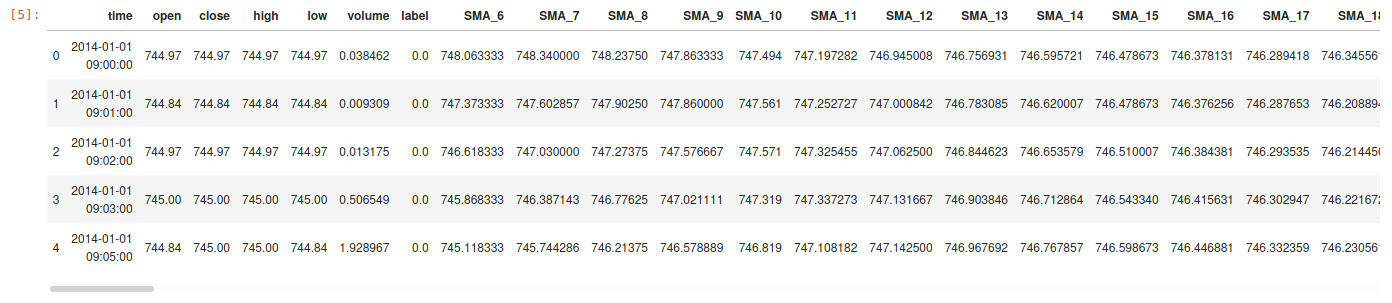
From there, each observation in the dataset was labeled as either 0, 1, 2 (“Hold”, “Buy”, “Sell”). To label the data a sliding window approach was used, since the paper used daily data they used an 11 day window, since I was using minutes data I used a 21 minute window. The labeling process was as follows: look at 21 minutes of data and see if the midpoint of the window is the low during that period, if yes label as “Buy”, if the midpoint is the high during the period then label as “Sell”, otherwise label as “Hold”. Then slide the window down by 1 minute and repeat the process until all data points are labeled.

After the data was labeled technical indicators were calculated, the paper used 15 different indicators and since I tired to code my own indicators two of those used in the paper I could figure out how to calculate. In the paper these 15 indicators were calculated over 15 different time periods and would be used to create 15 x 15 images for each minute, since I had 13 indicators I used 13 different time periods to create 13 x 13 images.

List of indicators used over a 6, 7, 8, … , 18 minutes:

|  |  |  |
| --- | --- | --- |
| Simple Moving Average | Relative Strength Index (RSI) | Stochastic Oscillator |
| Exponential Moving Average | Williams %R | Chaikin Money Flow |
| Weighted Moving Average | Commodity Channel Index | Rate of Change |
| Hull Moving Average | MACD |  |
| Triple Exponential Average | Rate of Change |  |

Here is an example of the dataframe with labels and technical indicators:

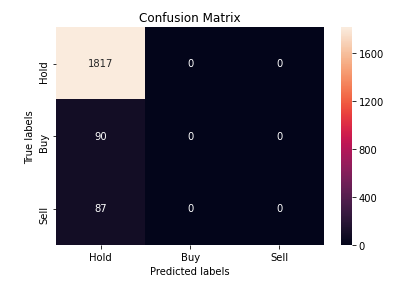


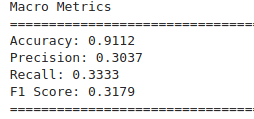
To create the images each row is transformed, once all data except for the technical indicators is dropped each minute is essentially a 1 x 169 array. The data can be reshaped into a 13 x 13 matrix where each column is a time period (6 , 7, 8, … , 18) and each row of the matrix represents a different technical indicator. Once the data is reshaped into this matrix it is transformed into a gray scale image shown below.

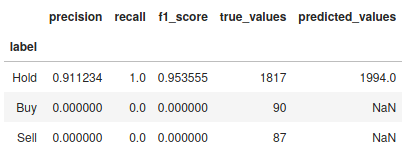


I believe I made some mistakes in calculating the indicators and maybe minutes data is much more noisy than daily data since my images look much different than the papers.

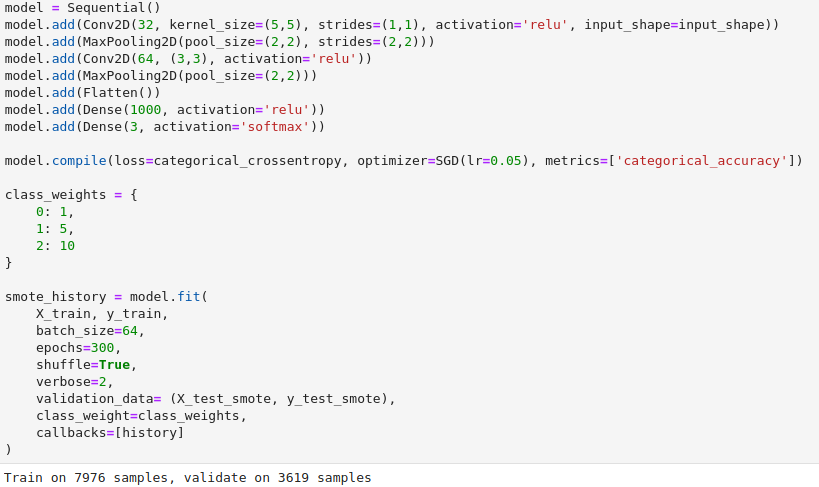
Once the images are created we can treat the problem as a classification problem since we have Buy, Sell and Hold labels. The problem is we have very imbalanced classes. My data was to large to work all of it in memory, there are over 2.6 million rows so it was broken into 10 batches to label, calculate technical indicators, and make the images. So for the CNN model I used the first 1,000 images of the first batch which has 91.2% Hold, 4.6% Sell and 4.2% Buy. To create a baseline model I did no under/over sampling and used no weights when training the CNN which lead to the model always predicting the majority class of Hold. This lead to a test accuracy of 91.2 % but is useless since it cannot identify Buy or Sell signals at all. Here is the Confusion Matrix, precision, recall and f1 score for the baseline model of just predicting Hold.



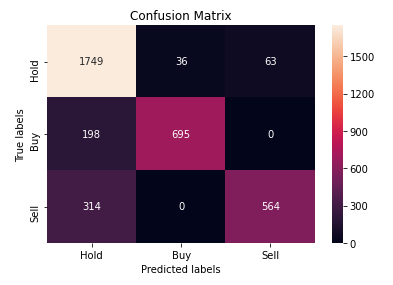


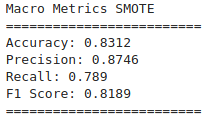


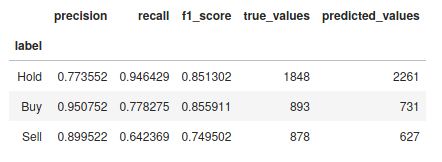
After the baseline was built I used SMOTE to oversample the minority classes to bring the distribution of Hold / Buy / Sell to 50 / 25 / 25 and I also weighted the classes as 1 / 5 / 10 so that the model would treat each Hold as 1 sample, each Buy as 5 samples, and each Sell as 10 samples. Here is the final model showing the different layers:



I didn’t have time to really tune the model and there is significant optimizations that could be done in the future. I know very little about building models like this so my felling is that it is over fitting and I need to really study how each layer is working and what the best practices are when working on a model like this. With that said the results look good, the model accuracy is about 83% and it was able to predict a significant amount of Buys and Sells correctly.







Next Steps:

The model seems to show promise and I plan on continuing my work on it. The first step is to study some more about CNN’s and to make sure I am setting up the model correctly and how to tune the parameters to find the best model. From there I will work on training the model in batches over the entire data set and see how it performs on out of sample data. Once it has been trained I will look at back testing the signals it generated over the entire data set to see how well it performed. If the results show the model has the ability to be profitable I will try to implement it and bring it into production to trade a little of my Bitcoin.