MDS Capstone Project – Seahorse Strategies

Final Report

Simon Chiu, Gilbert Lei, Fan Wu, LinYang Yu

Executive Summary

In this project, we developed a supervised learning model that helps Seahorse Strategies to screen buy and sell signals generated by its proprietary Oscillator system and identify good signals on which the company may choose to trade. Our final model is based on a LightGBM framework and some other additional features we engineered. With this model, Seahorse Strategies can have a much more automated workflow, do more winning trades and possibly make more profits.

Introduction

Seahorse Strategies has a proprietary algorithm (the Oscillator) that tracks stock price movements, identifies overbought and oversold situations and gives buy/sell signals when certain conditions are met. Currently, human judgement is required to decide on whether a buy/sell signal is indeed a good trade. Seahorse Strategies is interested in knowing whether the quality of a signal could be quantified. For example, is it possible to quantify the winning probability, or the expected return, of each signal?

To support this project, Seahorse Strategies sent us more than 200,000 signals generated by the Oscillator on 30 stocks based on historical stock prices in the past 10 years. Each signal comes with the prior three-day stock price information (closing prices for every 30 minutes) and its corresponding Oscillator values and MACD values (MACD represents moving average convergence/divergence. It is one of the most popular technical indicators among stock traders worldwide), as well as the return value, which is calculated based on the assumption that we follow the buy signal and sell when the next sell signal appears. When the return values are over zero, we consider the signals as winning and profitable. Otherwise, the signals will be considered as a loss. Our goal would be differentiating the winnings signals (labeled as 1) out of the losing signals (labeled as 0).

Scientific Objective

To achieve this goal, we first defined our project as a classification problem. The objective for this stage would be building a supervised machine learning model as our classifier based on the given signals. The response variable that the model was designed to predict is the sign (positive or negative) of the return for each individual signal. The evaluation method for this model was based on the overall accuracy of prediction. Our partner later pointed out that the false positives are more harmful than false negatives in our case because due to the large volume of trading, we can simply focus on a small portion of true positive signals that are profitable. Thus, the evaluation method for our classification model is not limited to the accuracy but the confusion matrix, or, more precisely, the precision. Our model also outputs a winning probability instead of just a simple win or loss label. However, that model raised another critical issue that a higher winning rate does not always lead to a higher profitability, which is the ultimate goal of any stock investment. To acquire more information such as the values of the

returns into our model, we redefined our goal as a regression rather than a classification. In this case, we evaluate ~~and score~~ our model on its ability to capture the patterns ~~and differentiating~~ the winning signals ~~out of~~ the losing signals ~~more efficiently with the signals provided~~.

Data Science Methods - Feature Engineering

We started this project by trying all the models on the original features (a list of models attempted will be provided in the Algorithm session later) but none of the model gave us a reasonable result. We then realized the need of finding a better way to present our data to the model, in the hope that it will allow any model to have a better understanding of the connection between the features and the label. We started working on extracting additional features such as parameters of a time series (e.g. smoothness) from the original features shortly after.

The pro of having additional features is that it actually provides significant modeling power to support our final product. In fact, all of the additional features ~~that got selected~~ have higher feature importance~~/usefulness~~ than the original features in both number of split and gain of information ~~(two different ways of feature usefulness measurements with our final choice of algorithm)~~.

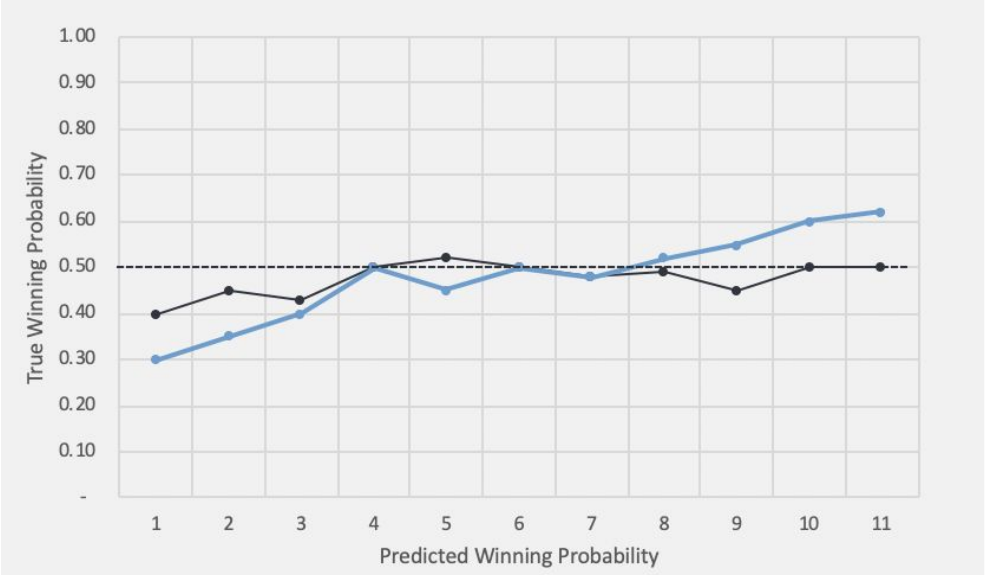

Figure1: Model Before Adding Additional Features (Black) vs. After (Blue)

One potential ~~concern that data scientist might have~~ is that useless additional features might affect the performance of the ~~machine learning~~ model. This issue is avoided since the algorithm of our choice can ~~pick~~ the useful features ~~out of all the features~~. However, we still tried to address this issue by ~~not calculating the~~ additional features that ~~is shown to have~~ a ~~lower~~ feature importance than the original features, which also cuts the training time and prediction time.

The ideas of our additional features comes from both our partner and our own understanding of the stock market. Here is a list of selected additional features that shown to be useful and their corresponding reasonings:

1. ~~Oscillator Curve's smoothness~~ - partner's suggestion
2. Stock Volatility - common measure of the risk level of a stock
3. (Relative) Derivatives - measurement of the movement(s) of a curve
4. Ratios - Ratios of the movement of one curve over the other. It is also an indicator that our partner usually looks into (1 means two curves are moving in the same direction, -1 means two curves are moving in the opposite directions)

We are still open to ideas of new features that could make the model better. Possible features includes but not limited to: trade volumes, option expiration date (Yes/No),

Data Science Methods - Algorithm

Our final choice of algorithm for this project is the LightGBM regressor. It is a gradient boosted decision tree algorithm like the XGBoost or AdaBoost that was developed by Microsoft.

The LightGBM algorithm is known for its speed over its competitors. It utilizes a decision tree growing method called the leaf-wise tree growth which it only grows the decision tree by node (check figure below) when it is necessary in order to ~~achieve such an outstanding~~ performance. It also supports both categorical feature and numerical features so we can easily add features such as the stock name and the type of signal (buy/sell) without having to do a one hot encoding.

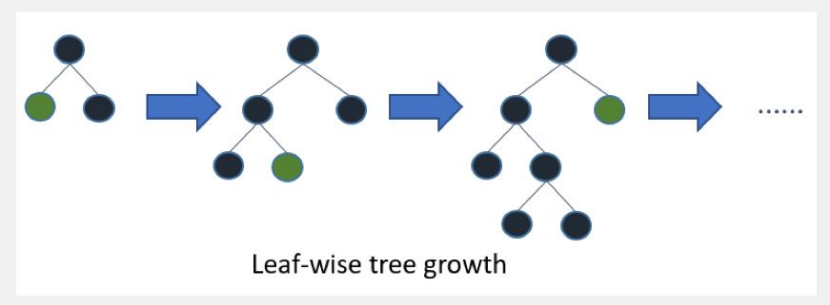

Figure 2: Leaf-wise Tree Growth in a Decision Tree

One drawback of using the LightGBM algorithm is that it does not work well with small datasets. It is generally recommended to use it with dataset that has over 10,000 records. Also, there are a lot of possible hyperparameters to be tuned ~~if we would like to have the model super fine-tune for each training~~. Fortunately, there are a few hyperparameters that are way more critical than the others that we have ~~already tuned in our final product~~. Although we believe it

might have a low time efficiency, one might have to fine tune the model case by case depending on the training dataset in order to achieve optimal model performance.

There are a list of other algorithm we have tried and they either run slower or have worse results than the LightGBM algorithm.

Random Forest | k-Nearest Neighbor | Logistic Regression

Convolutional Neural Network | Recurrent Neural Network | Bernoulli Naïve Bayes

Data Product and Results

Our final product is a Python pipeline in a Github repo that would automatically generate scores for trade signals. A detailed tutorial will be provided with a screencast of how the data product is intended to be used. The following is a brief overview:

1. Users should first clone the repo to their local computer
2. Place the training data in the Train_data folder and the prediction data in the Prediction folder.
3. Run the Train.py script to generate training report.
4. Run the Predict.py script to generate scores, with a score of 0 being the worst and a score of 100 being the best. Refer to the score/return chart in the training report for investment performance estimates (example below).

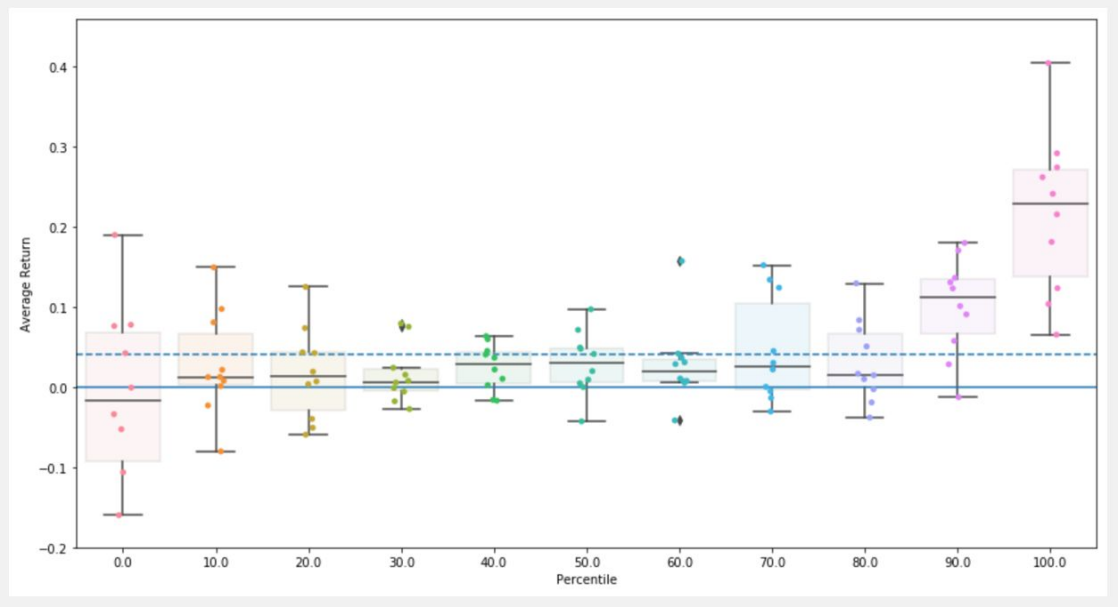

Figure 3: Box Plot in the Training Report Shows Performance of Signals by Their Scores

This procedure allows the input files and the output files to be in the text file format which is our partner's preferred data format. It is also quick to be implemented as all of our code were already written in a python script. The drawbacks of this interface is that it requires our users to have knowledge in using the terminal in their computer to call the scripts. It also takes time to generate the text files and interpret the results. We believe that the best solution in this case is to have our system integrated into our client's Oscillator system. However, given the limited time of this capstone project and the confidentiality of the Oscillator system, we believe that having this separated system is a great choice that gives both good user useability and speedy program implementation.

Conclusions and recommendations

In conclusion, we developed a supervised learning model that can be used to score signals generated by Seahorse Strategies's Oscillator system. Our model suggests Seahorse Strategies to focus on signals with high scores and discard signals with low scores. With this model, Seahorse Strategies can significantly reduce their reliance on human judgment in evaluating signals and as such they could scale up their stock portfolio, do more trades and generate more profits.

Beyond this project, we recommend Seahorse Strategies to take additional efforts to make this model more robust and effective. Firstly, Seahorse Strategies should feed in new data and retrain the model regularly (monthly or at least quarterly) so that the model can adapt to the change in the stock market. Secondly, we suggest Seahorse Strategies to explore new features, such as trading volumes, that might be able to further improve the current model. Lastly, Seahorse Strategies may try ensemble method to handle outliers (extreme return cases) more effectively.