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. The final model is based on a LightGBM framework with additional data features for this specific project. With this model, Seahorse Strategies may identify quality signals more efficiently and achieve more winning trades.

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 provided 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 Moving Average Convergence/Divergence (MACD) values, as well as the return value, which is the percentage change of the stock price based on the assumption that we follow the signal and buy/sell when the next signal appears. When a return value is above zero, the corresponding signal is considered as winning and profitable. Otherwise, the signal will be considered as a loss. Our goal would be differentiating the winning signals (labeled as 1) out of the losings (labeled as 0).

Scientific Objective

To achieve this goal, the project was first defined as a classification problem. The objective for this stage was to build a classifier based on the given information. 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. However, in the stock market, false positives are more harmful than false negatives because we are only investing in the signals predicted to have positive returns. Also, due to the large volume of tradable signals generated, even extracting only a small portion of true positive signals is considered to be valuable. Thus, the loss function for the classification

model focused not limited to the accuracy but also the precision. Meanwhile, to further accomplish the goal, the model also output a predicted winning probability besides a win or loss label. However, another critical issue is that a higher winning rate does not always lead to a higher profitability, which is the ultimate goal of any stock investment. To incorporate more information such as the percentages of returns into the model, the goal was re-defined as a problem of regression rather than classification. The final model was evaluated on its ability to capture the patterns of the winning and losing signals effectively.

Data Science Methods - Feature Engineering

The project was started 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 provided a meaningful result. Due to the indirect connections between the original features and the labels, the models were not able to learn valuable information from the data. To allow a model to work better, additional features, including but not limited to smoothness, derivatives and frequencies, were engineered based on the original features.

The key advantage of having additional features is that if the new features have a direct connection with the labels, then they would allow the model to learn more from the training data and perform better in prediction. In fact, all of the additional features selected have higher feature importance than the original features in both number of split and gain of information.

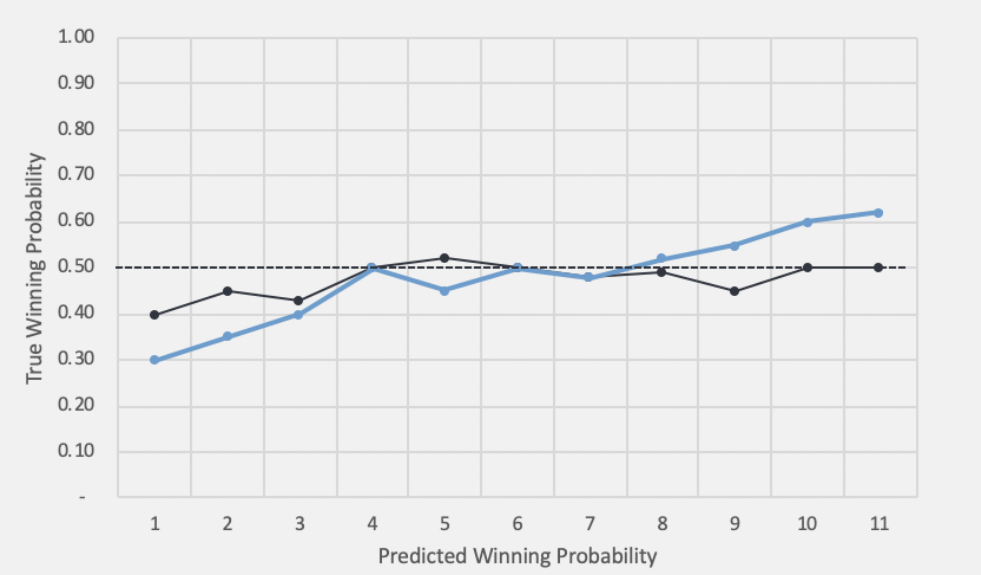

Figure1: Model Before Adding Additional Features (Black) vs. After (Blue)

One potential point of concern is that useless new features might affect the performance of the model. This issue is avoided in this case since the algorithm of choice can automatically select the useful features. However, we still tried to address this issue by removing any additional features that has a feature importance lower than the original features. This also speeds up the feature calculation process.

The ideas of these new features come from both Seahorse Strategies and our team's own understanding of the stock market. Here is a list of selected additional features that shown to be useful and their corresponding reasonings:

1. The Smoothness of the Oscillator curve, which is assessed by Seahorse Strategies as part of their decision-making process
2. Stock Volatility - common measure of the risk level of a stock
3. (Relative) Derivatives - measurement of the velocity of the movement(s) of a curve
4. Ratios - Ratios of the movement of one curve over the other. It is an indicator that Seahorse Strategies usually looks into. A ratio of 1 means two curves are moving in the same direction and -1 means two curves are moving in the opposite directions.

Besides the features above, other meaningful features such as trade volumes or option expiration date (Yes/No) are up for future investigation. These could possibly further improve the performance of the model.

Data Science Methods - Algorithm

The final choice of algorithm for this project is the LightGBM regressor. It is a gradient boosted decision tree algorithm that was developed by Microsoft.

The LightGBM algorithm is known for its speed over its alternatives. 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 improve its speed performance. It also supports both categorical feature(s) and numerical feature(s) 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.

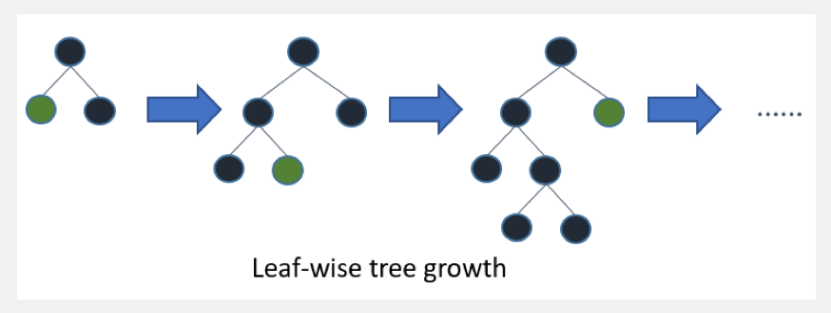

Figure 2: Leaf-wise Tree Growth in a Decision Tree

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 a dataset that has over 10,000 records. The other potential drawback is that there are a lot of possible hyperparameters. To achieve optimal model performance, one might have to tune many of these hyperparameters, and it is usually a time-consuming process. Fortunately, for this case, a few hyperparameters are way more critical than the others. As such, we investigated and tuned a handful of critical hyperparameters.

Besides the LightGBM framework, our team also investigated other algorithms as well. Below is a list of other algorithms we have tried. We did not use them in the end because the LightGBM algorithm outperformed all of them.

Random Forest	k-Nearest Neighbor	Logistic Regression
Convolutional Neural Network	Recurrent Neural Network	Bernoulli Naïve Bayes

Data Product and Results

The final product is a Python pipeline in a GitHub repo that could automatically generate scores for trade signals. A detailed tutorial will be provided with screenshots 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 data for prediction in the Test_data folder.
3. Run the Train.py script to generate a 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).

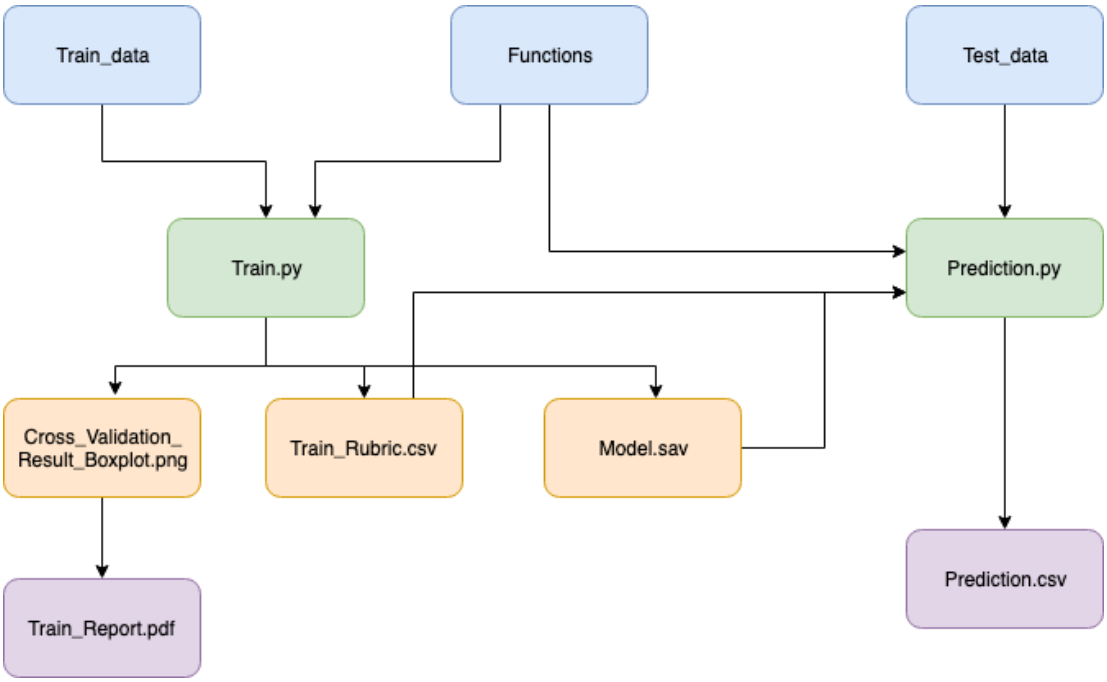

Figure 3: Workflow of the Final Product

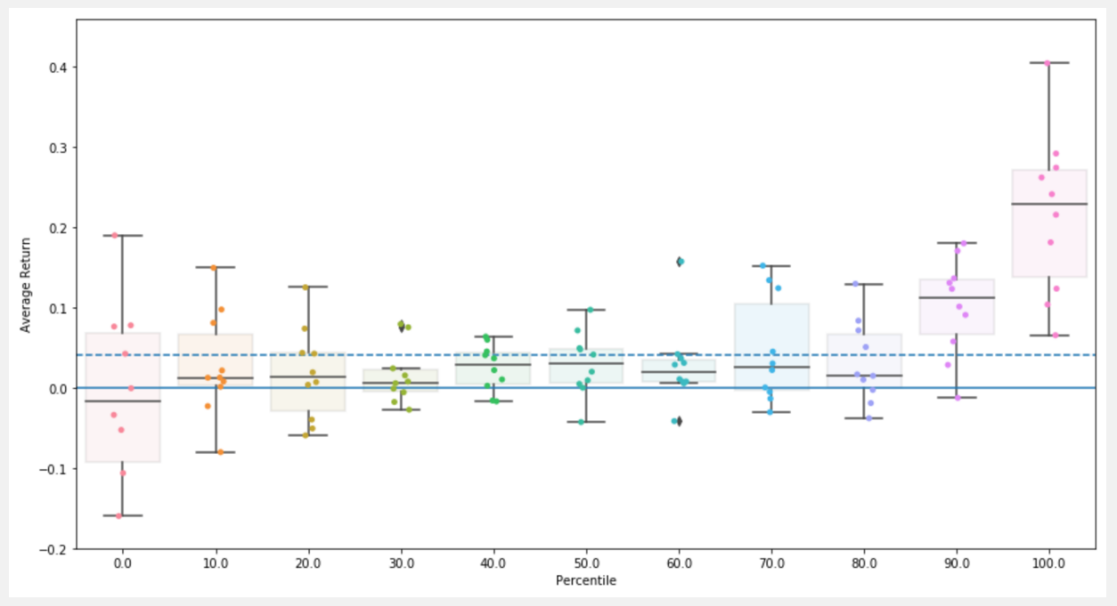

Figure 4: Box Plot in the Training Report Shows Performance of Signals by Their Scores

This procedure allows the input files and the output files to be in the text file format which is Seahorse Strategies' preferred data format. It is also quick to be implemented as all of the code were already written in a python script. The drawbacks of this interface are that it requires 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 the system integrated into Seahorse Strategies 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 good balance between user usability and program implementation.

Conclusions and recommendations

In conclusion, we developed a supervised learning model that can be used to score signals generated by Seahorse Strategies' Oscillator system. The model should point 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 possibly scale up their stock portfolio, do more trades and generate more profits.

As for potential future improvements of the current model, we recommend Seahorse Strategies to consider the following points:

1. 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.
2. We suggest Seahorse Strategies to explore new features, such as trading volumes, that might be able to further improve the current model.
3. Seahorse Strategies may try an ensemble model structure to handle outliers (extreme return cases) more effectively.