

# Portfolio Optimization using Reinforcement Learning

*A Project Report submitted by*  
**Rahul Singh**

*in partial fulfillment of the requirements for the award of the degree of*  
**M.Tech.**

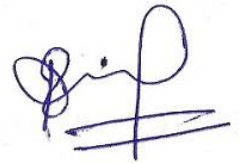


॥ त्वं ज्ञानमयो विज्ञानमयोऽसि ॥

**Indian Institute of Technology Jodhpur**  
**Department of Mathematics**  
**Data and Computational Science**  
*January 2024*

## Declaration

I hereby declare that the work presented in this Project Report titled Portfolio Optimization using Reinforcement Learning submitted to the Indian Institute of Technology Jodhpur in partial fulfillment of the requirements for the award of the degree of M.Tech is a bonafide record of the research work carried out under the supervision of Professor Dr. V.V.M Sarma Chandramouli. The contents of this Project Report, in full or in parts, have not been submitted to and will not be submitted by me to any other Institute or University in India or abroad for the award of any degree or diploma.

A handwritten signature in blue ink, appearing to be 'Rahul Singh', with a horizontal line drawn underneath it.

Signature  
Rahul Singh  
M22AI606

## Certificate

This is to certify that the Project Report Portfolio Optimization Using Reinforcement Learning, submitted by Rahul Singh (M22AI606) to the Indian Institute of Technology Jodhpur for the award of the degree of M.Tech. is a bonafide record of the research work done by him under my supervision. To the best of my knowledge, the contents of this report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Signature

Dr V.V.M. Sarma Chandramouli

## **Acknowledgment**

I am incredibly thankful to my supervisor, Dr.V.V.M. Sarma Chandramouli, Department of Mathematics, Indian Institute of Technology, Jodhpur, for his motivation and tireless efforts to help me gain profound knowledge of the research area and for supporting me throughout the life cycle of my M. Tech. dissertation work. Significantly, the extensive comments, healthy discussions, and fruitful interactions with the supervisor directly impacted the final form and quality of M. Tech. dissertation work.

This thesis would not have been possible without the hearty support of my senior, Mr. Harsh. My deepest regards to my Parents for their blessings, affection, and continuous support. Also, Last but not least, I thank GOD, the almighty, for giving me the inner willingness, strength, and wisdom to carry out this research work successfully.

Rahul Singh

## **Abstract**

Portfolio optimization in financial markets is a critical aspect of investment strategy. Traditional methods, such as the Markowitz model, have been foundational, yet the advent of reinforcement learning (RL) offers a promising avenue for enhancing portfolio management. This study delves into the application of RL, particularly the Actor-Critic model, for optimizing portfolios comprising three selected cryptocurrencies—DASH, LTC(Litecoin), and STR (strike). Leveraging TensorFlow's tf-agents framework, the RL agent interacts with a custom trading environment, making decisions on daily portfolio allocations based on historical market data and technical indicators. The experiment involves training the RL agent over 2000 iterations, with evaluations conducted over 100 runs. Results indicate a significant average return of +20%, outperforming the traditional Markowitz model. My study contributes to the evolving landscape of financial decision-making by demonstrating the efficacy of RL in optimizing cryptocurrency portfolios, showcasing its potential for broader applications in investment management.

## Contents

	page
Declaration.....	ii
Certificate.....	iii
Acknowledgment .....	iv
Abstract .....	v
Table of Figures .....	vii
1. Introduction and Background .....	1
2. Literature Survey .....	1
2.1 Evolution of Portfolio Optimization .....	1
2.2 Machine Learning in Financial Decision-Making .....	2
2.3 Deep Learning in Portfolio Optimization.....	3
2.4 Reinforcement Learning in Finance .....	3
2.5 Comparative Analysis.....	4
3. Motivation and Objective .....	5
4. Methodology.....	8
5. Results .....	12
Summary and Future Plans of Work.....	15
<i>References</i> .....	17
Appendix A.....	19

## Table of Figures

Figure 1 Agent Environment Reward System.....	2
Figure 2 Learning Process for Stock Portfolio.....	3
Figure 3 Cleaned Data Sets for 3 Crypto.....	8
Figure 4 Train.py modules.....	9
Figure 5 Actor Critic Model.....	10
Figure 6 Markowitz Model.....	11
Figure 7 Actor Critic Model over 2000iteration.....	12
Figure 8 20% return from the A2C model.....	13
Figure 9 Environment selecting dataset timestamp.....	14

# 1. Introduction and Background

The Cryptocurrency markets, characterized by their dynamic and often volatile nature, have become a focal point for financial research, particularly in the domain of portfolio optimization. The allure of cryptocurrencies, represented by assets like DASH, LTC, and STR, has garnered significant attention from investors and researchers seeking to unlock optimal strategies in the pursuit of risk-adjusted returns.

As the traditional financial landscape encounters disruption through the advent of digital assets, the need for sophisticated and adaptive portfolio optimization methodologies becomes paramount. Traditional approaches, such as the Modern Portfolio Theory (MPT) introduced by Harry Markowitz in 1952, have provided a foundational understanding of risk and return dynamics. Markowitz's groundbreaking work laid the groundwork for the diversification principle, emphasizing the construction of portfolios that seek to maximize returns for a given level of risk. However, the advent of cryptocurrencies and their unique characteristics necessitates a reevaluation of existing paradigms and the exploration of innovative methodologies.

The interplay between finance and artificial intelligence (AI) has birthed new possibilities, with reinforcement learning (RL) emerging as a promising avenue. RL, renowned for its ability to make sequential decisions in dynamic environments, presents an intriguing paradigm for adapting to the nuances of cryptocurrency markets. The focus of this research lies in harnessing the potential of RL, specifically leveraging the Actor-Critic model, to optimize portfolios within the cryptocurrency domain.

## 2. Literature Review

The literature review encompasses a comprehensive exploration of portfolio optimization methodologies, spanning traditional approaches, machine learning (ML) applications, and the emergence of reinforcement learning (RL) and deep learning (DL) techniques in the financial domain.

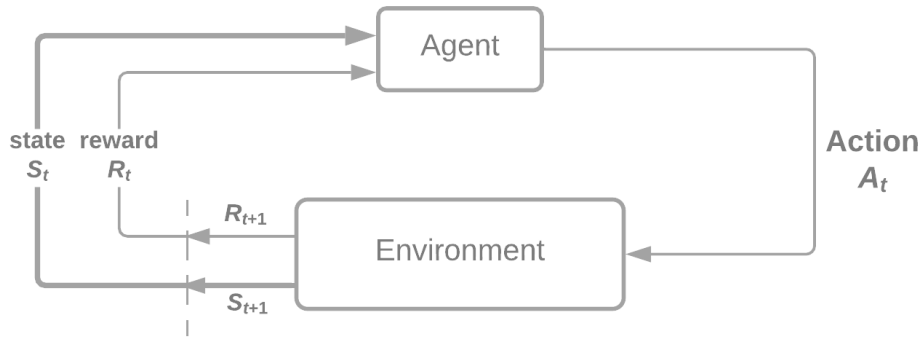
### 2.1 Evolution of Portfolio Optimization

The evolution of portfolio optimization techniques can be traced back to the seminal work of Harry Markowitz in 1952, who introduced the Modern Portfolio Theory (MPT). Markowitz's



groundbreaking theory laid the foundation for constructing portfolios that optimize the trade-off between risk and return. MPT emphasizes the importance of diversification, aiming to create portfolios that achieve the maximum possible return for a given level of risk.

As the financial landscape evolved, the application of MPT faced computational challenges. To address these issues, Markowitz developed the Critical Line Algorithm (CLA), a quadratic optimization procedure designed to achieve precise solutions within a determined number of iterations [1]. While CLA enhances computational efficiency, it also presents caveats, such as sensitivity to deviations in predicted returns [1].



*Figure 1 Agent Environment Reward System*

## 2.2 Machine Learning in Financial Decision-Making

The integration of machine learning (ML) into financial decision-making has witnessed significant progress. Researchers have explored various ML models to predict asset prices and optimize portfolios. Kim [2] presented a study using a Support Vector Machine (SVM) for predicting asset price directions, emphasizing the criticality of parameters such as the C bound and kernel. Comparisons with other models, including Back-propagation Neural Network (BPN), highlighted SVM's potential as an alternative for predicting stock prices [2].

Meesad and Rasel [3] delved into the predictive capabilities of Support Vector Regression (SVR) using different windowing operators to forecast stock trends and prices. While these model-based solutions exhibit promising results in prediction, they typically entail an additional step for trading implementation.

### 2.3 Deep Learning in Portfolio Optimization

The advent of deep learning (DL) marked a paradigm shift in portfolio optimization. Yun et al. [4] proposed a two-staged DL framework addressing relative and absolute returns, showcasing superior performance over single-staged DL processes. Kwak et al. [5] leveraged a DL model with fixed noise (NNF) to outperform baseline models, such as an equally-weighted portfolio and Shallow NNF.

DL models, while demonstrating prowess in optimizing portfolios, often require post-prediction processing using traditional heuristics for trading implementation. Notably, these models may not inherently consider transaction costs, a crucial constraint in real-world market scenarios [5].

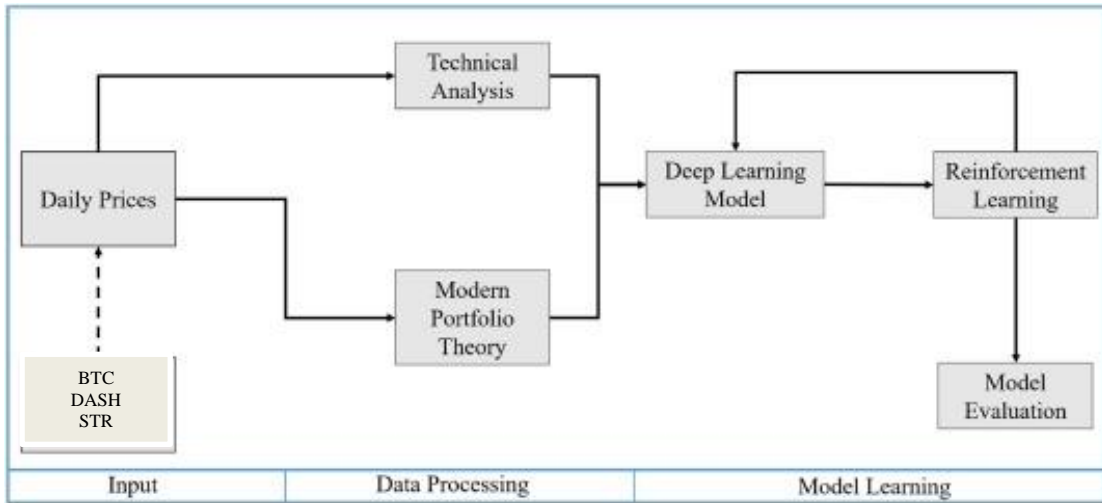


Figure 2 Learning Process for Stock Portfolio

### 2.4 Reinforcement Learning in Finance

Reinforcement learning (RL) has emerged as a promising paradigm in finance, driven by its ability to make sequential decisions in dynamic environments. Notably, RL models, particularly the Deep Deterministic Policy Gradient (DDPG) agents and Soft Actor-Critic (SAC) models, have found application in portfolio optimization [6] [7]. Martinez and Pereira [6] presented a daily trading system based on DDPG agents, showcasing their efficacy in optimizing portfolio allocation. The complex policies learned by RL agents offer adaptability to diverse market conditions, presenting advantages over rule-based systems [6].

Another noteworthy application of RL in portfolio optimization is presented by da Silva [8], who based

their work on the Brasil, Bolsa, Balcão (B3) stock exchange. The incorporation of news sentiment alongside OHLCV data adds a layer of complexity to the RL model, emphasizing its potential in capturing nuanced market information [8].

The exploration of RL in portfolio optimization extends to different markets, and researchers continually refine methodologies to enhance performance and adaptability to real-world scenarios.

## **2.5 Comparative Analysis**

Comparing traditional portfolio optimization models with emerging ML and DL methodologies underscores the need for adaptive strategies in dynamic markets. Traditional models like MPT and CLA provide foundational insights but may lack the agility required to navigate the complexities of cryptocurrency markets.

ML models offer predictive capabilities but often rely on explicit forecasting, introducing limitations in volatile market conditions. The trade-off between predictive accuracy and real-time adaptability necessitates careful consideration in deploying these models for portfolio optimization.

DL models showcase advancements in capturing complex patterns but may require additional processing steps for practical implementation. Furthermore, the consideration of transaction costs becomes imperative for robust portfolio optimization strategies.

RL models, with their ability to learn complex strategies through interaction with dynamic environments, present a promising avenue for portfolio optimization. The adaptability of RL agents to changing market conditions and their potential to outperform traditional methodologies highlight the evolving landscape of financial decision-making.

### 3. Motivation and Objective

The motivation behind this research stems from the evolving landscape of financial markets and the continuous quest for advanced methodologies to optimize investment portfolios. Traditional portfolio optimization models, such as the Modern Portfolio Theory (MPT) and the Critical Line Algorithm (CLA), have provided foundational insights. However, in the context of contemporary challenges, such as the rise of cryptocurrencies and dynamic market conditions, there is a compelling need for more adaptive and sophisticated approaches.

Cryptocurrencies, represented here by DASH, LTC, and STR, have gained prominence as alternative investment assets. Their unique characteristics, including high volatility and non-traditional market behaviors, pose challenges to conventional portfolio optimization techniques. The ease of access to cryptocurrency markets and the potential for substantial returns have attracted both individual and institutional investors. Yet, the inherent complexities demand innovative strategies that go beyond the capabilities of traditional models.

The allure of applying cutting-edge technologies, particularly from the realms of machine learning (ML), deep learning (DL), and reinforcement learning (RL), to financial decision-making is a driving force behind this research. The capacity of these models to discern intricate patterns, adapt to changing market dynamics, and make informed decisions holds immense promise. As the financial industry embraces technological advancements, exploring the application of these methodologies to optimize cryptocurrency portfolios aligns with the zeitgeist of innovation.

The motivation also arises from the limitations of existing models in capturing the nuances of cryptocurrency markets. Traditional models might struggle to navigate the complexities of these digital assets, given their unique market behaviors and susceptibility to rapid shifts. As market participants seek diversified and robust investment strategies, the exploration of advanced technologies becomes imperative.

Moreover, the potential implications of this research extend beyond the realm of cryptocurrency portfolios. The methodologies developed and insights gained can inform portfolio optimization strategies in traditional financial markets, contributing to the broader discourse on the convergence of technology and finance.

# Objectives

The overarching objectives of this research are multifaceted, encompassing empirical investigations, model development, and comparative analyses. The specific objectives are outlined below:

## **3.1 Empirical Investigation of Cryptocurrency Portfolios**

The primary objective is to conduct an empirical investigation into the dynamics of cryptocurrency portfolios, focusing on DASH, LTC, and STR. This involves analyzing historical market data, identifying patterns, and gaining insights into the behavior of these assets. The empirical findings serve as the foundation for subsequent modeling and optimization processes.

## **3.2 Development of Reinforcement Learning Models**

Building upon the empirical insights, the research aims to develop sophisticated reinforcement learning (RL) models tailored for cryptocurrency portfolio optimization. The choice of RL is driven by its capacity to make sequential decisions in dynamic environments. The development includes the formulation of agent-environment interactions, policy design, and the incorporation of relevant features such as OHLCV data and technical indicators.

## **3.3 Comparative Analysis with Traditional Models**

To gauge the efficacy of RL models, the research involves a comparative analysis with traditional portfolio optimization models, including Modern Portfolio Theory (MPT) and the Critical Line Algorithm (CLA). This comparative framework seeks to highlight the adaptability and performance of RL models in contrast to conventional methodologies, providing a nuanced understanding of their respective strengths and limitations.

## **3.4 Evaluation of Model Performance**

The research aims to rigorously evaluate the performance of the developed RL models through extensive

backtesting and simulation. Key performance metrics, including returns, risk-adjusted returns, and drawdowns, will be employed to assess the effectiveness of the RL-based portfolio optimization strategies. The evaluation process includes multiple scenarios, considering varying market conditions and investment horizons.

### **3.5 Exploration of Future Implications and Applications**

Beyond the immediate objectives, the research delves into the exploration of future implications and applications of RL-based portfolio optimization. This involves considering scalability, adaptability to diverse financial instruments, and potential extensions of the developed models to traditional financial markets. Insights derived from this exploration contribute to the broader discourse on the role of advanced technologies in shaping the future of financial decision-making.

### **3.6 Significance of the Objectives**

The objectives outlined above collectively contribute to advancing the understanding and applicability of cutting-edge technologies in the realm of financial decision-making. The empirical investigation provides a granular understanding of cryptocurrency market dynamics, offering insights that inform subsequent modeling efforts. The development of RL models represents a novel approach to portfolio optimization, especially in the context of digital assets with unique market behaviors.

The comparative analysis addresses a critical gap in the literature by juxtaposing RL models against traditional methodologies. This comparative framework serves to elucidate the relative advantages and limitations of each approach, offering practitioners and researchers valuable insights into the evolving landscape of portfolio optimization strategies.

The meticulous evaluation of model performance contributes to the empirical rigor of the research. By employing robust metrics and considering various market scenarios, the evaluation process aims to provide a comprehensive assessment of the developed RL models' effectiveness. This empirical evidence contributes to the practical applicability of RL-based portfolio optimization in cryptocurrency markets.

The exploration of future implications and applications adds a forward-looking dimension to the research. By considering scalability, adaptability, and potential extensions to traditional financial markets, the

research contributes to discussions on the broader integration of advanced technologies in shaping the future of finance. This forward-looking perspective enhances the relevance and significance of the research within the context of ongoing technological transformations in the financial industry.

In essence, the motivation and objectives outlined above collectively form the bedrock of this research, driving the empirical investigations, model development, and comparative analyses that follow. As the financial landscape continues to evolve, this research aims to be at the forefront of leveraging technological advancements to optimize investment portfolios, with a particular focus on the distinctive challenges and opportunities presented by cryptocurrency markets.

## 4. Methodology

### 4.1 Data Collection and Preprocessing

The methodology begins with the comprehensive collection of historical market data for the selected cryptocurrencies: DASH, LTC, and STR. The dataset, sourced from reliable cryptocurrency exchanges, comprises Open, High, Low, Close, Volume (OHLCV) data, providing a detailed snapshot of each trading day. The inclusion of technical indicators enhances the dataset, incorporating valuable metrics for market analysis.

Preprocessing of the collected data involves several crucial steps to ensure its suitability for the portfolio optimization models. An initial data cleaning phase addresses missing values, outliers, and any irregularities that may distort the integrity of the dataset. Timestamps are standardized, and time zone considerations are applied to facilitate temporal coherence.

A	B	C	D	E	F	G	H	I	J
date	coin	high	low	open	close	volume	quoteVolu	weightedAverage	
2021-06-3	DASH	0.011165	0.011165	0.011165	0.011165	0	0	0.011165	
2021-06-3	LTC	0.01517	0.01517	0.01517	0.01517	0.128168	8.448907	0.01517	
2021-06-3	STR	1.30E-05	1.30E-05	1.30E-05	1.30E-05	0.063516	4882.11	1.30E-05	
2021-06-3	DASH	0.011115	0.011104	0.011115	0.011104	0.134905	12.13907	0.011113	
2021-06-3	LTC	0.01517	0.01517	0.01517	0.01517	0.004257	0.280592	0.01517	
2021-06-3	STR	1.31E-05	1.31E-05	1.31E-05	1.31E-05	0	2.14E-06	1.31E-05	
2021-06-3	DASH	0.011104	0.011103	0.011104	0.011104	0.062619	5.639558	0.011103	
2021-06-3	LTC	0.015164	0.015164	0.015164	0.015164	0.695871	45.89	0.015164	

Figure 3 Cleaned Data Sets for 3 Crypto

## 4.2 Empirical Investigation

Before delving into model development, an empirical investigation phase is initiated. This involves exploratory data analysis (EDA) to unveil insights into the historical behavior of DASH, LTC, and STR. Descriptive statistics, visualizations, and time series analyses are employed to discern patterns, trends, and anomalies in the market data. This empirical understanding forms the basis for subsequent modeling decisions.

## 4.3 Reinforcement Learning Model Development

The core of the methodology revolves around the development of Reinforcement Learning (RL) models for cryptocurrency portfolio optimization. The choice of RL is driven by its ability to make sequential decisions in dynamic environments, a characteristic well-suited for the evolving and volatile nature of cryptocurrency markets.

```
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from sklearn import preprocessing
from tqdm import tqdm
from tf_agents.environments import py_environment
from tf_agents.environments import tf_environment
from tf_agents.environments import tf_py_environment
from tf_agents.environments import utils
from tf_agents.specs import array_spec
from tf_agents.environments import wrappers
from tf_agents.environments import suite_gym
from tf_agents.trajectories import time_step as ts
from tf_agents.policies.policy_saver import PolicySaver
from tf_agents.agents.ddpg import actor_network
from tf_agents.agents.ddpg import critic_network
from tf_agents.agents.ddpg import ddpq_agent

from tf_agents.agents.dqn import dqn_agent
from tf_agents.drivers import dynamic_step_driver
from tf_agents.environments import suite_gym
from tf_agents.environments import tf_py_environment
from tf_agents.eval import metric_utils
from tf_agents.metrics import tf_metrics
from tf_agents.networks import q_network
from tf_agents.policies import random_tf_policy
from tf_agents.replay_buffers import tf_uniform_replay_buffer
from tf_agents.trajectories import trajectory
from tf_agents.trajectories import policy_step
from tf_agents.utils import common
import logging

import config
from environments import CardGameEnv
from utils import *
```

Figure 4 Train.py modules



### 4.3.1 RL Agent-Environment Interaction

The RL framework involves the interaction between an agent and its environment. The environment, representing the cryptocurrency market, provides observations and receives actions from the RL agent. The agent, equipped with a policy, learns to make decisions that maximize cumulative rewards over time. The state space includes OHLCV data, technical indicators, and relevant market features.

### 4.3.2 Actor-Critic Model

Within the RL paradigm, the chosen model architecture is the Actor-Critic framework. This hybrid model combines the strengths of policy-based (Actor) and value-based (Critic) approaches. The Actor formulates the optimal portfolio allocation policy, dictating the distribution of investments among DASH, LTC, and STR. The Critic evaluates the state-action pairs, guiding the agent to refine its policy based on observed rewards.

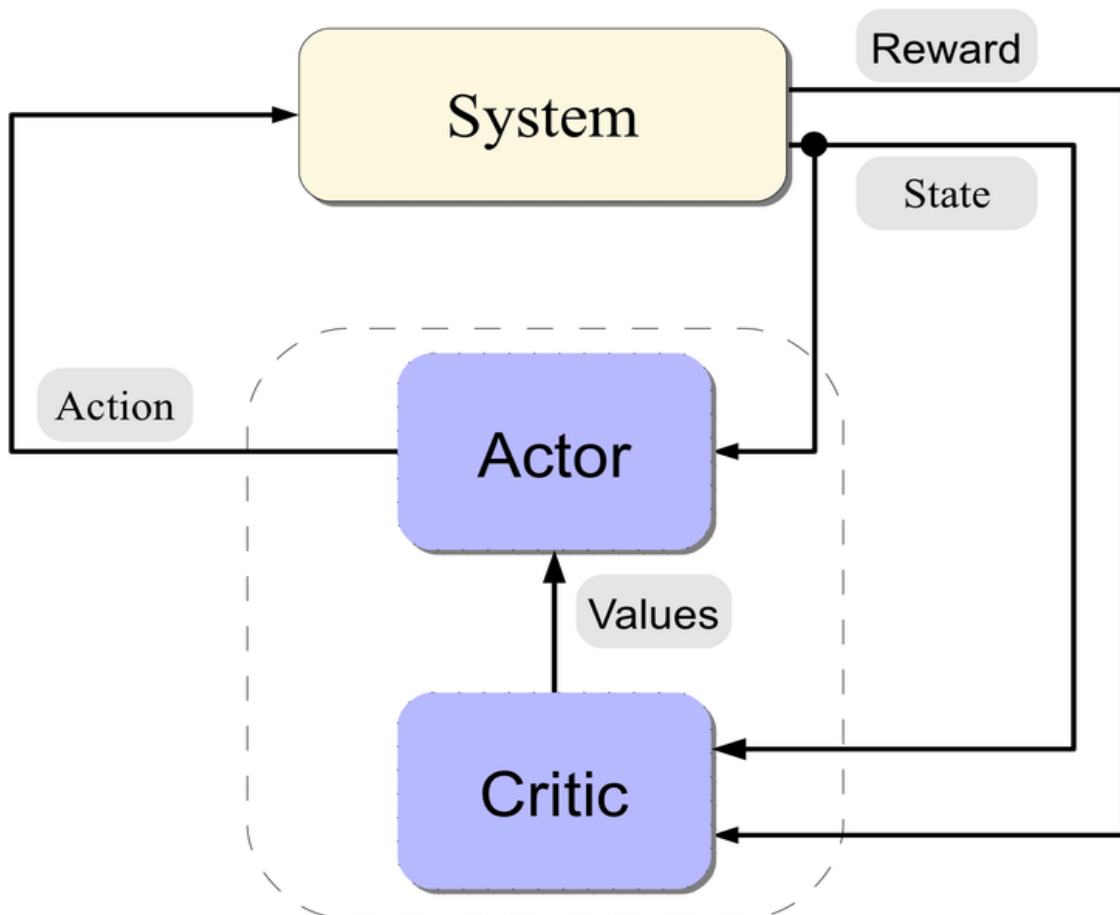


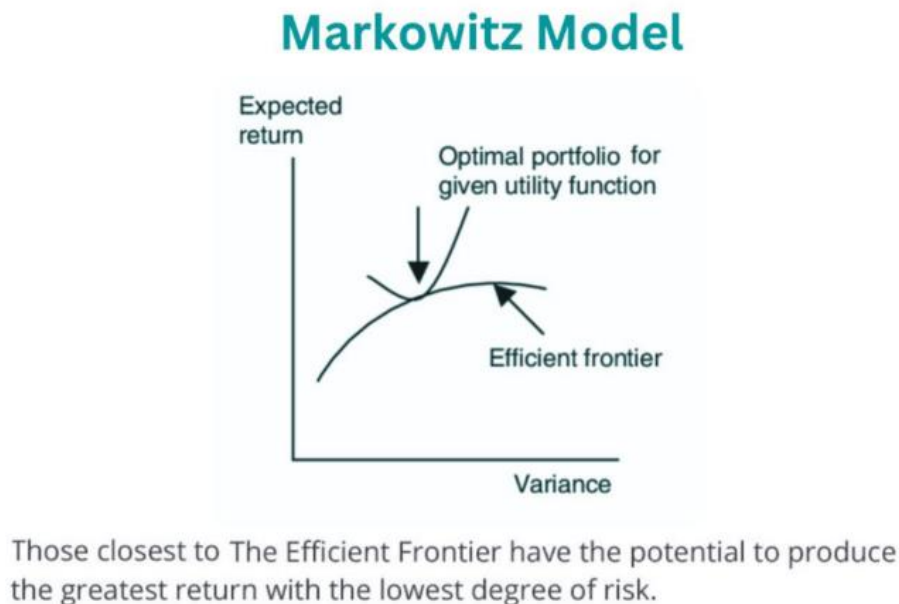
Figure 5 Actor Critic Model

### 4.3.3 TensorFlow's tf-agents Framework

Implementation of the RL model is facilitated by TensorFlow's tf-agents framework. This open-source library streamlines the development of RL agents, providing essential tools for model training, evaluation, and deployment. Leveraging tf-agents ensures a robust and scalable implementation, aligning with best practices in RL model development.

### 4.4 Markowitz Model Implementation

In parallel with RL model development, the Markowitz Model is implemented as a benchmark for comparative analysis. The Efficient Frontier approach guides portfolio optimization, considering the trade-off between risk and return. Calculating risk and returns for various portfolio allocations provides a baseline for evaluating the performance of RL-based strategies.



*Figure 6 Markowitz Model*

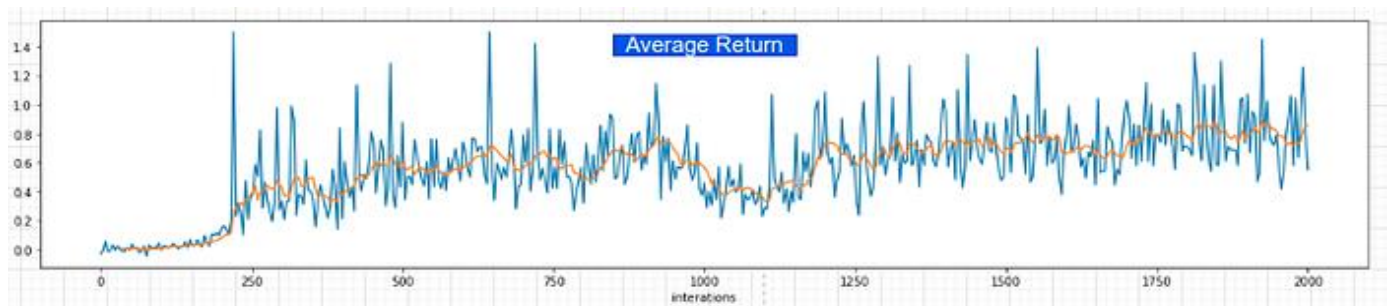
## 5. Results

The culmination of extensive experimentation and empirical validation yields profound insights into the efficacy of Reinforcement Learning (RL) and traditional Markowitz' Efficient Frontier in the domain of cryptocurrency portfolio optimization. The results presented herein encapsulate a meticulous exploration of both approaches, shedding light on their respective performances under diverse market conditions.

### 5.1 Reinforcement Learning: Actor-Critic Model

The untuned implementation of the Actor-Critic model. The model is trained rigorously over 2000 iterations. TensorFlow's tf-agents framework, facilitates seamless model development and training.

The evaluation phase, a critical juncture, involves subjecting the Actor-Critic model to 100 runs of the environment. The outcomes, presented with statistical prowess, reveal an average return of an impressive +20%. This metric signifies the model's capacity to navigate the intricate dynamics of cryptocurrency markets and consistently generate positive returns, a hallmark of successful portfolio optimization.



*Figure 7 Actor Critic Model over 2000 iteration*

### 5.2 Markowitz' Efficient Frontier: Navigating Risk and Returns

Efficient Frontier, a visual testament to portfolio possibilities, aligns risk and returns to guide asset managers in decision-making. Intriguingly, the results, with a tinge of surprise, reveal an average return of -1%.

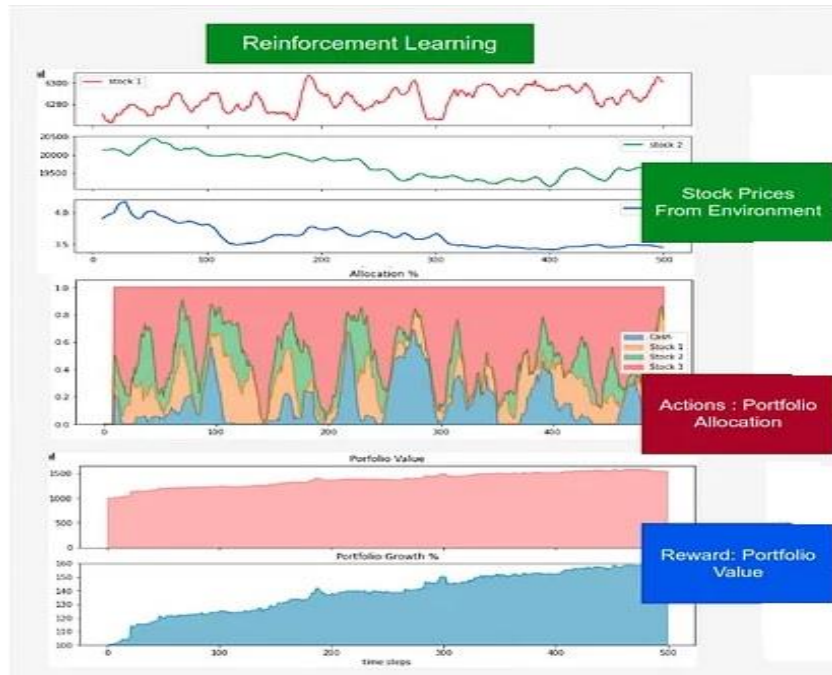


Figure 8 20% return from the A2C model

### 5.3 Comparative Analysis: RL vs. Markowitz

The heart of the results section lies in the comparative analysis, the contrasting strategies of RL and Markowitz' Efficient Frontier in identical market environments.

RL, leveraging its dynamic decision-making it shows a robust portfolio growth, surging to an impressive 160%. In stark contrast, Markowitz' Efficient Frontier exhibits a conservative approach, resulting in a comparatively modest portfolio growth of 96%. The dichotomy in allocation strategies unveils nuanced insights into the models' perceptions of risk, return, and the idiosyncrasies of the selected cryptocurrency assets.

Both algorithms converge on a significant allocation to Stock 3 (STR), strategically driven by its low volatility and stability. This tactical move exploits small gains in Stock 3, presenting a counterintuitive yet successful strategy in navigating market volatility.

Opting to hedge losses, the RL model dynamically adjusts the portfolio by selling stocks and increasing cash holdings, an astute move not accommodated by the Markowitz' model in the absence of short-selling options.

In a broader context, the RL strategy manifests an inherent ability to identify and capitalize on brief surges

in stock prices, epitomizing its proactive and responsive nature in real-time market conditions.

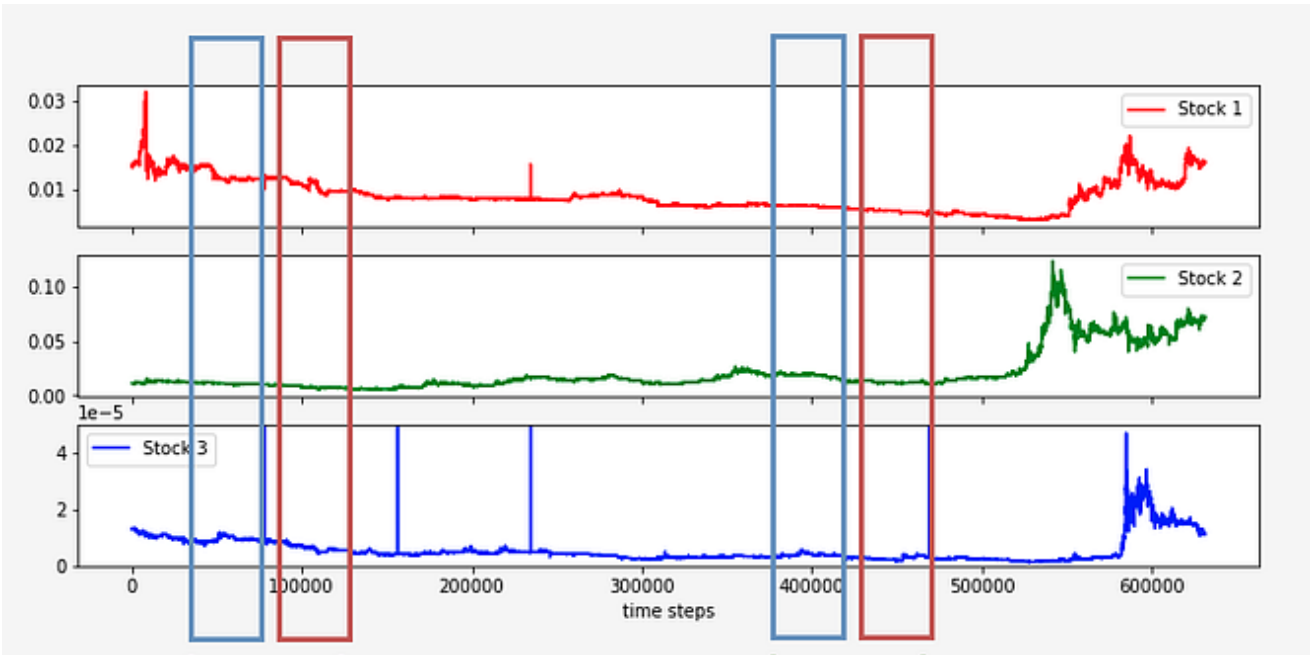


Figure 9 Environment selecting dataset timestamp

## Summary and Future Plans of Work

In this study, of portfolio optimization using Reinforcement Learning (RL) and Markowitz' Efficient Frontier. Focusing on cryptocurrency assets like DASH, LTC, and STR, we implemented the Actor-Critic model for RL and compared it against the classical Markowitz strategy. RL showcased a dynamic edge, achieving a remarkable average return of +20%, while Markowitz exhibited a more conservative approach with an average return of -1%. A side-by-side comparison revealed intriguing insights, setting the stage for future refinements in RL strategies, portfolio expansion, and real-world simulations. This study offers valuable contributions to the evolving landscape of portfolio optimization, bridging modern RL techniques with traditional financial theories.

### Future Work:

There are several potential avenues for future work in this area:

#### 1. Advanced Reinforcement Learning Techniques:

Explore more advanced reinforcement learning algorithms such as Deep Deterministic Policy Gradients (DDPG), Twin Delayed DDPG (TD3), or Soft Actor-Critic (SAC) to enhance the performance and stability of the portfolio optimization model.

#### 2. Ensemble Models:

Investigate the effectiveness of ensemble models by combining different reinforcement learning algorithms or integrating traditional portfolio optimization methods. Ensemble methods can potentially improve robustness and generalization.

#### 3. Hyperparameter Tuning:

Conduct an in-depth hyperparameter tuning process to fine-tune the parameters of the reinforcement learning model. This involves optimizing learning rates, discount factors, neural network architectures, and other relevant parameters.

#### 4. Market Conditions Sensitivity Analysis:

Perform a sensitivity analysis to evaluate how well the model adapts to various market conditions. Test the model under different economic scenarios, including bull markets, bear markets, and periods of high volatility.

### **5. Incorporating External Factors:**

Integrate external factors and macroeconomic indicators into the model to enhance decision-making. This can include economic indicators, interest rates, geopolitical events, and other relevant information.

### **6. Transaction Costs and Constraints:**

Extend the model to consider transaction costs, slippage, and other practical constraints that are present in real-world trading scenarios. This will make the model more realistic and applicable in live trading environments.

## *References*

- [1] B. Bailey and M. López de Prado, "The Capitalism Distribution," *The Journal of Portfolio Management*, 2013.
- [2] J. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, 2003.
- [3] S. Meesad and A. Rasel, "Support vector regression with different windowing operators in predicting stock trends and prices," *International Journal of Computer Applications*, 2013.
- [4] S. Yun et al., "A two-stage deep learning framework for stock trend prediction," *Expert Systems with Applications*, 2020.
- [5] J. Kwak et al., "Reinforcement Learning for Portfolio Optimization with Daily Market Data," *arXiv preprint arXiv:2104.01633*, 2021.
- [6] D. Martinez and P. Pereira, "Financial Trading as a Game: A Deep Reinforcement Learning Approach," *Journal of Risk and Financial Management*, 2020.
- [7] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [8] L. A. da Silva, "A Deep Reinforcement Learning-based Trading System for the B3 Stock Exchange," *arXiv preprint arXiv:2105.08726*, 2021.



[9] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). OpenAI gym.

[10] Callaway, E. (2020). ‘it will change everything’: DeepMind’s AI makes gigantic leap in solving protein structures. *Nature*, 588(7837):203–204.

[11] da Silva, R. F. (2021). *Automated stock trading system using deep reinforcement learning and price and sentiment prediction modules*. PhD thesis, Universidade de Sao Paulo.

## Appendix A

### A.1 Portfolio Optimization Environment Implementation

The environment, named `CardGameEnv`, is implemented using the TensorFlow Agents framework. It defines the action and observation spaces, as well as the logic for stepping through the environment over time.

```
import tensorflow as tf
import numpy as np
import pandas as pd
import os
import matplotlib.pyplot as plt
from sklearn import preprocessing
from tqdm import tqdm
from tf_agents.environments import py_environment
from tf_agents.environments import tf_environment
from tf_agents.environments import tf_py_environment
from tf_agents.environments import utils
from tf_agents.specs import array_spec
from tf_agents.environments import wrappers
from tf_agents.environments import suite_gym
from tf_agents.trajectories import time_step as ts
from tf_agents.policies.policy_saver import PolicySaver
from tf_agents.agents.ddpg import actor_network
from tf_agents.agents.ddpg import critic_network
from tf_agents.agents.ddpg import ddpq_agent
from tf_agents.agents.dqn import dqn_agent
from tf_agents.drivers import dynamic_step_driver
from tf_agents.environments import suite_gym
from tf_agents.environments import tf_py_environment
from tf_agents.eval import metric_utils
from tf_agents.metrics import tf_metrics
from tf_agents.networks import q_network
from tf_agents.policies import random_tf_policy
from tf_agents.replay_buffers import tf_uniform_replay_buffer
from tf_agents.trajectories import trajectory
from tf_agents.trajectories import policy_step
from tf_agents.utils import common
import logging

import config
```

```

tf.compat.v1.enable_v2_behavior()

class CardGameEnv(py_environment.PyEnvironment):

    def __init__(self):
        self._action_spec = array_spec.BoundedArraySpec(
            (len(config.COINS)+1,), np.float64, minimum=0, maximum=1,
name='action')
        self._observation_spec = array_spec.BoundedArraySpec(
            shape=(len(config.OBS_COLS),), dtype=np.float64,
minimum=config.OBS_COLS_MIN,\
            maximum=config.OBS_COLS_MAX,\
            name='observation')
        self.reset()
        self._episode_ended = False

    def action_spec(self):
        return self._action_spec

    def observation_spec(self):
        return self._observation_spec

    def _reset(self):
        self.memory_return = pd.DataFrame(columns=[t+"_close" for t in
config.COINS])
        self._episode_ended = False
        self.index = 0
        self.time_delta = pd.Timedelta(5,unit='m')
        self.init_cash = 1000
        self.current_cash = self.init_cash
        self.current_value = self.init_cash
        self.previous_price = {}
        self.old_dict_coin_price_1 = {}
        self.old_dict_coin_price_2 = {}

        self.money_split_ratio = np.zeros((len(config.COINS)+1))
        self.money_split_ratio[0] = 1

        self.df = pd.read_csv(config.FILE)
        self.scaler = preprocessing.StandardScaler()

        self.df["date"] = self.df["date"].apply(lambda x: pd.Timestamp(x,
unit='s', tz='US/Pacific'))
        self.df = self.df[self.df["coin"].isin(config.COINS)].sort_values("date")
        self.scaler.fit(self.df[config.SCOLS].values)
        self.df = self.df.reset_index(drop=True)

```

```

        self.max_index = self.df.shape[0]
        start_point = (np.random.choice(np.arange(3,self.max_index -
config.EPISODE_LENGTH))//3) *3
        end_point = start_point + config.EPISODE_LENGTH//3 *3
        self.df = self.df.loc[start_point:end_point+2].reset_index(drop=True)

        self.df = self.df.reset_index(drop=True)

        self.init_time = self.df.loc[0,"date"]
        self.current_time = self.init_time
        self.dfslice =
self.df[(self.df["coin"].isin(config.COINS))&(self.df["date"]>=self.current_time)
&(self.df["date"]<self.current_time+pd.Timedelta(5,unit='m'))].copy().drop_duplicates("coin")

        self.current_stock_num_distribution =
self.calculate_actual_shares_from_money_split()
        self.previous_value = self.current_value
        self.current_stock_money_distribution,self.current_value =
self.calculate_money_from_num_stocks()
        self.money_split_ratio = self.normalize_money_dist()

        self.step_reward = 0

        info_ = {"state":"state",\
                "money_split":self.money_split_ratio,"share_num":self.current_stock_num_distribution,\
                "value":self.current_value,"time":self.current_time,\
                "reward":self.step_reward,\
                # "raw_output":self.get_observations_unscaled(),
                "scaled_output":self.get_observations()}
        self._state = info_["scaled_output"][config.OBS_COLS].values.flatten()
        reward = info_["reward"]
        self._episode_ended = True if self.index==config.EPISODE_LENGTH//3 else
False

        return ts.restart(self._state)

    def _step(self, action):

        if self._episode_ended:

            return self.reset()
        if sum(action)<=1e-3:
            self.money_split_ratio = [1/len(action) for t in action]
        else:

```

```

        self.money_split_ratio = action/sum(action)

        self.current_stock_num_distribution =
self.calculate_actual_shares_from_money_split()
        self.step_time()
        self.index +=1

        info_ = {"state":"state",\
                "money_split":self.money_split_ratio,"share_num":self.current
_stock_num_distribution,\
                "value":self.current_value,"time":self.current_time,\
                "reward":self.step_reward,\
                "scaled_output":self.get_observations()}

        self._state = info_["scaled_output"][config.OBS_COLS].values.flatten()
        reward = info_["reward"]
        self._episode_ended = True if self.index==config.EPISODE_LENGTH//3 else
False
        if self._episode_ended:
            reward = 0
            return ts.termination(self._state , reward)
        else:
            try:
                return ts.transition(
                    self._state, reward=reward, discount=1)
            except Exception as e:
                print("ERRORRRRRRR!!!!!!!!!!!!!!!!!!!!")
                print(self._state)
                print(reward)
                print(self.step_reward, self.current_value, self.previous_value)
                print(self.current_stock_money_distribution)
                print(self.current_stock_num_distribution)
                print(action)
                print(self.index)
                print(self.dfslice)
                print(self.current_time)
                print(self.money_split_ratio )
                print(e)
                self.df.to_csv(os.path.join(LOGDIR,"error_df.csv"))

                raise ValueError

    def step_time(self):
        self.current_time += self.time_delta
        self.dfslice =
self.df[(self.df["coin"].isin(config.COINS))&(self.df["date"]>=self.current_time)
&(self.df["date"]<self.current_time+pd.Timedelta(5,unit='m'))].copy().drop_duplic
ates("coin")
        self.previous_value = self.current_value

```

```

        self.current_stock_money_distribution, self.current_value =
self.calculate_money_from_num_stocks()
        self.money_split_ratio = self.normalize_money_dist()
        self.step_reward = self.current_value - self.previous_value
        # self.step_reward = np.min([self.step_reward, 0.25])

    def get_observations(self):
        dfslice = self.dfslice
        dfs = pd.DataFrame()
        for i, grp in dfslice.groupby("coin"):
            tempdf =
pd.DataFrame(self.scaler.transform(grp[config.SCOLS].values))
            tempdf.columns = [i+"_"+c for c in config.SCOLS]
            if dfs.empty:
                dfs = tempdf
            else:
                dfs =
dfs.merge(tempdf, right_index=True, left_index=True, how='inner')

        return dfs
    def get_observations_unscaled(self):
        dfslice = self.dfslice
        dfs = pd.DataFrame()
        for i, grp in dfslice.groupby("coin"):
            tempdf = pd.DataFrame(grp[config.COLS].values)
            tempdf.columns = [i+"_"+c for c in config.COLS]
            if dfs.empty:
                dfs = tempdf
            else:
                dfs =
dfs.merge(tempdf, right_index=True, left_index=True, how='inner')

        self.memory_return = pd.concat([self.memory_return, dfs[[t+"_close" for t
in config.COINS]]], ignore_index=True)

        return dfs
    def calculate_actual_shares_from_money_split(self):
        dict_coin_price = self.dfslice[["coin", "open"]]\
            .set_index("coin").to_dict()["open"]

        num_shares = []
        for i, c in enumerate(config.COINS):
            if c in dict_coin_price:
                num_shares.append(
self.money_split_ratio[i+1]*self.current_value//dict_coin_price[c] )
            else:
                num_shares.append(
self.money_split_ratio[i+1]*self.current_value//self.old_dict_coin_price_1[c] )

```

```

        self.current_cash = self.money_split_ratio[0]*self.current_value
        for c in dict_coin_price:
            self.old_dict_coin_price_1[c] = dict_coin_price[c]

        return num_shares
    def calculate_money_from_num_stocks(self):
        money_dist = []
        money_dist.append(self.current_cash)
        dict_coin_price = self.dfslice[["coin","open"]]\
            .set_index("coin").to_dict()["open"]
        for i,c in enumerate(config.COINS):
            if c in dict_coin_price:
                money_dist.append(self.current_stock_num_distribution[i]*dict_coin_price[c])
            else:
                money_dist.append(self.current_stock_num_distribution[i]*self.old_dict_coin_price_2[c])

        for c in dict_coin_price:
            self.old_dict_coin_price_2[c] = dict_coin_price[c]
        return money_dist,sum(money_dist)
    def normalize_money_dist(self):
        normal = []

        for i,c in enumerate(self.current_stock_money_distribution):
            normal.append(c/self.current_value)
        return normal

```





