FinRl Contest Task 1

Nikolaus Holzer*
Elena Zhao*
nh2677@columbia.edu
elena.zhao@columbia.edu
Columbia University
New York, New York, USA

ABSTRACT

Description of Finrl Contest Task 1 submission. This approach utilizes custom indicators to increase the performance of the model over the baseline. Moreover this approach optimized a custom loss function to improve model performance. Overall this approach increases model performance.

ACM Reference Format:
Nikolaus Holzer and Elena Zhao. 2023. FinRl Contest Task 1. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Stock price prediction is a data heavy task, that requires good data to perform well. Logical steps to improve the performance of the model thus include data cleaning and preprocessing to highlight important correlations and trends in the data for the model to train on. This solution applies significant data cleaning and preprocessing to introduce new indicators in addition to the already included ones. Another important step to optimize model performance was modifying the loss function. We found that the base model had a tendency to incur high losses in times with high stock market turbulence. Thus a modified loss function was proposed and optimized to improve model performance.

2 DATA PREPROCESSING

This submission uses feature engineering to add Moving averages and Price Rate of Change indicators to the data set. The most significant initial change was observed by changing the relationship between the sell and buy cost. Below is a list of indicators that we decided to add:

- **MACD (Moving Average Convergence Divergence)**:
- **RSI_x (x-period Relative Strength Index)**:
- **boll_ub (Bollinger Bands Upper Band)**:
- **boll_lb (Bollinger Bands Lower Band)**:
- **OBV (On-Balance Volume)**:
- **MA_y (y-period Moving Average)**:
- **MACD_Signal**:

Adding these indicators improved performance of the model over various training and testing periods, yet still produced high losses and max drawdowns.

3 CUSTOM LOSS FUNCTION

The next modification of this submission is the custom loss function. The approach encompassed a custom child class of StockTradingEnv which redefined the step function to modify the return values depending on hyper parameters which were tuned as well. This approach enabled for finer determination of model behaviour given different market environment, and during testing provided at least equal if not better results over various timeframes. This solution found that penalizing turbulent losses at a certain threshold by a factor of three provided good returns and low drawdowns.

```
class CustomStockTradingEnv(StockTradingEnv):
def __init__(self, turb_pen, current_turb,\
    reward_mult, toggle, *args, **kwargs):

    super().__init__(*args, **kwargs)
    self.turb_pen = turb_pen
    self.current_turb = current_turb
    self.reward_mult = reward_mult
    self.toggle = toggle

def step(self, actions):
 state, reward, done, t1, t2 = super().step(actions)

    if self.toggle:
        return state, reward, done, t1, t2

    turbulence_penalizer = self.turb_pen
    current_turbulence =
        self.data[self.risk_indicator_col].iloc[0]
    # increase negative rewards
    if (
        current_turbulence > self.current_turb
    ):
        if reward < 0:
            reward *= turbulence_penalizer
            reward = -(reward**self.reward_mult)

    # return state, reward, done, info
    return state, reward, done, t1, t2
```

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

4 PARAMETER TUNING

The final step of optimizing a solution included parameter tuning. For this task we modified the parameters of the custom loss function, sell and buy cost and the sizes of the windows of the data preprocessing task. In doing so this solution was able to optimize model performance. An intuitive step that was undertaken was raising the buying and selling cost which paired with the exponential loss increase for turbulent losses made the model perform better and increase returns. By increasing the cost of selling and buying, this encourages the model to take less risks and sell less thereby leverage market trends that it is confident in. We observed that this change significantly increased final winnings, but introduced higher variance into the max lows and highs. To combat this higher variance the custom loss parameter was introduced to encourage selling when turbulence is high.

This is the training and testing result of the standard model with data preprocessing, custom loss function, and tuned parameters. Trained 2010-2013, tested 2013-2022.

```
    Preprocessing,
    Trading Cost Configuration,
    Step Cost Configuration:
{
    "ma_windows": [
        3,
        7
    ],
    "roc_windows": [
        3,
        7
    ],
    "stoch_window": 5,
    "ema_span": 5,
    "rsi_window": 10,
    "buy_cost_pct": 0.0014,
    "sell_cost_pct": 0.0014,
    "turb_pen": 1.0,
    "current_turb": 10,
    "reward_mult": 3,
    "toggle": false
}
Backtest Results:
Annual return          0.240810
Cumulative returns     5.965883
Annual volatility      0.197587
Sharpe ratio           1.192011
Calmar ratio           0.796453
Stability              0.959661
Max drawdown          -0.302353
Omega ratio            1.257764
Sortino ratio          1.722761
Skew                        NaN
Kurtosis                    NaN
Tail ratio             1.058947
Daily value at risk   -0.023959
```

Testing supports the hypothesis that risk averse models with high loss penalty perform well.

5 CONCLUSION

The submission for task 1 utilizes an array of methods from data preprocessing to loss function tuning to improve the performance of the model on the testing set. The steps taken serve to improve model generalization, it's ability to understand the data and it's aversion to high drawdown moves.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009