ISY5002 PRS

NATIONAL UNIVERSITY OF SINGAPORE

ISS-IS

FINANCE TOOL ON STOCKS MARKET SYSTEM

Author:

YangWenkai

HuangRunxiang

ChenHaoyang

Course Lead:

Zhu Fangming

October 26, 2022

Contents

List of Figures

List of Tables

1 Introduction

Finance serves a purpose. ... Investors are lured to gamble their wealth on wide hunches originated by charlatans and encouraged by mass media. One day in the near future, ML will dominate finance, science will curtail guessing, and investing will not mean gambling. Our mission is to set up a system to efficiently automate trading.

- Help Users make decisions.
- Make the Profit maximum.
- Generate Users Financial Report.

We use multiple Deep Reinforcement Learning algorithm to train the policy and Q-value. The users just load our pre-trained models and the system will return the accumulative profit and everyday actions.We believe applying these intellectual and engineering system to finance will initiate a paradigm shift from the conventional trading routine to an automated machine learning approach, even RLOps in finance.
Deep Reinforcement Learning is a type of supervised machine learning. However, DRL doesn't need large labeled training datasets. This is a significant advantage since the amount of data grows exponentially today, it becomes very time-and-labor-consuming to label a large dataset. DRL uses a reward function to optimize future rewards, in contrast to an ML regression/classification model that predicts the probability of future outcomes.

1.1 Aims and Objectives

Our system use Dow 30 stocks and NDAQ 100 stocks from a real time database API(Yaha Finance). Our aim is training a profitable DRL trading agent that can help users decide where to trade, at what price, and what quantity. At the same time, the system will review the history action, stocks environment and daily return to generate your finance report. Besides, our DRL agents incorporates essential trading constraints such as transaction cost, market liquidity, and investor's degree of risk-aversion, which technical indicators could be a symbol of stocks market status. Our system simulated trading environments across various markets including Dow-30 and NASDAQ-100 and provides fine-tuned state-of-the-art DRL algorithms including DQN, DDPG, PPO, SAC, A2C, TD3.

1.2 Current Methods

Nowadays, most quantitative trading methods are based on simple financial features or classic statistical models. For instance, if there is a increasing and stable stock, most methods will suggest to buy this stock. Or using some time series forecasting to predict the price of stock. In general, these methods is only for single stock and could not provide the long-term suggestion. At the same time, these decision strategy could not update according to the market change. Therefore, theses methods could not fit the environment quickly and make suitable actions to get higher profit.

1.3 The Advantage of DRL

1) Deep reinforcement learning algorithms can outperform human players in many challenging games. For example, on March 2016, DeepMind's AlphaGo program, a deep reinforcement learning algorithm, beat the world champion Lee Sedol at the game of Go.

2) Return maximization as trading goal: by defining the reward function as the change of the portfolio value, Deep Reinforcement Learning maximizes the portfolio value over time.

3) The stock market provides sequential feedback. DRL can sequentially increase the model performance during the training process.

4) The exploration-exploitation technique balances trying out different new things and taking advantage of what's figured out. This is difference from other learning algorithms. Also, there is no requirement for a skilled human to provide training examples or labeled samples. Furthermore, during the exploration process, the agent is encouraged to explore the uncharted by human experts.

5) Experience replay: is able to overcome the correlated samples issue, since learning from a batch of consecutive samples may experience high variances, hence is inefficient. Experience replay efficiently addresses this issue by randomly sampling mini-batches of transitions from a pre-saved replay memory.

6) Multi-dimensional data: by using a continuous action space, DRL can handle large dimensional data.

7) Computational power: Q-learning is a very important RL algorithm, however, it fails to handle large space. DRL, empowered by neural networks as efficient function approximator, is powerful to handle extremely large state space and action space.

2 Business Case

2.1 Problem Statement

As mentioned above, the current quantitative trade just rely on the simple features analysis. Hence, our team intend to a finance tool based on the deep reinforcement learning to assist people make decision. In stocks market, there are many types of stocks. If we just use traditional quantitative trade method, we could not get distribution of stocks. So we try the intelligent agent algorithms to decide buy what stock and how much stock according to the environment.

With the development of machine learning, the reinforcement learning has been regarded as a advanced method to solve the problems about interaction with environment for better result. Once the deep reinforcement learning is a good solution to game or control. However why we can see the stock market as a finance game to get more profit and have a better strategy. Moreover, profitable automated stock trading strategy is vital to investment companies and hedge funds. It is applied to optimize capital allocation and maximize investment performance, such as expected return. Return maximization can be based on the estimates of potential return and risk. However, it is challenging to design a profitable strategy in a complex and dynamic stock market. Every player wants a winning strategy. Needless to say, a profitable strategy in such a complex and dynamic stock market is not easy to design. Yet, we are to reveal a deep reinforcement learning scheme that automatically learns a stock trading strategy by maximizing investment return.

2.2 Proposed Solution

A ensemble deep reinforcement learning trading strategy is our system aim, which includes three actor-critic based algorithms: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). The reason why we applied different algorithms is that it combines the best features of the three algorithms, thereby robustly adjusting to different market conditions.

Our system provides common building blocks that allow strategy builders to configure stock market datasets as virtual environments, to train deep neural networks as trading agents, to analyze trading performance via extensive backtesting, and to incorporate important market frictions. On the lowest level is environment, which simulates the financial market environment using actual historical data from six major indices with various environment attributes such as closing price, shares, trading volume, technical indicators etc. In the middle is the agent layer that provides fine-tuned standard DRL algorithms (DDPG, PPO, SAC and A2C), commonly used reward functions and standard evaluation baselines to alleviate the debugging workloads and promote the reproducibility. The agent interacts with the environment through properly defined reward functions on the state space and action space. The top layer includes applications in automated stock trading, where we demonstrate three three cases, namely everyday return, action of buy or sell and portfolio allocation.

2.3 Market Analysis

Preceding DRL, conventional reinforcement learning (RL) has been applied to complex financial problems, including option pricing, portfolio optimization and risk management. In the past time, someone utilized policy search and direct RL for stock trading. But it could not react to the stock market change and not use in multiple stocks trade. Recently, many research applying deep neural networks profits more and be more universal. There are industry practitioners who have explored trading strategies fueled by DRL, since deep neural networks are significantly powerful at approximating the expected return at a state with a certain action. With the development of more robust models and strategies, general machine learning approaches and DRL methods in specific are becoming more reliable. For example, DRL has been implemented on sentimental analysis on portfolio allocation and liquidation strategy analysis, showing the potential of DRL on various financial tasks. Hence, we develop the ensemble deep reinforcement learning trading strategy system for the financial users.

3 Product Details

3.1 Member Roles

Name	ID	Role
Yang Wenkai	A0261636A	Backend Application Deveopment, Algorithm/Model Development and Training
Huang Runxiang	A0261627A	Backend Application Deveopment, UI/UX Development
Chen Haoyang	A0261905E	Backend Application Deveopment, UI/UX Development

Table 1: Role Table

3.2 Main Features

To better help users make financial decisions and market conditions, our system has developed two basic modes, and select the most classic stocks which could represent the current stock market situation.

- Mode 1. Dow 30 stocks

  The system provide the Dow Jones 30 stocks, the user could visit the details of every stocks via the link on Yahoo finance. The users just input your trade start date, end data and initial asset, the system will automatically obtain the information of every stocks as the trade environment. The system will generate your finance report and portfolio allocation.

- Mode 2. NASDAQ 100 stocks

  Comparing with Mode 1, the system provide more stocks for users, which could be a symbol of American stocks market. At the same time, our system could supply more information about portfolio allocation which is also influenced by your selected period and initial asset.

3.3 System Architecture

The chart below shows the general Architecture of our system. There are mainly two parts of our system

- The interactive webpage - Our system is reflected on one web page for the user to access, which have three pages home, Dow-30 and NASDAQ-100.
- Backend System - The web page is supported by the backend system with API interface corresponding to the functions, pretrained models and data preprocess modules.

3.4 Related Works

Recent applications of deep reinforcement learning in financial markets consider discrete or continuous state and action spaces, and employ one of these learning approaches: critic-only approach, actor-only approach, or and actor-critic approach.

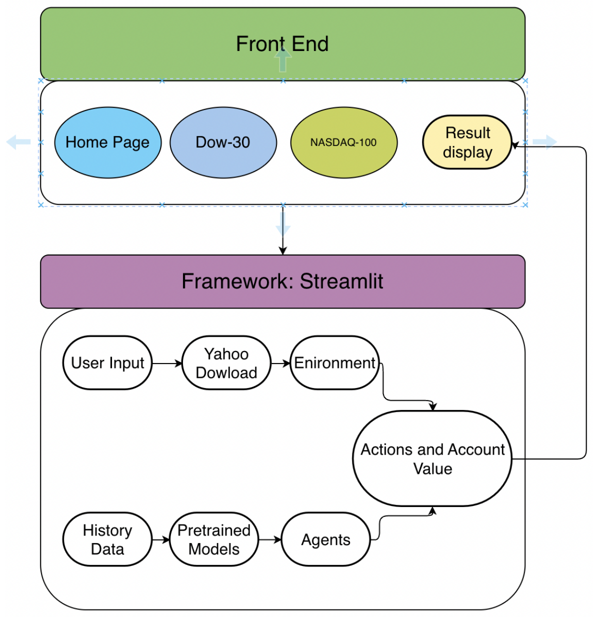


Figure 1: Overall Structure

Critic-only approach: the critic-only learning approach, which is the most common, solves a discrete action space problem using, for example, Q-learning, Deep Q-learning (DQN) and its improvements, and trains an agent on a single stock or asset. The idea of the critic-only approach is to use a Q-value function to learn the optimal action-selection policy that maximizes the expected future reward given the current state. Instead of calculating a state-action value table, DQN minimizes the mean squared error between the target Q-values, and uses a neural network to perform function approximation. The major limitation of the critic-only approach is that it only works with discrete and finite state and action spaces, which is not practical for a large portfolio of stocks, since the prices are of course continuous.

- **Q-learning**: is a value-based Reinforcement Learning algorithm that is used to find the optimal action-selection policy using a Q function.
- **DQN**: In deep Q-learning, we use a neural network to approximate the Q-value function. The state is given as the input and the Q-value of allowed actions is the predicted output.

Actor-only approach: The idea here is that the agent directly learns the optimal policy itself. Instead of having a neural network to learn the Q-value, the neural network learns the policy. The policy is a probability distribution that is essentially a strategy for a given state, namely the likelihood to take an allowed action. The actor-only approach can handle the continuous action space environments.

- **Policy Gradient**: aims to maximize the expected total rewards by directly learns the optimal policy itself.

Actor-Critic approach: The actor-critic approach has been recently applied in finance. The idea is to simultaneously update the actor network that represents the policy, and the critic network that represents the value function. The critic estimates the value function, while the actor updates the policy probability distribution guided by the critic with policy gradients. Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach has proven to be able to learn and adapt to large and complex environments, and has been used to play popular video games, such as Doom. Thus, the actor-critic approach fits well in trading with a large stock portfolio.

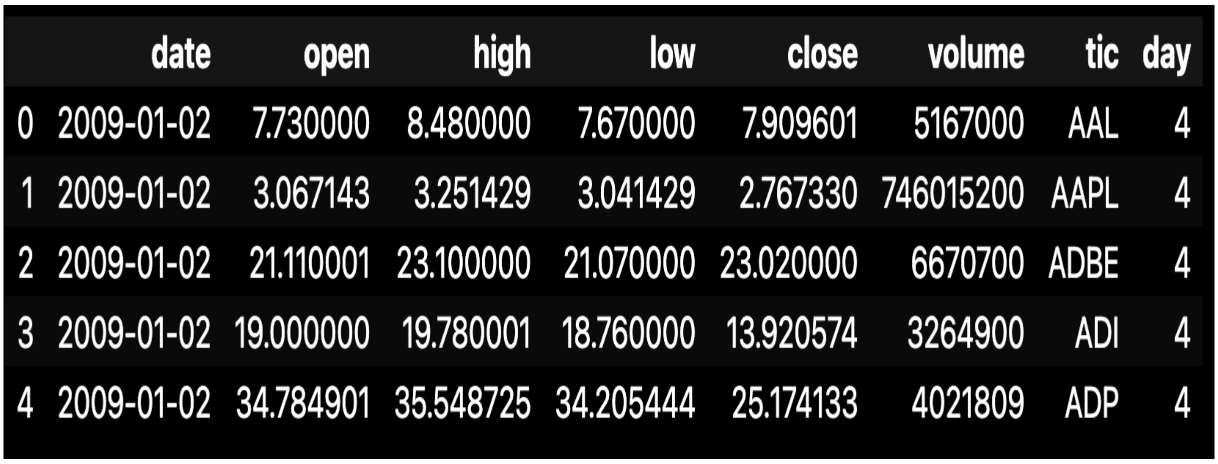
- **A2C**: A2C is a typical actor-critic algorithm. A2C uses copies of the same agent working in parallel to update gradients with different data samples. Each agent works independently to interact with the same environment.
- **PPO**: PPO is introduced to control the policy gradient update and ensure that the new policy will not be too different from the previous one.
- **DDPG**: DDPG combines the frameworks of both Q-learning and policy gradient, and uses neural networks as function approximators.

4 Modeling Details

4.1 Training for the best performance agent

4.1.1 Data Preparation

For this function, we intend to utilize the Yahoo Finance API to obtain all stocks history data such Dow-30 and NASDAQ-100. Assume we are at time t, at the end of day at time t, we will know the open-high-low-close price of the Dow 30 and NASDAQ-100 stocks. We will set these raw data as the training environment to verify whether our actions have the higher profit.
Firstly, we will split our all data into two parts training data and trading data. The training data will last from 2009 to 2020 and the trade data will be 2020 to 2022. In that case, we could check whether our models have the best performance in trade part. Although in real life trading, the models needs to be updated periodically using rolling windows. Our system provide the trade rolling windows, users could select what they want period to check whether their assets have improved.



	date	open	high	low	close	volume	tic	day
0	2009-01-02	7.730000	8.480000	7.670000	7.909601	5167000	AAL	4
1	2009-01-02	3.067143	3.251429	3.041429	2.767330	746015200	AAPL	4
2	2009-01-02	21.110001	23.100000	21.070000	23.020000	6670700	ADBE	4
3	2009-01-02	19.000000	19.780001	18.760000	13.920574	3264900	ADI	4
4	2009-01-02	34.784901	35.548725	34.205444	25.174133	4021809	ADP	4

Figure 2: Raw Data Details

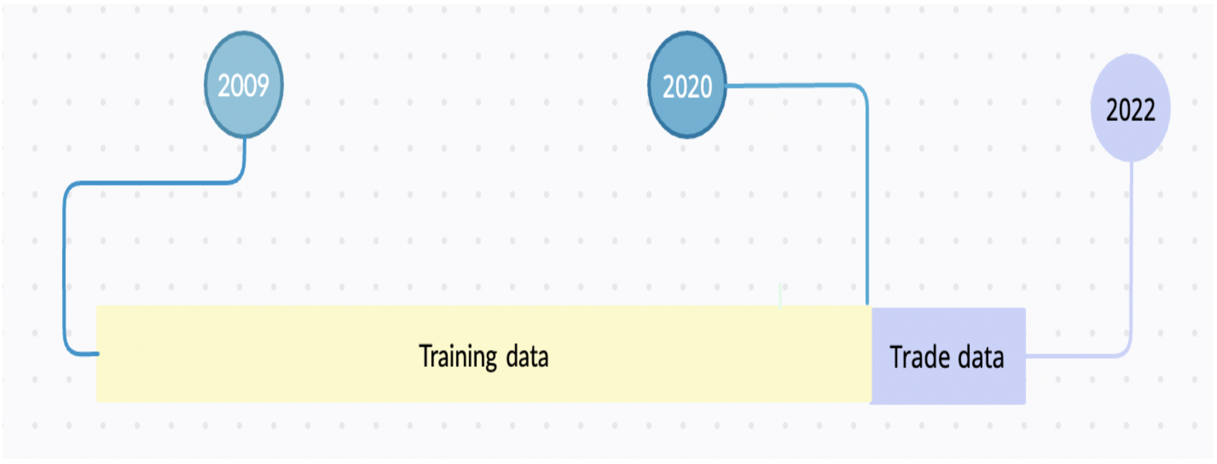
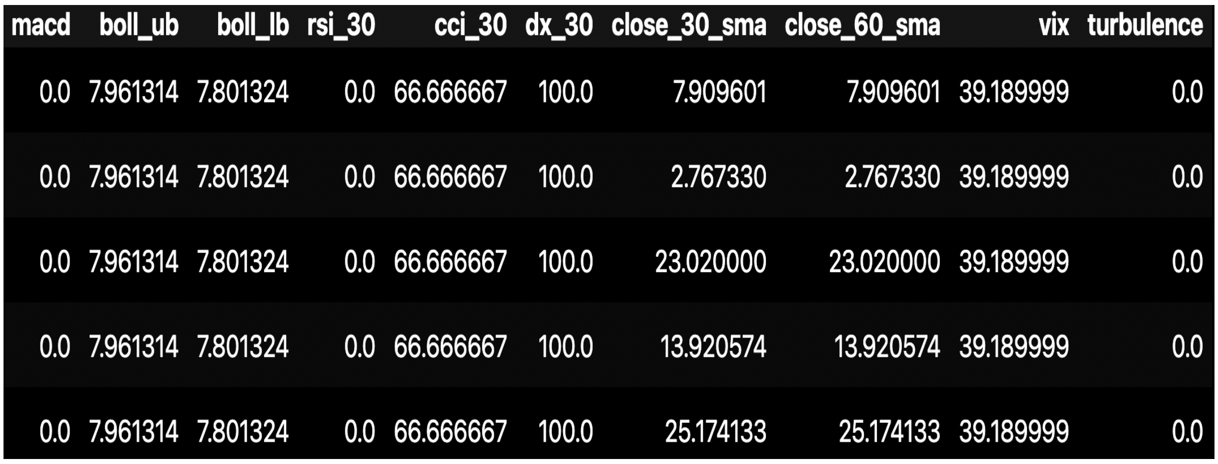


Figure 3: Raw Data Details

4.1.2 Data Preprocessing

It is vital to choose a good way to preprocess the raw data before training them into the models, we create the feature engineer to calculate technical indicators such as MACD, RSI, CCI, ADX according to the basic stocks price. In Reinforcement Learning, we call these data or features as “states”. For multiple stock trading,



macd	boll_ub	boll_lb	rsi_30	cci_30	dx_30	close_30_sma	close_60_sma	vix	turbulence
0.0	7.961314	7.801324	0.0	66.666667	100.0	7.909601	7.909601	39.189999	0.0
0.0	7.961314	7.801324	0.0	66.666667	100.0	2.767330	2.767330	39.189999	0.0
0.0	7.961314	7.801324	0.0	66.666667	100.0	23.020000	23.020000	39.189999	0.0
0.0	7.961314	7.801324	0.0	66.666667	100.0	13.920574	13.920574	39.189999	0.0
0.0	7.961314	7.801324	0.0	66.666667	100.0	25.174133	25.174133	39.189999	0.0

Figure 4: Preprocessed data

as the number of stocks increase, the dimension of the data will increase, the state and action space will grow exponentially. So stability and reproducibility are very essential.

4.1.3 Set the Training Environment

To start with, the environment must be a time-driven trading simulator. Considering the stochastic and interactive nature of the automated stock trading tasks, a financial task is modeled as a Markov Decision Process (MDP) problem. The training process involves observing stock price change, taking an action and reward’s calculation to have the agent adjusting its strategy accordingly. By interacting with the environment, the trading agent will derive a trading strategy with the maximized rewards as time proceeds. Our trading environments, based on OpenAI Gym framework, simulate live stock markets with real market data according to the principle of time-driven simulation.

State Space: We use a 181-dimensional vector (30/100 stocks * 6 + 1) consists of seven parts of information to represent the state space of multiple stocks trading environment

1) **Balance**: available amount of money left in the account at current time step

2) **Price**: current adjusted close price of each stock.

3) **Shares**: shares owned of each stock.

4) **MACD**: Moving Average Convergence Divergence (MACD) is calculated using close price.

5) **RSI**: Relative Strength Index (RSI) is calculated using close price.

6) **CCI**: Commodity Channel Index (CCI) is calculated using high, low and close price.

7) **ADX**: Average Directional Index (ADX) is calculated using high, low and close price.

4.1.4 Algorithm and Model Training

Our target is to design an automated trading solution for multiple stock trading. We model the stock trading process as a Markov Decision Process (MDP). We then formulate our trading goal as a maximization problem. The components of the reinforcement learning environment are:

- **Action**: The action space describes the allowed actions that the agent interacts with the environment. Normally, a A includes three actions: a 1, 0, 1, where 1, 0, 1 represent selling, holding, and buying one stock. Also, an action can be carried upon multiple shares. We use an action space k, ..., 1, 0, 1, ..., k, where k denotes the number of shares. For 30 stocks the entire action space is $(2k+1)^3$, in this system we use khmax=100, so the entire action space is around 10. It means we can sample a maximum of 10 pairs of state and action combinations.For example, ”Buy 10 shares of AAPL” or ”Sell 10 shares of AAPL” are 10 or 10, respectively

- **State**: The state space describes the observations that the agent receives from the environment. Just as a human trader needs to analyze various information before executing a trade, so our trading agent observes many different features to better learn in an interactive environment.balance, close price, shares, MACD, RSI, CCI, ADX, 181-dimensional vector (30/100 stocks * 6 + 1)

- **Reward**: r(s, a, s) is the incentive mechanism for an agent to learn a better action. The change of the portfolio value when action a is taken at state s and arriving at new state s’, i.e., r(s, a, s) = v v, where v and v represent the portfolio values at state s and s, respectively

- **Policy** $\pi(s)$: the trading strategy at state s, which is the probability distribution of actions at state s.

- **Q-value** $Q\pi(s,a)$: the expected reward of taking action a at state s following policy π.

- **Environment**: multiple stock trading for Dow 30 and NASDASQ 100 constituents.

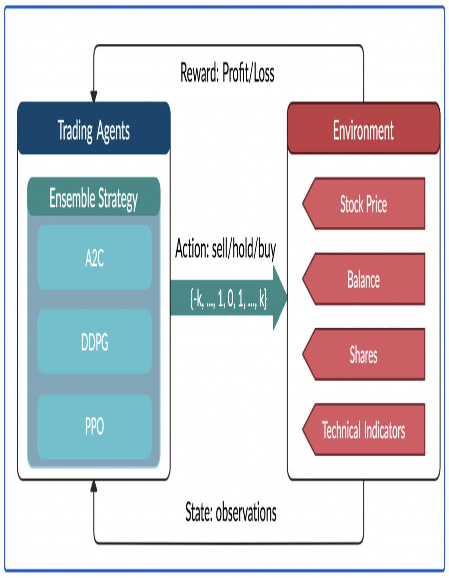


Figure 5: Training Overview

A2C is a typical actor-critic algorithm which we use as a component in the ensemble method. A2C is introduced to improve the policy gradient updates. A2C utilizes an advantage function to reduce the variance of the policy gradient. Instead of only estimates the value function, the critic network estimates the advantage function. Thus, the evaluation of an action not only depends on how good the action is, but also considers how much better it can be. So that it reduces the high variance of the policy networks and makes the model more robust.

A2C uses copies of the same agent working in parallel to update gradients with different data samples. Each agent works independently to interact with the same environment. After all of the parallel agents finish calculating their gradients, A2C uses a coordinator to pass the average gradients over all the agents to a global network. So that the global network can update the actor and the critic network. The presence of a global network increases the diversity of training data. The synchronized gradient update is more cost-effective, faster and works better with large batch sizes. A2C is a great model for stock trading because of its stability.
We utilize a layered structure, as shown . DRL models consists of three layers: data layer, environment layer, and agent layer. Each layer executes its functions and is relatively independent. Meanwhile, layers interact through end-to-end interfaces to implement the complete workflow of algorithm trading.

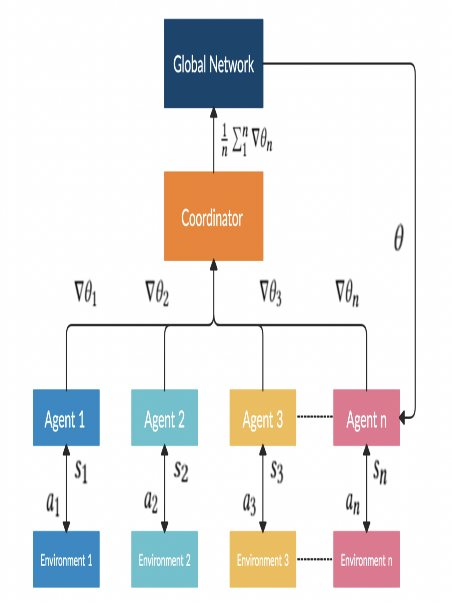


Figure 6: A2C Structure

DDPG is an actor-critic based algorithm which we use as a component in the ensemble strategy to maximize the investment return. DDPG combines the frameworks of both Q-learning and policy gradient, and uses neural networks as function approximators. In contrast with DQN that learns indirectly through Q-values tables and suffers the curse of dimensionality problem, DDPG learns directly from the observations through policy gradient. It is proposed to deterministically map states to actions to better fit the continuous action space environment.

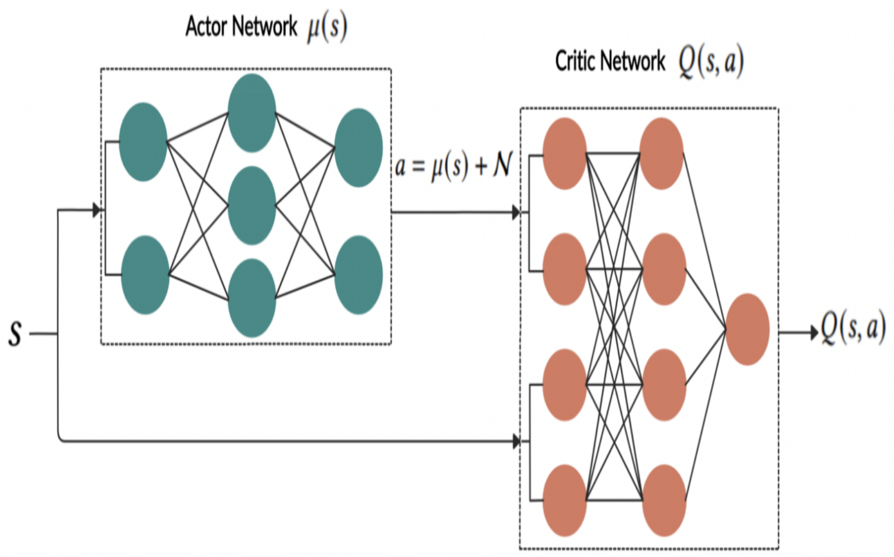


Figure 7: DDPG Structure

We explore and use **PPO** as a component in the ensemble method. PPO is introduced to control the policy gradient update and ensure that the new policy will not be too different from the older one. PPO tries to simplify the objective of Trust Region Policy Optimization (TRPO) by introducing a clipping term to the objective function.

The objective function of **PPO** takes the minimum of the clipped and normal objective. PPO discourages large policy change move outside of the clipped interval. Therefore, PPO improves the stability of the policy networks training by restricting the policy update at each training step. We select PPO for stock trading because it is stable, fast, and simpler to implement and tune.

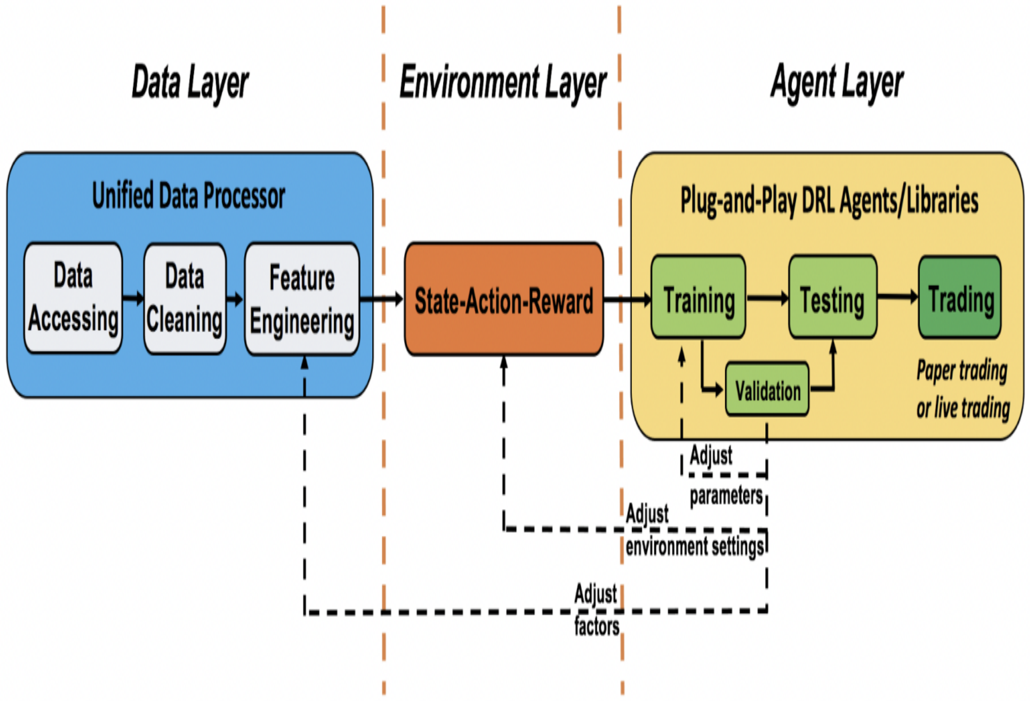


Figure 8: Model Overview

4.1.5 Modeling Conclusion and Performance

After training for all algorithms we could choose the best performance models. The details is access in Table 2.

Our purpose is to create a highly robust trading strategy. So we use an ensemble method to automatically select the best performing agent among all algorithms to trade based on the Sharpe ratio. The ensemble process is described as follows:

Algorithm	**Input**	**Output**	**Type**	**State-action spaces support**	**Finance use cases support**	**Features and Improvements**	**Advantages**
DQN	States	Q-value	Value based	Discrete only	Single stock trading	Target network, experience replay	Simple and easy to use
Double DQN	States	Q-value	Value based	Discrete only	Single stock trading	Use two identical neural network models to learn	Reduce overestimations
Dueling DQN	States	Q-value	Value based	Discrete only	Single stock trading	Add a specialized dueling Q head	Better differentiate actions, improves the learning
DDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Being deep Q-learning for continuous action spaces	Better at handling high-dimensional continuous action spaces
A2C	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Advantage function parallel gradients updating	Stable, cost-effective, faster and works better with large batch sizes
PPO	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Clipped surrogate objective function	Improve stability, less variance, simple to implement
SAC	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Entropy regularization, exploration-exploration trade-off	Improve stability
TD3	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Clipped double Q-learning, delayed policy updated, target policy smoothing	Improve DDPG performance
MADDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Handle multi-agent RL problem	Improve stability and performance

Table 2: Algorithms Performance

- We use a growing window of n months to retrain our three agents concurrently. In this system, we retrain our six agents at every three months.
- We validate all three agents by using a 3-month validation rolling window followed by training to pick the best performing agent which has the highest Sharpe ratio. We also adjust risk-aversion by using turbulence index in our validation stage.
- After validation, we only use the best model with the highest Sharpe ratio to predict and trade for the next quarter.

4.2 Trade on the best agent

4.2.1 How to trade on the agent

The environment for training and trading is different in multiple stock trading case.For trading: turbulence index is used as a risk aversion signal after the actions generated by the DRL algorithms. Turbulence index should not be included in training, because it is not a part of model training, so only a trading environment should include the risk aversion signal.

Suppose that we have a well trained DRL agent “DRL Trader”, we want to use it to trade multiple stocks in our portfolio.

- As mention above, we get the states from the technical indicators such as MACD, RSI, CCI, ADX.
- We know that our portfolio value V(t) = balance (t) + dollar amount of the stocks (t).
- We calculate k = actions *hmax, hmax is a predefined parameter that sets as the maximum amount of shares to trade. So we will have a list of shares to trade.
- The dollar amount of shares = shares to trade* close price (t).

- Update balance and shares. These dollar amount of shares are the money we need to trade at time t. The updated balance = balance (t) amount of money we pay to buy shares +amount of money we receive to sell shares. The updated shares = shares held (t) shares to sell +shares to buy.
- So we take actions to trade based on the advice of our DRL Trader at the end of day at time t (time t's close price equals time t+1's open price). We hope that we will benefit from these actions by the end of day at time t+1.
- Take a step to time t+1, at the end of day, we will know the close price at t+1, the dollar amount of the stocks (t+1)= sum(updated shares * close price (t+1)). The portfolio value V(t+1)=balance (t+1) + dollar amount of the stocks (t+1).
- So the step reward by taking the actions from DRL Trader at time t to t+1 is r = v(t+1) v(t). The reward can be positive or negative in the training stage. But of course, we need a positive reward in trading to say that our DRL Trader is effective.
- Repeat this process until termination.

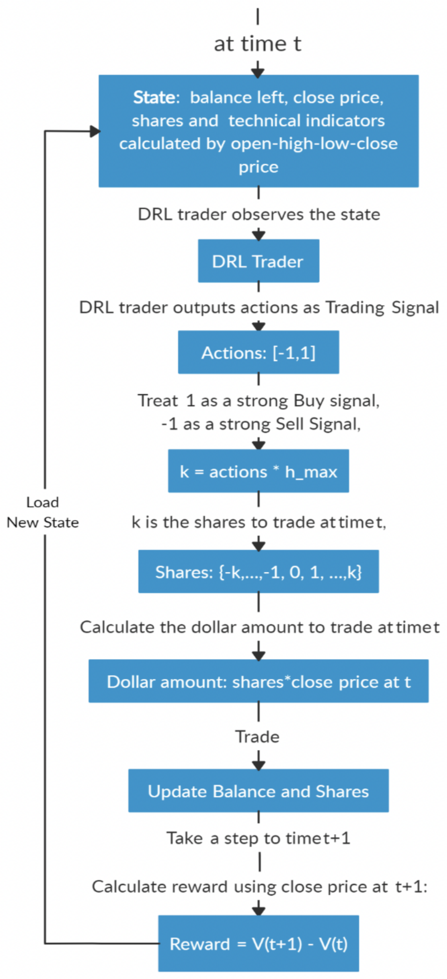
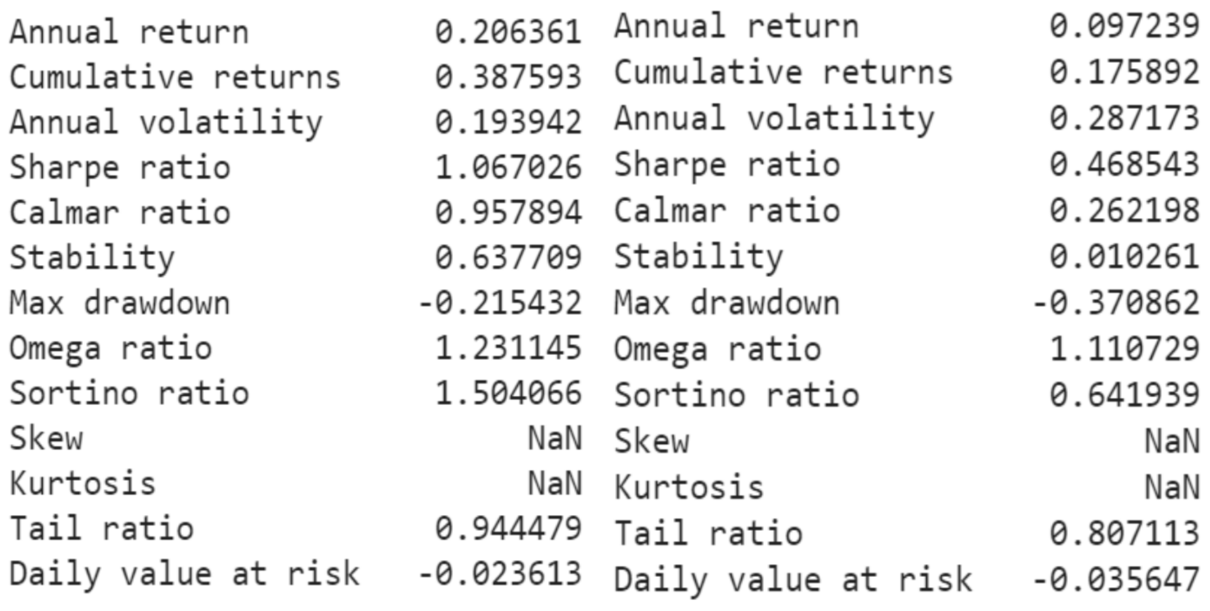

at time t
State: balance left, close price, shares and technical indicators calculated by open-high-low-close price
DRL trader observes the state
DRL Trader
DRL trader outputs actions as Trading Signal
Actions: [-1,1]
Treat 1 as a strong Buy signal, -1 as a strong Sell Signal,
k = actions * h_max
k is the shares to trade at time t,
Shares: {-k,...,-1, 0, 1, ...,k}
Calculate the dollar amount to trade at time t
Dollar amount: shares*close price at t
Trade
Update Balance and Shares
Take a step to time t+1
Calculate reward using close price at t+1:
Reward = V(t+1) - V(t)
Load New State

Figure 9: Algorithm step

4.2.2 Result Analysis

Assume that we have 1,000,000 dollor initial capital at start date. We use the SAC model to trade the Dow 30 stocks. We use pyfolio to do the backtesting. The charts look pretty good, and it takes literally one line of code to implement it. You just need to convert everything into daily returns.

The left table is the stats for backtesting performance, the right table is the stats for Index (DJIA) performance. You can also check your cumulative return and monthly return etc.



```
Annual return          0.206361   Annual return          0.097239
Cumulative returns     0.387593   Cumulative returns     0.175892
Annual volatility      0.193942   Annual volatility      0.287173
Sharpe ratio           1.067026   Sharpe ratio           0.468543
Calmar ratio           0.957894   Calmar ratio           0.262198
Stability              0.637709   Stability              0.010261
Max drawdown          -0.215432   Max drawdown          -0.370862
Omega ratio            1.231145   Omega ratio            1.110729
Sortino ratio          1.504066   Sortino ratio          0.641939
Skew                        NaN   Skew                        NaN
Kurtosis                    NaN   Kurtosis                    NaN
Tail ratio             0.944479   Tail ratio             0.807113
Daily value at risk   -0.023613   Daily value at risk   -0.035647
```

Figure 10: Performance Details

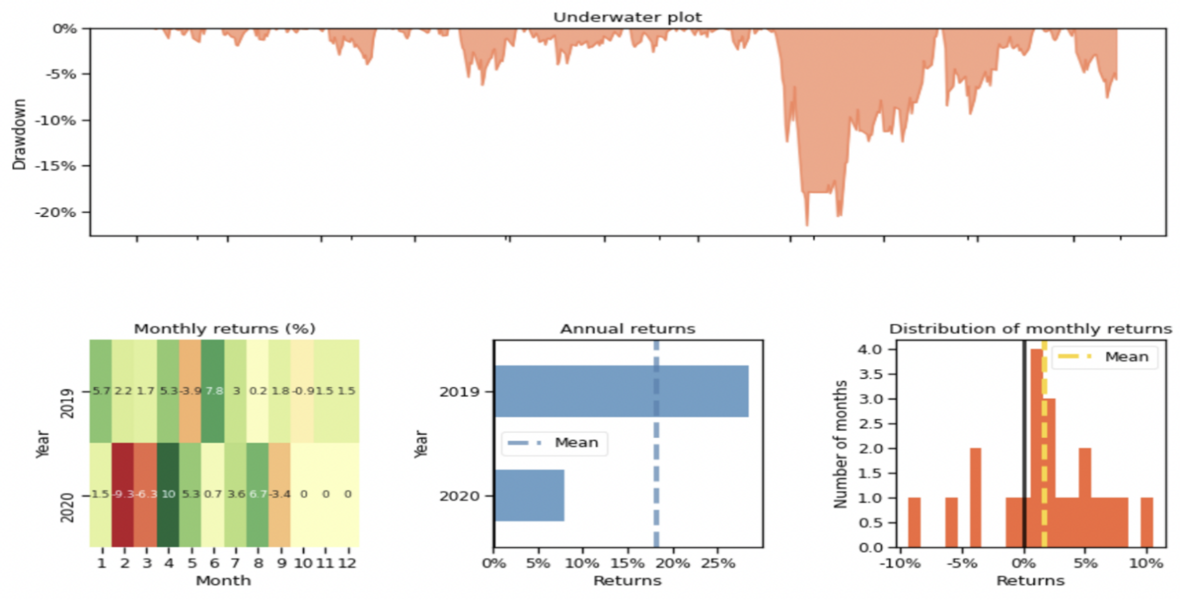


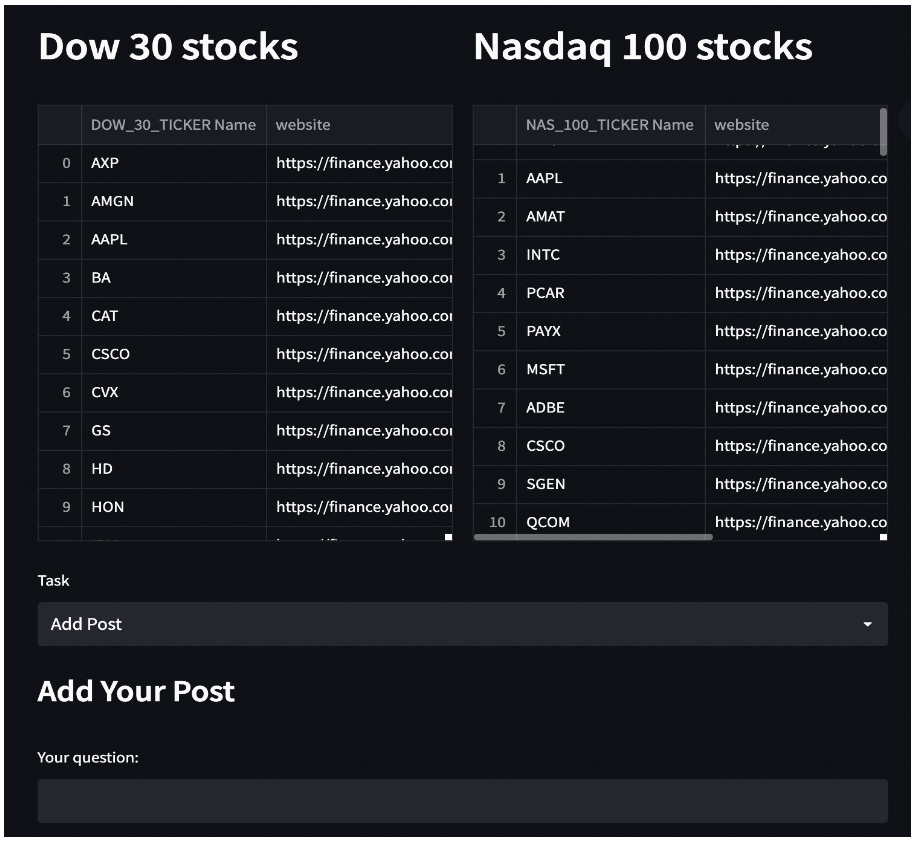
Figure 11: Return Report

5 System Development & Frontend

The system framework called streamlit, which is a easy package to set up machine learning web. At the same time, because we need to display huge dataframes and figures. It requires our frontend has too many pages and the normal html script could not fit the pandas very well. Hence, the streamlit is the faster way to build and share data apps. To local users, you just need to install the streamlit package and the algorithms required packages, and you can run our app on your laptop successfully. Also to the Internet users, the users just click our web link , dont't need to install any packages and you can access the resources.

5.1 Web page User interface Design

According to the goal and aim of our system, this system include a login page for convenient usage. Thus, the system user should register and login in the home page. After above two steps, you could directly see our Dow-30 and NASDAQ-100 information and the linf of our data source(YaHoo Finance). In that case, the users could access the details of every stock including low price, high price and close price etc. In order to improve the users experience and solve system bugs, we add the Post function. Users could feedback the information about the web or algorithms.



Dow 30 stocks

	DOW_30_TICKER Name	website
0	AXP	https://finance.yahoo.co
1	AMGN	https://finance.yahoo.co
2	AAPL	https://finance.yahoo.co
3	BA	https://finance.yahoo.co
4	CAT	https://finance.yahoo.co
5	CSCO	https://finance.yahoo.co
6	CVX	https://finance.yahoo.co
7	GS	https://finance.yahoo.co
8	HD	https://finance.yahoo.co
9	HON	https://finance.yahoo.co

Nasdaq 100 stocks

	NAS_100_TICKER Name	website
1	AAPL	https://finance.yahoo.co
2	AMAT	https://finance.yahoo.co
3	INTC	https://finance.yahoo.co
4	PCAR	https://finance.yahoo.co
5	PAYX	https://finance.yahoo.co
6	MSFT	https://finance.yahoo.co
7	ADBE	https://finance.yahoo.co
8	CSCO	https://finance.yahoo.co
9	SGEN	https://finance.yahoo.co
10	QCOM	https://finance.yahoo.co

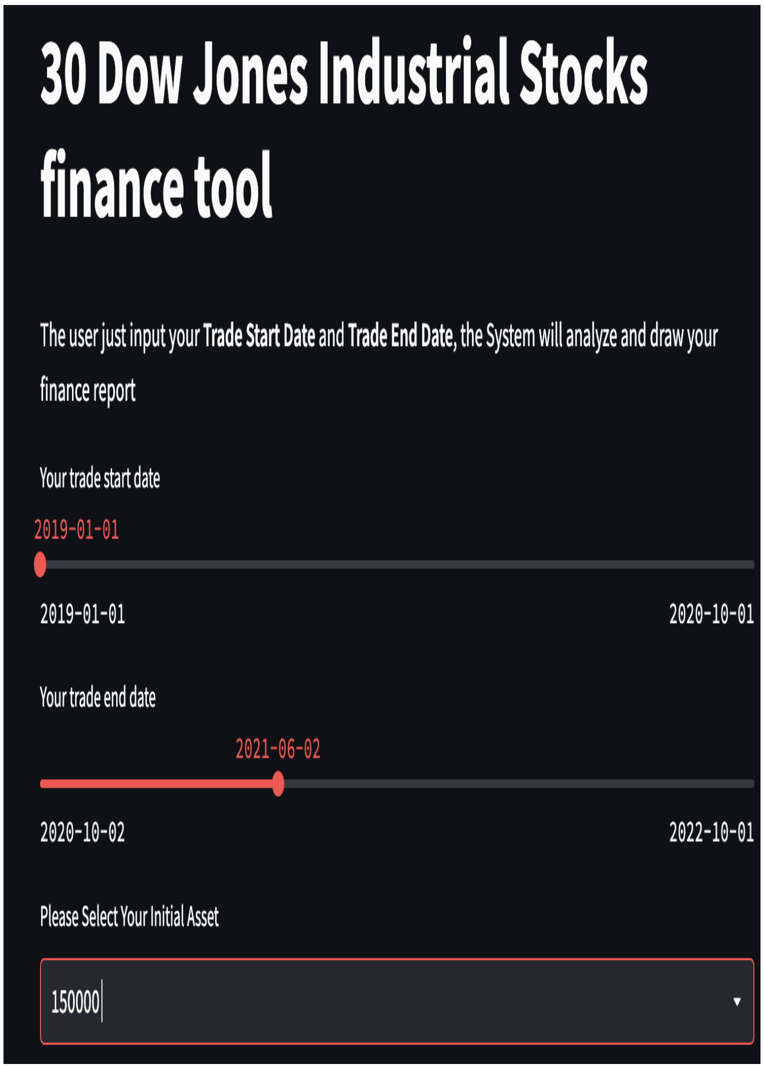
Task: Add Post

Add Your Post

Your question:

Figure 12: Home Pages

Our system main function is about the Dow-30 and NASDAQ-100 pages. You could find them on the left menu bar of home page. In both two pages, the users input the trade start date, trade end data and your initial asset. The system will obtain the corresponding data and load the pretrained models to calculate the profit and generate the finance report. At the same time, your everyday action and cumulative return will be returned by system. We believe these report definitely help you on portfolio allocation and selection of stocks. or algorithms.



30 Dow Jones Industrial Stocks finance tool

The user just input your **Trade Start Date** and **Trade End Date**, the System will analyze and draw your finance report

Your trade start date

2019-01-01

2019-01-01 2020-10-01

Your trade end date

2021-06-02

2020-10-02 2022-10-01

Please Select Your Initial Asset

150000

Figure 13: User Input

5.2 Streamlit Framework usage

Streamlit is an open source python based framework for developing and deploying interactive data science dashboards and machine learning models. This means that you do not have to rely on a team of front end developers or spend large amounts of time learning web design languages such as HTML, CSS or Javascript in order to deploy your dashboard or model.

1) **Easy and Fast**:For the data scientist or Machine learning Engineer you don't need to study knowledge about backend and frontend.

2) **Support on Python**:Because our many algorithms is written on Python. At the same time, like the object detection or prediction time series, which are all on the Python and many packages could support on the community.

3) **Easy to deploy**:About the deployment on the server, you don't need to worry. You even don't need to configure the environment, just push your project on the Github and you could deploy on the streamlit cloud successfully.

6 Conclusion and Limitation

In summary, our model achieves the basic goals we set out at the beginning, and achieves good results. Our DRL training part consume much time because we need to try multi algorithms, however we get the good result. The selection on different period also cause different, hence we try different periods. At first, our NASDAQ-100 always get negative profit in 2022. We all know the real reason is about the circumstance of down US stocks. In other words, our trained models could not fit the environment fast. But when we change the period about the training part which include the rapidly drop. So our models could adapt the environment to get more profit during the rapid drop period. For trade part, the policy will give different actions regard to the states. Because our critic network is pre-trained, the system could just collect the data from the real time database and do the feature engineer. Subsequently, the system could select the proper model and calculate the action and reward according to the environment. Because our backend framework is simple we design a simple login system to verify the user's legal identity. Another shortcoming is that we don't use sliding window to train the models and do the trade. It may lead to our models will be outdated as time passed by. The last point I would like to mention is that the different initial asset will lead to different actions, cumulative return and portfolio allocation.

7 Future Work

Our outlook for the future of our project is as follows. First of all, we hope to find more excellent algorithm to fit the environment and could help the users to get more profit. We also hope to use better framework to display the result for users. In another hand, the DRL could also be used on different areas such as game, planning, optimization. In multi agents areas, it could play a role in the robotics field. We believe human could utilize these to creat a new style of life.

8 Reference and Datasets

Our System Website Link:

https://sezer12138-finance-tool-home-rnjqsu.streamlitapp.com/

The APP Source Code:

https://github.com/sezer12138/finance-tool

Dow-30 and NASDAQ-100 Stocks

https://finance.yahoo.com/quotes/API,Documentation/view/v1/

The Algorithms Sources

https://github.com/AI4Finance-Foundation/FinRL

ACM International Conference on AI in Finance:

https://arxiv.org/abs/2111.03995

NeurIPS 2021 Data-Centric AI Workshop

https://arxiv.org/abs/2112.06753

9 Appendix

9.1 Appendix A

Item List:

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5