Authors: Felix Yeung , Sebastian Samuel Steiner

Neural Network for Stock Trading

Objective

Design a neural network trading model which would train on historical market data in a rolling 3-day window and predict whether the price action of the following trading horizon would reach a certain defined threshold.

In other words, the neural network will act as a binary classifier to determine whether a profit event would actually materialize after analyzing data on the 3 previous days. This model has been designed to make these predictions for SPY stocks, with the aim that our findings could also be applicable to predict price movements of other stocks.

The model was designed with the following specifications in mind:

1. Researchers can determine the probability of success via confusion matrix
2. Researchers have the flexibility to quickly alter model architecture
3. Researchers can use the model to determine whether specific technical trading patterns work (e.g. triple tops, bullish flags, etc.)

Methods

User-Defined Parameters

The model has been designed to answer "YES" or "NO" to the following type of question:
*"Given the consecutive 3 days of stock data, will the stock price **increase by 2%** from the open of day 4 to the **close** of **day 6**?"*

A similar question that may be asked to the model is:
*"Given the consecutive 3 days of stock data, will the **high** of the stock price **increase be 3%** from the open of day 4 until the close of **day 7**?"*

While the structure of the question is the same, certain parameters have changed, namely:

- Stock price increase
- Trade horizon
- Comparison between open-to-close and open-to-high
- Whether stock prices between the start and end of the trade horizon should be considered

The model allows the user to define all of these parameters, so that different trade strategies may be explored.

Input Features

Features were selected on the hypothesis that they would impact market dynamics. Primarily, the features that we have decided to explore are:

- SPY Stock Data (Source: Yahoo Finance)
    - *What is it?* Historical data about SPY stock, such as open, close, high, lows and trade volumes each day
    - *Why did we use it?* We are trying predict these relationships, so clearly these are key features for the model
- VIX (Source: Yahoo Finance)
    - *What is it?* A measure of the market's expectation for volatility; otherwise known as the "fear index". It is calculated based on the short term SPX option implied volatility.
    - *Why did we use it?* This feature was chosen because it captures the market expectation for the range that it could trade in. Furthermore, in recent years option flow has been an important driver for index prices. The explosion of market participants trading options has created structural flows in the market from Market Makers delta hedging the underlying stock.
- Day of Week/Month/Year (Source: Engineered from other sources)
    - *What is it?* Integer representation of which day of the week, month, or year.
    - *Why did we use it?* The data can help us capture potential seasonality trends (santa rally), notable days (option expiry that could have gamma/vanna/charm flows), or psychological behavior ("turnaround" Tuesdays).

Discarded Features:

- Yield Curve (Source: hometreasury.gov)
    - *What is it?* Interest rates of different maturities of US Treasuries
    - *Why didn't we use it?* The team first selected the yield curve because the hypothesis is that the features would provide context to the macro environment. However, after comparing the accuracy with & without the features, we noticed that it provided no benefits to the model. It seems like for our trading horizon the information is not useful.
- DXY (Yahoo Finance)
    - *What is it?* Index that tracks the dollars strength against a basket of currencies
    - *Why didn't we use it?* Originally, the team added Dollar to help provide more context for Macro behavior. However, this feature seems similar to the yield curve and macro environment doesn't add too much value to daily trades.

Future features to explore:

- Other Option Information (open interest, Dealer Gamma, etc.)
    - *What is it?* Beyond VIX, SPX options provide useful information about how market participants are positioned. Furthermore, Dealer Gamma could also provide context for whether the market has tendencies to overshoot or revert to the mean.
    - *Why would we use it?* As briefly mentioned before, all these option flows have started to have outsized impact on the markets. Take Covid drawdown for example, during that period most market participants are trying to hedge by buying SPX puts resulting in a very short dealer gamma environment so that any subsequent selling resulted in a vicious cycle of dealers becoming even more short to hedge their books. The selling didn't really stop until quarterly options expired. There could be many other factors that also affect day to day price action; however, from our research, options flow seem to be pretty promising to use as additional features.

Data Transformation

Now that we have our user-defined YES/NO question and input features for each day, we setup each datapoint as follows:

- Input: Joined 3 consecutive days of data (including all features for each day), hence there are 45 input features per data point
- Output: A label of 1 if the stock price high/close exceeds the threshold for the horizon, and 0 otherwise

Each feature of the input data point was scaled using a MinMaxScaler. This ensured that every data point lied in the range of 0 to 1, which allows for input weights to be comparable in further analysis.

Neural Network

The neural network has 3 fully-connected layers: an input layer with 45 nodes, a hidden layer with 45 nodes and an output layer with 1 node.

It may be argued that having a fully-connected input layer and hidden layer with 45 nodes each, which is the same as the number inputs, might lead to overfitting. However, we have taken a “hands-off” approach using the hypothesis that the neural network will figure out how to avoid overfitting by assigning small edge weights to less useful inputs and large weights to important features.

The following parameters were also tuned during validation:

- The hidden layer used a ReLu activation function
- The output layer used a sigmoid activation function (whose output is rounded to 0 or 1)
- A binary cross entropy function was used to measure the loss
- A learning rate of 0.00002 was used
- 100 epochs were used in training
- Batch sizes of 10 were used

Train, Validate and Test Splits

Data was collected from 2/1/93 to 11/23/22 giving us 7511 data points. We decided to split the data into train, validate and test, where the model was trained using training data, the models parameters were tuned using validating data and the model performance was measured using test data.

The data was split as follows:

- Training data 7027 data points (2/1/93 - 12/31/20)
- Validating data 247 data points (1/1/21 - 12/31/21)
- Testing data 221 data points (1/1/22 - 11/23/22)

Validation Loss/Accuracy & Confusion Matrix

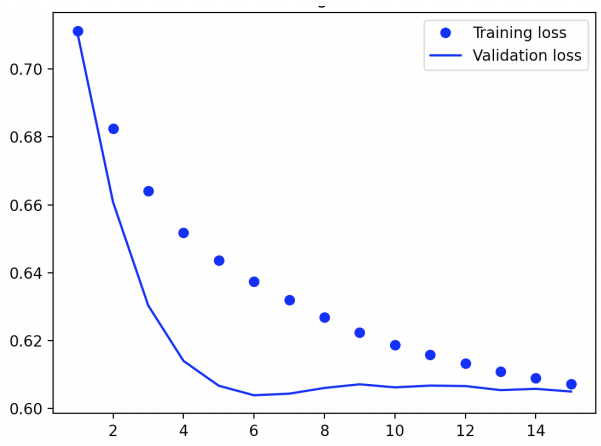
An important aspect of quantifying performance and effectiveness of model training/architecture is to track validation loss and accuracy.

Validation loss is used to measure errors made on the validation set.
Accuracy measures how accurate the model prediction is.

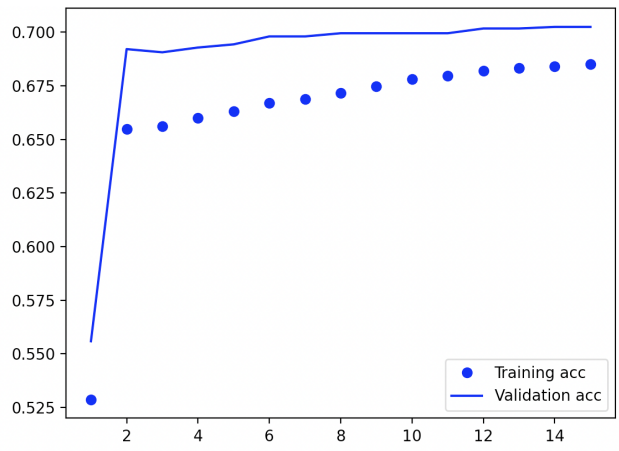
These 2 metrics are important because it helps the team understand whether the model is under/over fitting and the actual performance on predictions. These will help the team understand how to further tune the model.

An example of how we used validation split is when we decided to first train/test the model using Stratefied K fold instead of splitting by time. The initial results (screenshot below) showed that the validation loss is lower than the training loss and the validation accuracy is significantly higher than training. This is very counterintuitive because typically the model should perform worse on unseen data. We realized that using the Stratified K Fold, a test sample could ask about trading on day x and while the model has not seen this sample before, it has probably seen sample x+1, x+2, etc… which means it could be **interpolating rather than extrapolating**. Therefore, we split the data by date.

Exhibit 1. – Validation Loss is lower than training loss throughout training indicating the neural net train/test is not properly separated.



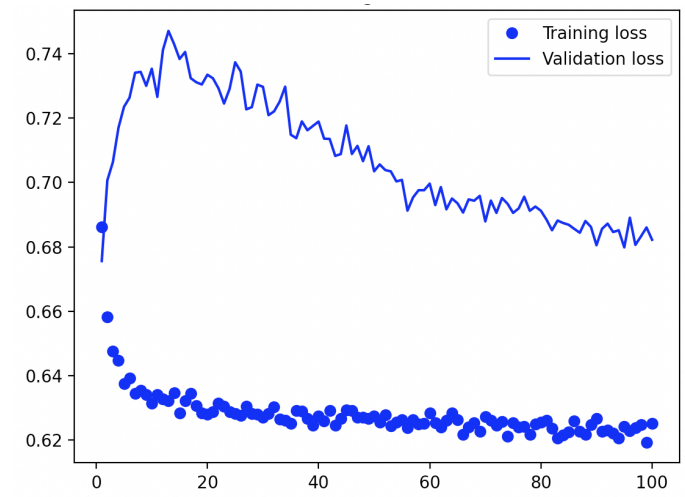
Y = loss / X = epochs



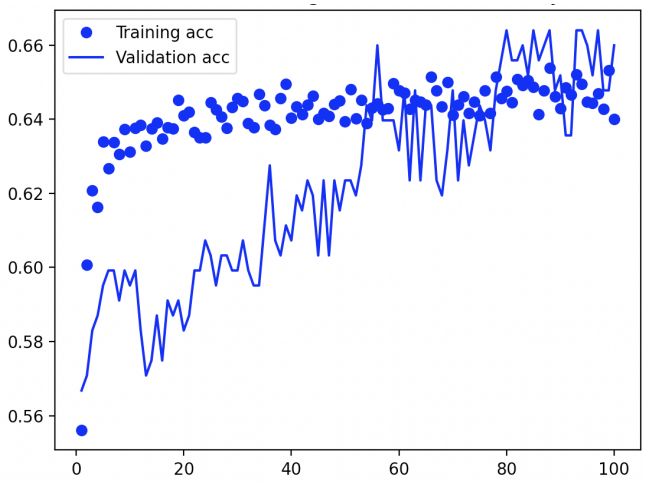
Y = Accuracy / X = epochs

Indeed, the preferred behavior researchers would want to see is the validation loss start high and converge over time to training loss while accuracy is improving.

Exhibit 2. – After splitting the data to hide "future" data from training, the NN training trajectory is much more similar to what's expected. Validation Loss is decreasing over time to converge while accuracy is increasing.



Y = loss / X = epochs



Y = Accuracy / X = epochs

Another method the team used to measure accuracy is via the **confusion matrix**. This gives us good summary statistics of how well the model classifies tradeable events. In addition to accuracy, the team is also focused on the true positive rate. This will ensure that the algorithm actually has a tradeable edge going for a particular direction. The worst is if it is only slightly better at each individual classification.

Results

We tested the model under 84 different user-defined parameters by changing the length of horizon, open-close/open-high and the stock increase.

- Trading horizon = [1, 2, 3]
- Percent change = [0.5, 0.75, 1, 1.5, 2, 2.5, 3]
- Days Between = [0,1] //toggles between focusing on last day of trading horizon or select best day
- Close High Comparison = ['High', 'Close'] //toggles between comparing delta of high vs close of horizon

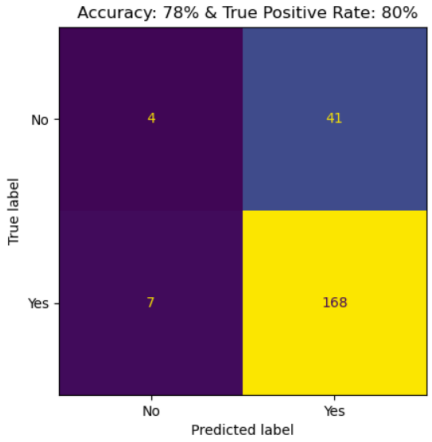
We found that the classification accuracy ranged between 52%-94% and the true positive rate ranged between 0%-80%. This clearly demonstrates that the model performs better under certain conditions.

Here is the link to all the charted results.

The accuracy and true positive rate were summed together in order to assign a score for each test. Below is a table showing the models top 5 performances. From this table, we see that the model performs best when predicting the high over a 2 day horizon. Furthermore, the best results are seen when predicting if the stock will increase by at least 0.5%. In other words, the model performs best when the following question is asked: *"Given the consecutive 3 days of stock data, will the high of the stock price increase by 0.5% from the open of day 4 to the close of day 5?"*

Horizon (Days)	High/Close	Stock Increase (%)	Accuracy (%)	True Positive Rate (%)
2	High	0.5	75	76
2	High	0.75	67	69
2	High	0.5	78	80
2	High	0.75	71	73
2	High	1	65	68

For the best performing model (in terms of accuracy+TPR score), we have included the confusion matrix below.



Conclusion and Further Work

In summary the team is happy with the results. We found that while the model's performance varies under different test parameters, its best performance is seen when predicting if the high will exceed 0.5% over a 2-day horizon.

For further work, the team can focus on the following avenues:

- Translating to trading strategy
    - A suitable strategy needs to capitalize on our edge which can identify up 0.5% up days 80% of the time while preventing drawdowns on the one that are missed.
    - Perhaps we can impose stop losses to ensure on the days that we trade incorrectly, our losses do not become too large.
- Ensembling different models
    - Currently, we developed a model that purely identifies positive % movements in SPY. This same model can be used to identify negative % movements. In the future, the team can explore combining both models by ensembling the results so that the strategy makes money and avoids large drawdowns.
    - By using outputs of different models at the same time, we can diversify strategies and diversify risk.
- Model Optimization
    - The team can look into including new input features.

- Also, the team should look to include pruning of the network because it is likely that the model has edges with small weights that are propagating noise through the network, which could be avoided if pruning were deployed

Extra Notes

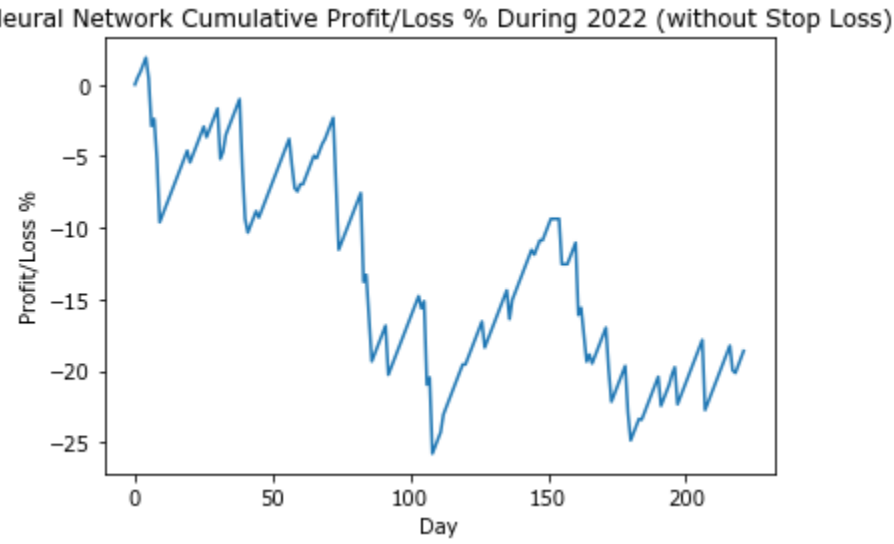
Does the neural network discover patterns or triggers?

With further analysis of the confusion matrix, it can be seen that the model should have traded 175 times this year and shouldn't have traded 45 times this year. Given this large contrast between days the model should and should not trade, a conclusion may be drawn that the model has not discovered a "trigger" to decide whether to trade or not. Rather it has a statistical pattern that buying at the open and taking a profit at a 0.5% price increase over a 2-day horizon is a profitable trade strategy.

Even though this model has identified a statistical pattern rather than a trade trigger, we believe other models that we tested identified trade triggers. For example, when looking at the high at 0.75% over a 2-day horizon the real trade and no trade days were split 116 and 106, respectively. The results of the model were 56% for both accuracy and true positive rate. While these statistics are lower than the best model, in this environment the trade/no trade split was so close that we believe the network identified a data-dependent trigger that determines whether to trade or not. Therefore, while the model has proven that it can be used to identify statistical patterns to develop trade strategies, we believe there is potential that it can be refined to also make decisions on specific trades.

Does the confusion matrix tell the whole story?

In the graph below, we see the cumulative profit-loss during 2022 is -18.6% (note this is simple P/L not compound). Therefore, despite the high accuracy and true positive rate, the model actually loses money. The reason for this is that there is no stop loss to prevent the model from losing large sums of money. Given that the maximum profit made in a trade is 0.5%, the days where the model loses 5% are detrimental to its performance. Hence, as mentioned in the further work, introducing a stop loss to eliminate the days of extreme loss is an important next step, along with an investigation into ensembling the decoder to predict down days and making better informed decisions



To illustrate how adding a stop loss could improve performance. We introduced a condition that if the model incorrectly predicts the trade, mark the loss as the maximum between a stop loss of -2% and the close of day 2. As can be seen in the figure below, the model's loss is now reduced to -3.6%. It is important to highlight that an assumption was made when collecting these results that a low of more than -2% was not realized before the high of the trade was reached for cases where the model correctly predicted a trade. Therefore, the results of this stop loss adjustment should be viewed as a demonstration of the potential this approach has rather than proof that this adjustment works.