Optimizing Stock Trading Strategy with Reinforcement Learning

PROBLEM STATEMENT

We propose to determine a single exchange-traded fund (SPY) investing strategy that will maximize our total wealth. We compare our findings to the tried-and-true “buy-and-hold” and Moving average convergence divergence (MACD) strategies.Reinforcement learning is all about taking the right steps to maximize your reward in a given situation. It is used by a variety of software and computers to determine the best feasible action or path in a given situation. The Reinforcement Machine Learning mod.

DESCRIPTION ON THE PROBLEM STATEMENT

The project, "Optimizing Stock Trading Strategy with Reinforcement Learning," endeavors to create an advanced algorithmic framework using reinforcement learning (RL) techniques to revolutionize stock market trading strategies. This initiative aims to leverage historical market data, sophisticated machine learning algorithms, and dynamic decision-making processes to optimize trading actions, maximize returns, and minimize risks in the volatile and complex landscape of stock markets.

Overview: The project involves integrating cutting-edge RL methodologies into the realm of stock trading, empowering an algorithmic agent to autonomously make trading decisions based on learned patterns from historical market data. This agent interacts with the stock market environment, continuously learning and adapting to changing conditions to improve its trading strategies.

Key Components and Objectives:

- Reinforcement Learning Integration: Constructing an RL model that defines the agent's actions (buy, sell, hold), observes states (market data, indicators), and receives rewards based on trading performance.
- Data Processing and Feature Engineering: Gathering and preprocessing vast amounts of historical market data, extracting relevant features, and creating informative datasets for the RL model.

- Model Development and Optimization: Designing and refining an RL algorithm that learns optimal trading strategies, aiming to maximize returns while managing risks effectively.

- Performance Evaluation and Validation: Rigorously testing the developed strategy on historical data to evaluate its performance metrics, including returns, risk-adjusted measures, and comparisons against benchmarks or traditional trading strategies.

- Adaptability and Real-time Testing: Ensuring the model's adaptability to new market conditions and validating its performance in simulated or live trading environments.

- Ethical and Regulatory Compliance: Addressing ethical concerns, ensuring transparency, and compliance with regulatory standards in algorithmic trading practices.

- Significance and Impact: Improved Trading Efficiency: The project aims to develop more efficient, adaptable, and potentially more profitable trading strategies, benefiting investors and financial institutions.

- Technological Advancements: By pushing the boundaries of algorithmic trading with RL, the project contributes to technological innovation in financial markets and fosters advancements in financial technology.

- Academic Contribution: This project aids in expanding knowledge in financial machine learning and reinforces the understanding of RL applications in finance and market dynamics.

- Ethical and Responsible AI Deployment: The project emphasizes ethical considerations, promoting fair and responsible trading practices, and ensuring compliance with regulatory standards in algorithmic trading.`

Executing this project necessitates expertise in finance, machine learning, data analysis, and a commitment to ethical AI deployment. The endeavor aims to develop a sophisticated trading strategy that adapts to market complexities, potentially revolutionizing the way trading decisions are made in stock markets.

SOFTWARE USED:

- Programming Language- Python
- Development environment- Jupyter Notebook, IDEs
- Framework- Open AI Gym, Pytorch, etc.

OVERVIEW OF THE PROBLEM

The primary goal is to create intelligent trading strategies that autonomously decide when to buy, sell, or hold stocks, with the aim of maximizing returns while minimizing risks in the ever-changing and complex stock market environment. This project involves several key steps:

- Data Utilization and Preprocessing: Gathering historical market data such as stock prices, volumes, and various indicators. This data is then processed and refined to extract meaningful patterns and features.

- Reinforcement Learning Framework: Constructing a framework that defines the agent's actions, states, and rewards within the context of trading. This framework allows algorithms to learn optimal strategies through interaction with the market environment.

- Model Development and Training: Building and training reinforcement learning models using historical data. These models learn from past market behavior to make informed decisions about future trades.

- Performance Evaluation: Testing the trained models on historical data to assess their effectiveness. Metrics like returns, risk-adjusted measures, and comparisons against benchmarks are used to evaluate the strategies' performance.

- Real-time Adaptability and Continuous Improvement: Ensuring that the developed strategies can adapt to new market conditions in real-time.

This involves refining and improving the models to better navigate evolving market dynamics.

The ultimate aim of this project is to create robust and adaptive trading strategies that outperform traditional methods. It emphasizes the application of cutting-edge machine learning techniques to make more informed and profitable decisions in the complex realm of stock trading.

DATASET

1	date	open	high	low	close	volume	Name
2	2/8/2013	15.07	15.12	14.63	14.75	8407500	AAL
3	#########	14.89	15.01	14.26	14.46	8882000	AAL
4	#########	14.45	14.51	14.1	14.27	8126000	AAL
5	#########	14.3	14.94	14.25	14.66	10259500	AAL
6	#########	14.94	14.96	13.16	13.99	31879900	AAL
7	#########	13.93	14.61	13.93	14.5	15628000	AAL
8	#########	14.33	14.56	14.08	14.26	11354400	AAL
9	#########	14.17	14.26	13.15	13.33	14725200	AAL
10	#########	13.62	13.95	12.9	13.37	11922100	AAL
11	#########	13.57	13.6	13.21	13.57	6071400	AAL
12	#########	13.6	13.76	13	13.02	7186400	AAL
13	#########	13.14	13.42	12.7	13.26	9419000	AAL
14	#########	13.28	13.62	13.18	13.41	7390500	AAL
15	#########	13.49	13.63	13.39	13.43	6143600	AAL
16	3/1/2013	13.37	13.95	13.32	13.61	7376800	AAL
17	3/4/2013	13.5	14.07	13.47	13.9	8174800	AAL
18	3/5/2013	14.01	14.05	13.71	14.05	7676100	AAL
19	3/6/2013	14.52	14.68	14.25	14.57	13243200	AAL
20	3/7/2013	14.7	14.93	14.5	14.82	9125300	AAL
21	3/8/2013	14.99	15.2	14.84	14.92	10593700	AAL
22	#########	14.85	15.15	14.71	15.13	6961800	AAL
23	#########	15.14	15.6	14.95	15.5	8999100	AAL

51	#########	15.99	16	15.5	15.52	9227100	AAL
52	#########	15.33	16.49	15.33	16.3	12302300	AAL
53	#########	16.26	16.5	16	16.45	6114400	AAL
54	#########	16.55	16.73	16.19	16.22	5548800	AAL
55	#########	16.38	16.73	16.16	16.59	7272100	AAL
56	#########	16.7	16.97	16.56	16.81	5436400	AAL
57	#########	16.8	17.05	16.57	16.9	3640700	AAL
58	5/1/2013	16.91	17.17	16.6	16.6	4943600	AAL
59	5/2/2013	16.72	16.98	16.6	16.94	4888900	AAL
60	5/3/2013	17.02	17.19	16.89	17.02	6451900	AAL
61	5/6/2013	17.05	17.11	16.91	17	3930700	AAL
62	5/7/2013	17.15	17.15	16.95	16.98	3157000	AAL
63	5/8/2013	17.01	17.55	16.99	17.34	6706200	AAL
64	5/9/2013	17.53	17.86	17.34	17.38	6424000	AAL
65	#########	17.61	17.81	17.41	17.76	4248200	AAL
66	#########	17.74	17.95	17.57	17.72	4250900	AAL
67	#########	17.82	18.3	17.8	18.1	5989700	AAL
68	#########	18.37	18.99	18.31	18.81	8951500	AAL
69	#########	18.96	19.52	18.88	19.12	6679300	AAL
70	#########	19.38	19.7	18.75	19.01	7540000	AAL
71	#########	19.05	19.39	18.39	18.59	6055000	AAL
72	#########	18.55	18.7	17.65	17.95	10018700	AAL
73	#########	18.06	18.43	17.71	17.93	8662000	AAL

121	#########	19.25	19.49	19.19	19.35	9571100	AAL
122	8/1/2013	19.44	19.59	19.24	19.38	7989100	AAL
123	8/2/2013	19.38	19.39	18.9	18.92	7382700	AAL
124	8/5/2013	18.9	19.25	18.9	19.17	4365800	AAL
125	8/6/2013	19.17	19.2	18.83	18.92	3350300	AAL
126	8/7/2013	18.84	18.98	18.52	18.87	4318500	AAL
127	8/8/2013	18.99	19.13	18.9	18.98	4269900	AAL
128	8/9/2013	18.91	19.08	18.56	18.6	4720600	AAL
129	#########	18.49	18.88	18.07	18.82	4674800	AAL
130	#########	18.86	18.95	16.29	16.36	78591200	AAL
131	#########	16.05	16.34	15.35	16.17	28026200	AAL
132	#########	15.97	16.16	15.65	15.72	10372800	AAL
133	#########	15.7	16.42	15.7	16.01	8824700	AAL
134	#########	15.97	15.99	15.62	15.64	6141500	AAL
135	#########	15.64	16.21	15.6	15.8	6373900	AAL
136	#########	15.8	15.99	15.75	15.81	4234300	AAL
137	#########	15.91	16.37	15.9	16.27	4770200	AAL
138	#########	16.3	16.4	16.1	16.16	2190400	AAL
139	#########	16.16	16.41	16.03	16.21	2808200	AAL
140	#########	15.87	16	15.33	15.47	6422300	AAL
141	#########	15.38	15.53	15.28	15.34	3613100	AAL
142	#########	16.03	16.39	15.72	15.96	7645000	AAL
143	#########	16.1	16.73	16.07	16.16	7569500	AAL

360744	#########	65.74	67.88	65.65	66.68	4907280	MAR
360745	#########	64.1	65.68	64	64.94	7969291	MAR
360746	#########	64.5	65.81	64.25	65.71	3596820	MAR
360747	#########	66.21	67.53	66.18	66.79	5341629	MAR
360748	#########	66.51	66.86	66.05	66.27	3016301	MAR
360749	#########	65.64	67.26	65.03	67.03	3740715	MAR
360750	#########	67.56	68.33	66.775	68.3	4236249	MAR
360751	#########	68.97	69	67.8	68.15	2961937	MAR
360752	#########	68.33	68.85	67.68	68.15	4900574	MAR
360753	3/1/2016	68.33	69.18	68.28	68.93	3416008	MAR
360754	3/2/2016	68.57	69	68.14	68.83	2733127	MAR
360755	3/3/2016	68.57	69.005	68.06	68.68	2341004	MAR
360756	3/4/2016	68.8	69.28	68.4	68.98	1828883	MAR
360757	3/7/2016	68.51	69.54	68.352	69.54	2742371	MAR
360758	3/8/2016	68.9	69.45	68.18	68.41	4362110	MAR
360759	3/9/2016	68.74	68.77	68.09	68.59	2283898	MAR
360760	#########	68.74	69.145	66.72	67.96	2281313	MAR
360761	#########	68.97	69.17	68.395	68.89	1878081	MAR
360762	#########	71.05	71.3	69.94	70.93	8951395	MAR
360763	#########	70.06	71.03	70.01	70.41	4007640	MAR
360764	#########	69.98	71.69	69.8451	71.24	3382967	MAR
360765	#########	70.98	71.82	70.54	71.8	4546582	MAR
360766	#########	72.61	73.89	72.25	73.16	13530670	MAR

402854	#########	32.19	32.442	31.785	31.89	481035	NCLH
402855	#########	32	32.295	31.98	32.18	392057	NCLH
402856	#########	32.3	32.46	31.84	32.055	621633	NCLH
402857	#########	32.4	32.86	32.28	32.54	660270	NCLH
402858	#########	32.5	32.94	32.04	32.4	1273126	NCLH
402859	#########	32.47	32.96	32.4	32.76	1413289	NCLH
402860	#########	32.1	34.33	32.1	33.535	2615611	NCLH
402861	#########	33.65	33.73	32.915	33.05	1658496	NCLH
402862	#########	32.78	33.08	32.55	32.78	1876311	NCLH
402863	8/1/2014	32.72	33.48	32.5	32.97	1338046	NCLH
402864	8/4/2014	32.96	33.34	32.66	32.96	499445	NCLH
402865	8/5/2014	32.7	33.15	32.25	32.43	980557	NCLH
402866	8/6/2014	32.19	32.555	31.38	31.88	1181313	NCLH
402867	8/7/2014	32.08	32.25	31.66	31.79	603382	NCLH
402868	8/8/2014	31.85	32.015	31.62	31.75	546803	NCLH
402869	#########	31.78	32.21	31.76	32.18	331638	NCLH
402870	#########	32.19	32.68	32.07	32.33	393255	NCLH
402871	#########	32.42	32.73	32.19	32.2	483642	NCLH
402872	#########	32.1	32.84	32.05	32.84	896081	NCLH
402873	#########	33.4	34.5	33.28	33.56	1509858	NCLH
402874	#########	33.57	34.15	33.57	33.68	795671	NCLH
402875	#########	33.86	34	33.445	33.495	619854	NCLH
402876	#########	33.37	33.95	33.21	33.79	679727	NCLH

537606	4/6/2016	71.71	72.47	71.13	72.25	966887	TIF
537607	4/7/2016	71.75	72.42	70.33	70.86	1504710	TIF
537608	4/8/2016	71.26	71.71	70.34	70.63	1386388	TIF
537609	#########	70.83	71.25	70.31	70.83	1719353	TIF
537610	#########	71.05	71.5	69.67	70.24	1876071	TIF
537611	#########	70.88	72.2	70.54	71.8	1612478	TIF
537612	#########	71.6	72	70.66	71.01	1623673	TIF
537613	#########	71.01	71.58	70.87	71.18	1256926	TIF
537614	#########	71.19	71.27	70.63	71.18	1486883	TIF
537615	#########	71.36	72.19	71.32	71.76	1307291	TIF
537616	#########	71.83	72.48	71.2	72.22	1016473	TIF
537617	#########	72.62	73.36	72.13	72.18	1265484	TIF
537618	#########	72.21	73.3	71.76	73.03	1411232	TIF
537619	#########	72.63	73.04	71.72	72.09	1021646	TIF
537620	#########	72.19	72.97	71.7	72.6	1045306	TIF
537621	#########	72.66	73.58	72.35	73.36	838802	TIF
537622	#########	72.69	73	71.92	72.05	943785	TIF
537623	#########	72.09	72.09	70.95	71.35	1438767	TIF
537624	5/2/2016	71.49	72.18	70.71	72.04	1309683	TIF
537625	5/3/2016	71.32	71.59	70.74	71.06	1666131	TIF
537626	5/4/2016	70.54	70.76	69.73	70.47	1537530	TIF
537627	5/5/2016	70.13	70.58	69.42	70.02	1097463	TIF
537628	5/6/2016	69.74	69.9	68.42	69.29	2000157	TIF

619019	1/5/2018	72.83	73.92	72.57	73.36	2166065	ZTS
619020	1/8/2018	73.43	74.42	73.1607	74.24	3631552	ZTS
619021	1/9/2018	74.7	75.475	74.465	75.11	2721946	ZTS
619022	#########	74.58	74.8	73.28	73.91	2257464	ZTS
619023	#########	74.05	74.68	73.72	74.59	1629279	ZTS
619024	#########	74.91	75.69	74.77	75.39	1915669	ZTS
619025	#########	76.06	76.4	75.21	75.54	2599094	ZTS
619026	#########	75.81	77.03	75.39	76.77	2769587	ZTS
619027	#########	76.65	76.65	75.85	76.33	2588995	ZTS
619028	#########	76.69	76.91	76.22	76.62	4829602	ZTS
619029	#########	76.67	77.58	76.62	77.48	4195103	ZTS
619030	#########	77.3	78.32	76.97	77.59	2264132	ZTS
619031	#########	78.16	78.51	77.6	78.33	2585326	ZTS
619032	#########	78.47	79.38	78.345	79.25	2327262	ZTS
619033	#########	79.49	80.13	79.38	80.09	2532808	ZTS
619034	#########	79.81	79.95	79.11	79.18	2662383	ZTS
619035	#########	78.44	78.69	77.91	78.35	3808707	ZTS
619036	#########	78.49	78.77	76.54	76.73	4136360	ZTS
619037	2/1/2018	76.84	78.27	76.69	77.82	2982259	ZTS
619038	2/2/2018	77.53	78.12	76.73	76.78	2595187	ZTS
619039	2/5/2018	76.64	76.92	73.18	73.83	2962031	ZTS
619040	2/6/2018	72.74	74.56	72.13	73.27	4924323	ZTS
619041	2/7/2018	72.7	75	72.69	73.86	4534912	ZTS

AIM

The main emphasis and objective of our project is to analyse given raw data and do exploratory data analysis in order to fully comprehend and identify patterns. Then, using a Neural Network approach, construct a model and train it to get the desired outcomes. Finally, it will be deployed as a web application.

ABSTRACT

We propose to determine a single exchange-traded fund (SPY) investing strategy that will maximize our total wealth. We compare our findings to the tried-and-true “buy-and-hold” and Moving average convergence divergence (MACD) strategies. It’s a Machine Learning model which integrates Data Science and Web Development. We have deployed the app on the Heroku Cloud Application Platform. Here, we intend to base an evaluation on every basic

criterion that is taken into account when establishing the pricing. As mentioned, we intend to forecast future price fluctuations for a specific stock. Prices from previous days, as well as financial media stories connected to the firm of interest, are used to create these forecasts. Reinforcement learning is all about taking the right steps to maximise your reward in a given situation. It is used by a variety of software and computers to determine the best feasible action or path in a given situation. The Reinforcement Machine Learning model is employed in this work to forecast the closure price using past data. We develop a predictor for multiple firms using these trained models that forecasts the every day close stock prices. By providing inputs such as open, high, and low prices, stock volume, and the latest events about each firm, this predictor may be used to determine the price at which the stock value will close for a certain day. The goal of this project is to learn and get hands-on experience in Data Analytics and Machine Learning.

IMPORTANCE OF THE PROJECT

Optimizing stock trading strategies using reinforcement learning holds significant importance across various dimensions:

- Financial Markets Efficiency:
    1. Enhanced Decision-Making: Reinforcement learning allows for adaptive decision-making in complex and dynamic market conditions. It aims to optimize strategies based on changing market data, potentially improving trading outcomes.
    2. Adaptability to Market Changes: RL models can adapt to changing market trends, news, and other factor.
- Investor Returns and Risk Management:

1. Improved Returns: Successful application of RL in trading can potentially lead to better returns, higher profitability, and improved risk-adjusted performance, benefiting investors and traders.
2. Risk Mitigation: RL-based strategies can help in risk management by identifying and managing exposure to market risks more effectively, potentially reducing portfolio volatility and downside risk.

- Technological Advancements:
    1. Advancements in Algorithmic Trading: RL in stock trading contributes to the evolution of algorithmic trading strategies, leveraging machine learning techniques to make more sophisticated and automated trading decisions.
    2. Innovation in Financial Technology (Fintech): Successful implementation of RL in trading can drive innovation in fintech by showcasing the capabilities of machine learning in optimizing financial strategies.
- Research and Development:
    1. Academic and Research Advancements: The intersection of reinforcement learning and finance fosters research in both fields, driving advancements in algorithmic trading, market dynamics analysis, and financial machine learning.
    2. Knowledge Expansion: Projects utilizing RL in stock trading contribute to a deeper understanding of market dynamics, human behavior in financial markets, and the potential for algorithmic decision-making.
- Challenges and Ethical Considerations:
    1. Ethical Challenges: Algorithmic trading, especially when using AI, raises ethical concerns related to market fairness, manipulation, and

the impact on market stability. Projects in this area prompt discussions and considerations about ethical trading practices.

2. Complexity and Interpretability: RL models might be highly complex and challenging to interpret, raising concerns about transparency, accountability, and the potential for unintended consequences in decision-making.

- Economic and Societal Impact:
    1. Economic Implications: Efficient trading strategies can have a broader economic impact by improving capital allocation, liquidity, and market efficiency, potentially benefiting the overall financial ecosystem.
    2. Employment and Industry Development: Advancements in algorithmic trading and fintech can contribute to job creation, expertise development, and the growth of industries related to finance and technology.

In summary, the application of reinforcement learning in optimizing stock trading strategies carries implications for investors, technology advancements, academic research, ethical considerations, economic impact, and societal development, influencing various aspects of financial markets and beyond`

APPLICATION OF THE PROJECT:

The application of optimizing stock trading strategies using reinforcement learning has diverse implications across multiple sectors:

- Finance and Investment:
    i. Algorithmic Trading: Implementing RL in stock trading enables the development of algorithmic trading systems that autonomously execute buy, sell, or hold decisions based on learned patterns and market dynamics.

ii. Portfolio Management: RL models can assist in portfolio optimization by dynamically adjusting asset allocations based on market conditions, risk tolerance, and investor preferences.

iii. Risk Management: Applications extend to risk management, where RL can help in identifying and managing various types of financial risks, including market risk, credit risk, and operational risk.

- Financial Technology (Fintech):
  i. Robo-Advisors: Integrating RL into robo-advisors enhances their capabilities to provide personalized investment advice and automate investment decisions for retail investors. High-Frequency Trading: RL can be employed in high-frequency trading strategies, optimizing trade execution and capturing short-term market inefficiencies more effectively.
  ii. Academic Research and Education: Financial Machine Learning: Projects exploring RL in stock trading contribute to advancements in financial machine learning, providing insights into how AI can be utilized to improve trading strategies.
  iii. Educational Tools: Applications of RL in stock trading serve as educational tools for students and researchers, helping them understand complex market dynamics and reinforcement learning principles.
- Risk Assessment and Prediction: Market Forecasting:
  i. RL models can assist in market forecasting by analyzing historical data and identifying patterns that could help predict future price movements.
  ii. Scenario Analysis: These models can also be used for scenario analysis, helping financial institutions assess the potential impact of various economic scenarios on their portfolios. Regulatory

Compliance and Governance: Market Surveillance: Regulators can use RL models for market surveillance, identifying anomalies or suspicious trading behavior more efficiently.

iii. Compliance Management: Financial institutions can employ RL for compliance management, ensuring adherence to regulatory requirements in trading activities.

- Behavioral Finance and Decision-Making:
  i. Understanding Market Behavior: Projects applying RL in trading contribute to understanding market behaviors and biases, shedding light on the interaction between human behavior and market dynamics.
  ii. Decision Support Systems: RL models can be utilized to develop decision support systems for traders, aiding them in making more informed and data-driven decisions.
  iii. Social Impact: Financial Inclusion: Improved trading strategies through RL can potentially enhance access to better investment opportunities for a wider range of investors, contributing to financial inclusion.
  iv. Market Stability: Well-optimized trading strategies may contribute to market stability by reducing excessive volatility resulting from inefficient trading practices.

The applications of optimizing stock trading strategies using reinforcement learning span across finance, technology, academia, risk management, regulatory compliance, and societal impacts, influencing various aspects of investment, trading, and market behavior.

INTRODUCTION

We've always been captivated by the stock market's seeming unpredictability. There are hundreds of stocks to select from, and day traders can trade almost any of them. So, for a day trader, deciding what to trade is the first and most important step. The following stage is to come up with some ways to benefit from the trading opportunity (one stock, several stocks, exchange-traded funds, ETFs, etc.). Intraday traders use a number of ways to profit from price movements in a particular asset. Day traders should look for equities with plenty of liquidity, moderate to high volatility, and a large number of followers. Isolating the present market trend from any surrounding noise and then capitalising on that trend is the key to finding the appropriate stocks for intraday trading.

This has traditionally been done in conjunction with the trade plan and current events. Various research techniques have been explored to automate this laborious procedure since the emergence of Data Science and Machine Learning. This automated trading method will assist in providing recommendations at the appropriate moment and with more accurate estimates. Mutual funds and hedge funds would benefit greatly from an automated trading approach that maximises profits. The type of profitable returns that may be expected will be accompanied by some risk. It's di cult to come up with a lucrative automated trading technique.

Every human being aspires to make as much money as possible in the stock market. It's critical to devise a well-balanced, low-risk plan that will benefit the majority of individuals. One such technique proposes the use of reinforcement learning agents to generate automated trading strategies based on previous data.

This project was made because we were intrigued and we wanted to gain hands-on experience with the Machine Learning Project.

REINFORCEMENT LEARNING

Machine learning models are taught to make a series of judgments via reinforcement learning. In an unpredictable and potentially complex environment, the agent must learn to attain a goal. Artificial intelligence is put in a game-like environment in reinforcement learning. To find a solution to the problem, the computer uses trial and error. Artificial intelligence is given either rewards or penalties for the acts it takes in order to get it to accomplish what the programmer desires. Its purpose is to increase the total prize as much as possible.

Despite the fact that the designer establishes the reward policy–that is, the game's rules–he provides the model with no clues or ideas for how to solve the game.

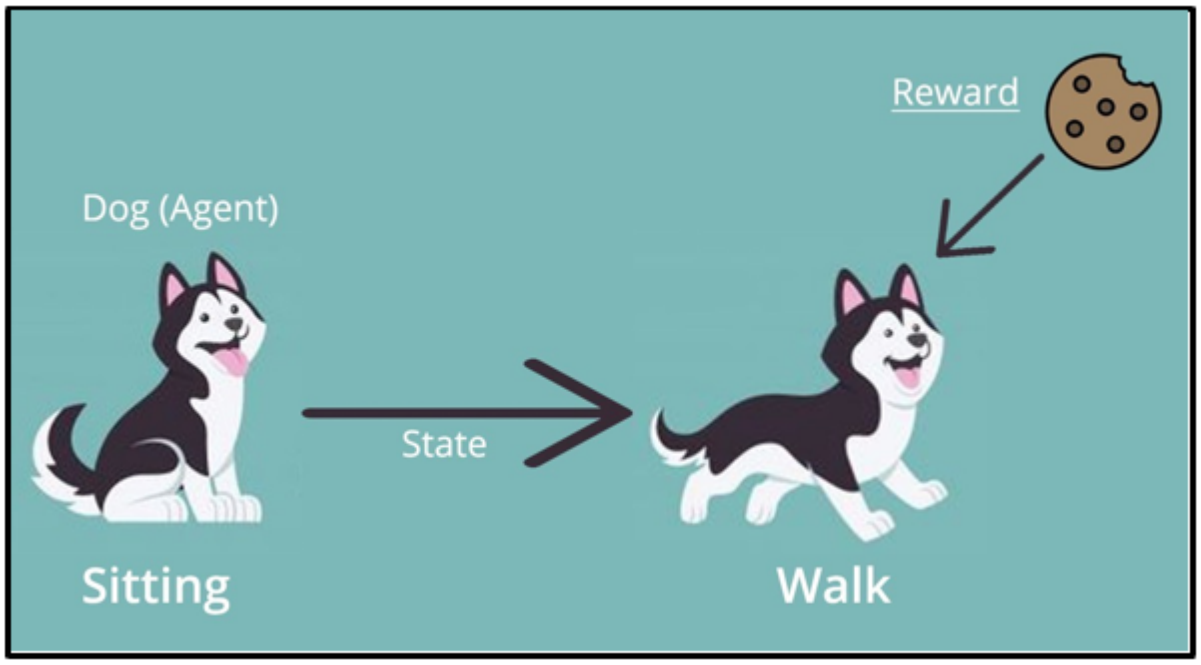
Starting with completely random trials and progressing to sophisticated tactics and superhuman skills, it's up to the model to find out how to do the task in order to maximise the reward.

In reinforcement learning, developers use a framework that rewards desired actions while penalising negative ones. To motivate the agent, this strategy assigns positive values to desired acts and negative values to undesirable

behaviours. To obtain an ideal solution, the agent is programmed to seek long-term and greatest overall return.

These long-term objectives keep the agent from stagnating on smaller objectives. The agent eventually learns to ignore the unpleasant and focus on the good. This technique of learning has been used in artificial intelligence (AI) to drive unsupervised machine learning using incentives and punishments.

In simple terms, You let the dog do whatever it wants and then whenever it behaves well you go, “Good dog” and when it misbehaves you go “Bad Dog!” and then over time the dog learns to do more of the good dog things and fewer of the bad dog things.



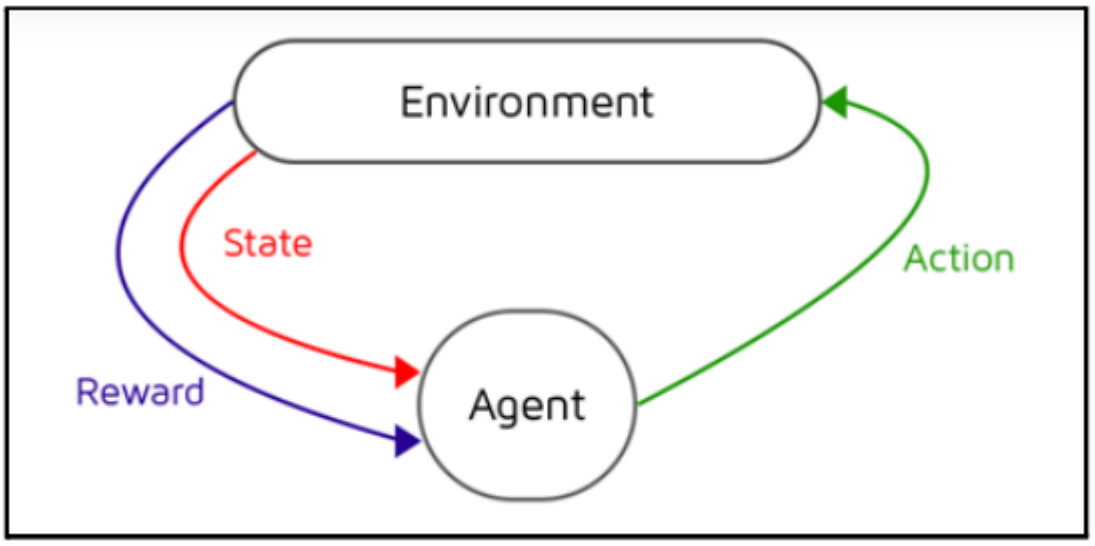
Trading is a never-ending process with no conclusion in sight. Because we don't have comprehensive information about the traders in the market, trading is likewise a stochastic Markov Decision Process. We employ model-free

reinforcement learning, also known as Q-Learning because we don't know the reward function or transition probability.

Markov Decision Process (MDP):

It is a mathematical framework used for modelling decision-making problems where the outcomes are partly random and partly controllable.

A model of anticipating outcomes is the Markov decision process. The model, like a Markov chain, tries to predict a result based solely on the present state's information. The Markov decision process, on the other hand, takes into account the features of actions and motivations. The decision-maker may choose an action accessible in the current state at each phase of the process, causing the model to advance to the next step and rewarding the decision-maker. At each step during the process, the decision-maker may choose to take any action available in the current state, resulting in the model moving to the next step and offering the decision-maker a reward.



An optimization problem may be given to a machine learning algorithm. The programme will use reinforcement learning to try to optimise the behaviours made inside a given environment in order to maximise the potential reward. Reinforcement learning employs Markov decision processes to establish an ideal balance of exploration and exploitation, whereas supervised learning approaches need accurate input/output pairings to construct a model. When the probability and rewards of a result are undefined or unknown, machine learning may employ reinforcement learning through the Markov decision process.

Terminologies:

1. Agent - An RL agent is a machine that we're teaching to make good judgments. For instance, a robot is being taught how to navigate a house without colliding. 2. Environment - The agent's environment is the setting in which the agent interacts. For example, the house where the Robot moves. The agent has no influence over the surroundings; all it has is control over its own actions. For instance, Although the Robot cannot control where a table is kept in the house, it can walk around it to avoid colliding with it.

3. State - The state defines the current situation of the agent, for example, It may be the Robot's specific position in the house, the alignment of its two legs, or its current posture, depending on how you approach the issue.

4. Action - The decision made by the agent in the current time step. For instance: it can move its right or left leg, or raise its arm, or lift an object, turn right or left, etc. We know ahead of time what actions or decisions the agent can take.

5. Policy - A policy is the thinking process that goes into deciding on a course of action. It is a probability distribution applied to the collection of activities in practice. Actions that are very rewarding will have a high likelihood, and vice versa.

Note: Even if an action has a low probability, that does not mean it will not be chosen. It's only that it has a lower chance of being chosen.

Q-LEARNING

The ‘Q in Q-learning stands for quality. Quality in this case represents how useful a given action is in gaining some future reward. Q-learning is an off-policy reinforcement learning algorithm that seeks to find the best action to take given the current state. It’s considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions, and therefore a policy isn’t needed. More specifically, q-learning seeks to learn a policy that maximizes the total reward.

An agent interacts with the environment in one of two ways – exploit or explore – and a Q-table is produced with the dimensions. An exploit option implies that all options are evaluated and the one with the greatest environmental benefit is chosen. An explore option is one that considers a random action without regard for the maximum future payoff.

POINTS TO CONSIDER

1. Gamification of trading - Historical & Current Prices, Technical Data, Buy/Sell/Do Nothing, Profit & Loss.

2. Train the system - Each entry and exit is an individual game, Run through the price series sequentially and randomly.

3. Reward function design - Pure Profit & Loss on exit, otherwise zero, PnL from the start of trade to every time step t.

4. Features to use - Open, High, Low, Close, Volume (OHLCV), Technical indicators, Time of day, Day of Week, Time of year, Different time granularity.

5. Test the system - Sine waves, Trend curves, Random walks, Adding Noise to clean test curves.

READING DATASET:

```
[ ] import csv
    import pandas as pd
```

```
[ ] df=pd.read_csv("/content/all_stocks_5yr.csv")
    print(df)
```

```
              date   open   high    low  close    volume Name
0       2013-02-08  15.07  15.12  14.63  14.75   8407500  AAL
1       2013-02-11  14.89  15.01  14.26  14.46   8882000  AAL
2       2013-02-12  14.45  14.51  14.10  14.27   8126000  AAL
3       2013-02-13  14.30  14.94  14.25  14.66  10259500  AAL
4       2013-02-14  14.94  14.96  13.16  13.99  31879900  AAL
...            ...    ...    ...    ...    ...       ...  ...
619035  2018-02-01  76.84  78.27  76.69  77.82   2982259  ZTS
619036  2018-02-02  77.53  78.12  76.73  76.78   2595187  ZTS
619037  2018-02-05  76.64  76.92  73.18  73.83   2962031  ZTS
619038  2018-02-06  72.74  74.56  72.13  73.27   4924323  ZTS
619039  2018-02-07  72.70  75.00  72.69  73.86   4534912  ZTS

[619040 rows x 7 columns]
```

```
[ ] import pickle
```

```
df=pd.read_pickle("/content/pickl.pkl")
print (df)
```

```
[[[-5.00000000e+01 -3.25700000e-02 -3.06450000e-02]
  [ 4.50175744e-02 -9.99965051e+02 -7.54361010e-01]]

 [[-5.00000000e+01  1.46115863e-01  0.00000000e+00]
  [ 1.11588379e-01 -9.99893150e+02  7.69867164e-01]]]
```

STEPS TO CREATE AND DEPLOY MODEL

1. Import necessary Libraries.
2. Load Dataset.
3. Perform Exploratory Data Analysis.
4. Create An Environment.
5. Prepare Data.

6. Train Data.
7. Test Data.
8. Evaluate Model.

IMPORT NECESSARY LIBRARIES

```
import time
import copy
import numpy as np
import pandas as pd
import chainer
import chainer.functions as F
import chainer.links as L
from plotly import tools, subplots
from plotly.graph_objs import *
from plotly.offline import init_notebook_mode, iplot, iplot_mpl
init_notebook_mode()
```

LOAD DATASET

Data on stock market values from 2013 to 2018 is included in this dataset. This dataset contains the following information: date, open, high, low, and close values, as well as volume and stock names. Also, the dataset is in CSV format.

```
data = pd.read_csv('../input/stock-prices/all_stocks_5yr.csv')
```

The following is a quick description of each component in the dataset:

1. DATE - The day on which stock is exchanged.
2. OPEN - Any listed stock's daily opening price is the price at which it is initially traded.
3. HIGH - The maximum price at which a stock can be bought or sold during a trading day is known as the high.
4. LOW - The minimum price at which a stock can be bought or sold during a trading day is known as the low.
5. CLOSE - During a standard trading session, the last price at which a stock trades is referred to as the closing price.
6. VOLUME - The total number of shares or contracts exchanged in a securities or market within a certain time period.
7. NAME - The name of the stock.

.

In [4]:
```
data.head()
```

Out[4]:

	date	open	high	low	close	volume	Name
0	2013-02-08	15.07	15.12	14.63	14.75	8407500	AAL
1	2013-02-11	14.89	15.01	14.26	14.46	8882000	AAL
2	2013-02-12	14.45	14.51	14.10	14.27	8126000	AAL
3	2013-02-13	14.30	14.94	14.25	14.66	10259500	AAL
4	2013-02-14	14.94	14.96	13.16	13.99	31879900	AAL

In [5]:
```
data.tail()
```

Out[5]:

	date	open	high	low	close	volume	Name
619035	2018-02-01	76.84	78.27	76.69	77.82	2982259	ZTS
619036	2018-02-02	77.53	78.12	76.73	76.78	2595187	ZTS
619037	2018-02-05	76.64	76.92	73.18	73.83	2962031	ZTS
619038	2018-02-06	72.74	74.56	72.13	73.27	4924323	ZTS
619039	2018-02-07	72.70	75.00	72.69	73.86	4534912	ZTS

EXPLORATORY DATA ANALYSIS

In every Data Analysis or Data Science project, exploratory data analysis, or EDA, is a critical stage. EDA is the investigation of a dataset to find patterns and anomalies as well as to generate hypotheses based on our knowledge of the information.

During the course of a data science or machine learning project, EDA consumes around half of the time spent on data analysis, feature selection, feature engineering, and other processes. Because it is the most essential element or backbone of a data science project, where one has to perform several tasks like data cleaning, dealing with missing values, handling outliers, treating unbalanced datasets, handling categorical features, and many others. To automate our tasks in exploratory data analysis, we may utilise python packages like dtale, pandas profiling, sweetviz, and autoviz.

```
import sweetviz as sv

advert_report = sv.analyze(df)

advert_report.show_html('EDA.html')
```

```
Done! Use 'show' commands to display/save. [100%] 00:00 -> (00:00 left)

Report EDA.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, re
gardless, the report IS saved in your notebook/colab files.
```

Find and resolve missing values in the dataset.

```
df.isnull().sum()
```

```
date      0
open     11
high      8
low       8
close     0
volume    0
Name      0
dtype: int64
```

```
df.dropna(inplace=True)
```

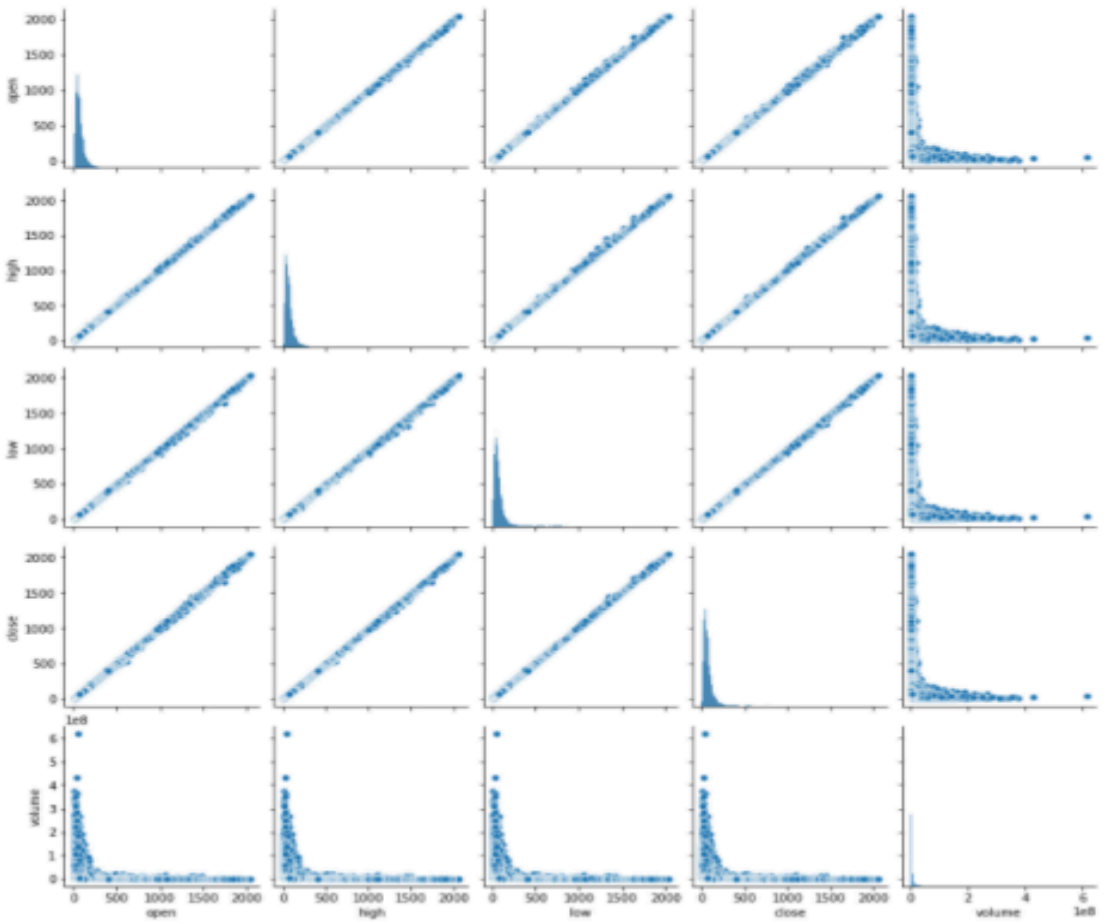
```
df.isnull().sum()
```

```
date      0
open      0
high      0
low       0
close     0
volume    0
Name      0
dtype: int64
```

Use a pairplot to visualise data and find patterns and relationships.

```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x7ff4939743d0>
```



To conduct calculations for better analysis, use the Groupby() method to break the data into distinct groups.

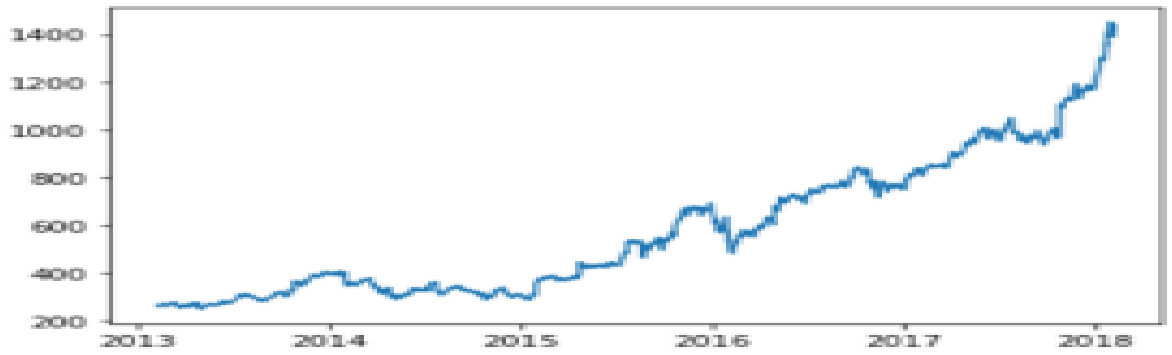
```
df.groupby('Name').mean()
```

Name	open	high	low	close	volume
A	49.187863	49.600059	48.782026	49.202025	2.338039e+06
AAL	38.390495	38.955554	37.825605	38.393252	9.390321e+06
AAP	132.439631	133.818297	131.036025	132.433463	1.078043e+06
AAPL	109.055429	109.951118	108.141589	109.066698	5.404790e+07
ABBV	60.802801	61.474133	60.177275	60.864440	7.870683e+06
...	...	...	...	...	...
XYL	41.415473	41.763885	41.076026	41.434095	1.183141e+06
YUM	75.422099	76.027123	74.844914	75.451009	3.209032e+06
ZBH	105.542014	106.419854	104.698038	105.606291	1.297144e+06
ZION	32.161477	32.509290	31.802241	32.171790	2.621178e+06
ZTS	45.091389	45.488826	44.665588	45.098648	3.681878e+06

505 rows x 5 columns

```
def stocks(name):
    a = df[df['Name']==name]
    a = a.set_index('date')
    plt.plot(a['close'])

stocks('AMZN')
```



Detecting and removing outliers from the dataset. Z-Score Method to detect outliers.

```
import numpy as np
outliers = []
def detect_outliers_zscore(df):
    thres = 3
    mean = np.mean(df)
    std = np.std(df)
    # print(mean, std)
    for i in df:
        z_score = (i-mean)/std
        if (np.abs(z_score) > thres):
            outliers.append(i)
    return outliers
sample_outliers = detect_outliers_zscore(df['close'])
print("Outliers from Z-scores method: ", sample_outliers)
```

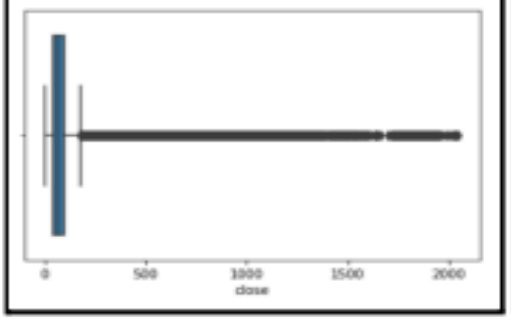
```
Outliers from Z-scores method:  [376.64, 381.37, 386.71, 393.62, 392.3, 384.66, 385.96, 384.
49, 386.95, 384.89, 387.78, 382.19, 381.25, 384.24, 388.97, 387.65, 395.96, 395.19, 402.2, 4
02.92, 399.2, 404.39, 398.08, 393.37, 398.79, 397.97, 396.44, 393.63, 398.03, 401.92, 401.0
1, 397.66, 390.98, 397.54, 395.87, 395.8, 399.61, 407.05, 404.54, 399.87, 387.6, 386.28, 39
4.43, 384.2, 403.01, 378.77, 377.17, 381.83, 375.43, 378.995, 383.66, 380.14, 378.59, 385.3
7, 384.8, 380.16, 385.655, 384.61, 382.72, 387.83, 380.09, 378.56, 378.49, 377.04, 381.2, 38
3.54, 382.65, 382.36, 385.11, 383.45, 386.04, 375.56, 389.51, 391.18, 389.8, 389.99, 445.1,
438.56, 429.31, 429.37, 421.78, 422.87, 423.04, 421.19, 419.1, 426.88, 433.69, 432.85, 431.6
2, 426.87, 432.28, 426.0, 425.24, 421.71, 423.86, 431.63, 427.63, 425.47, 431.42, 426.57, 42
9.23, 430.92, 430.99, 436.59, 430.78, 426.95, 423.5, 425.48, 430.77, 432.97, 429.92, 423.67,
427.26, 427.81, 439.39, 434.92, 436.29, 445.99, 440.84, 440.1, 438.1, 429.86, 434.09, 437.3
9, 437.71, 436.04, 436.72, 429.7, 434.39, 443.51, 455.57, 465.57, 461.19, 475.48, 483.01, 48
8.1, 488.0, 488.27, 482.18, 529.42, 531.41, 526.03, 529.0, 536.76, 536.15, 535.03, 531.9, 53
7.01, 529.46, 522.62, 524.0, 527.46, 525.91, 529.66, 531.52, 535.22, 535.02, 532.92, 515.78,
494.47, 463.37, 466.37, 500.77, 518.37, 518.01, 512.89, 496.54, 510.55, 504.72, 499.0, 517.5
4, 516.89, 522.24, 529.44, 521.38, 522.37, 527.39, 538.87, 540.26, 548.39, 538.4, 536.07, 53
3.75, 524.25, 504.06, 496.07, 511.89, 520.72, 532.54, 543.68, 537.48, 541.94, 533.16, 539.8,
550.19, 548.9, 544.83, 562.44, 570.76, 573.15, 560.88, 555.77, 563.91, 599.03, 608.61, 611.6
1, 617.1, 626.55, 625.9, 628.35, 625.31, 640.95, 655.65, 659.37, 655.49, 659.68, 673.25, 66
5.6, 642.35, 647.81, 643.3, 663.54, 661.27, 668.45, 678.99, 671.15, 675.34, 673.26, 664.8, 6
79.06, 676.01, 666.25, 672.64, 669.83, 677.33, 664.79, 662.32, 640.15, 657.91, 658.64, 675.7
7, 670.65, 664.14, 664.51, 663.15, 663.7, 662.79, 675.2, 693.97, 689.07, 675.89, 636.99, 63
3.79, 632.65, 607.94, 607.05, 617.74, 617.89, 581.81, 593.0, 570.18, 574.48, 571.77, 575.02,
596.38, 596.53, 601.25, 583.35, 635.35, 587.0, 574.81, 552.1, 531.07, 536.26, 502.13, 488.1,
482.07, 490.48, 503.82, 507.08, 521.1, 534.1, 525.0, 534.9, 559.5, 552.94, 554.04, 555.15, 5
55.23, 552.52, 579.04, 580.21, 577.49, 575.14, 562.8, 560.26, 559.47, 558.93, 569.61, 573.3
7, 577.02, 574.27, 559.44, 552.08, 553.98, 560.48, 569.63, 582.95, 579.87, 593.86, 598.69, 5
93.64, 598.5, 593.19, 586.14, 602.08, 591.43, 594.6, 595.93, 603.17, 614.82, 620.75, 625.89,
```

Use Boxplot and IQR (Inter-Quartile Range) for anomaly detection.

- Close

```
sns.boxplot(df['close'])
```

```
<AxesSubplot:xlabel='close'>
```




```
Q1 = df['close'].quantile(0.45)
Q3 = df['close'].quantile(0.55)
print(Q1,Q3)
iqr=Q3-Q1
iqr
```

```
57.47999999999999 67.71

10.230000000000004
```

```
upper = Q3+1.5*iqr
lower = Q1+1.5*iqr
print(upper,lower)
```

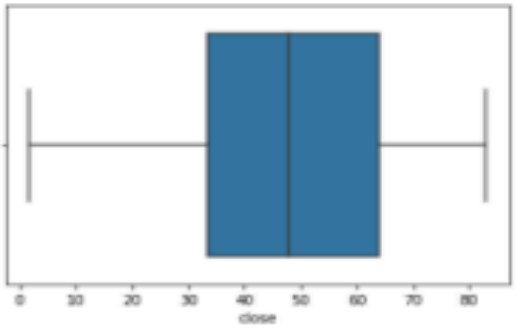
```
83.055 72.82499999999999
```

```
df1 = df[df['close']<upper]
df1.shape
```

```
(423200, 7)
```

```
sns.boxplot(df1['close'])
```

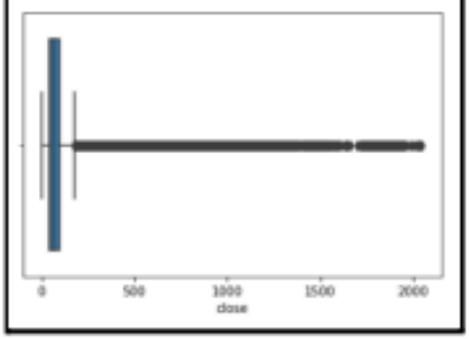
```
<AxesSubplot:xlabel='close'>
```



- Open

```
sns.boxplot(df['close'])
```

```
<AxesSubplot:xlabel='close'>
```




```
Q1 = df['close'].quantile(0.45)
Q3 = df['close'].quantile(0.55)
print(Q1,Q3)
iqr=Q3-Q1
iqr
```

```
57.47999999999999 67.71

10.230000000000004
```

```
upper = Q3+1.5*iqr
lower = Q1+1.5*iqr
print(upper,lower)
```

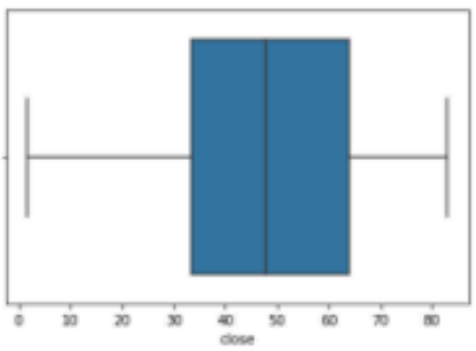
```
83.055 72.82499999999999
```

```
df1 = df[df['close']<upper]
df1.shape
```

```
(423200, 7)
```

```
sns.boxplot(df1['close'])
```

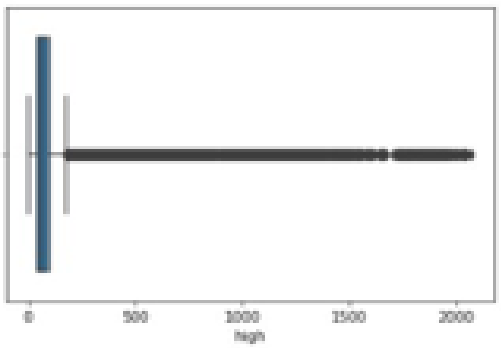
```
<AxesSubplot:xlabel='close'>
```



High

```
sns.boxplot(df['high'])
```

```
<AxesSubplot:xlabel='high'>
```




```
Q1 = df['high'].quantile(0.35)
Q3 = df['high'].quantile(0.65)
print(Q1,Q3)
iqr=Q3-Q1
iqr
```

```
48.89999999999999 79.47

30.570000000000007
```

```
upper = Q3+1.5*iqr
lower = Q1+1.5*iqr
print(upper,lower)
```

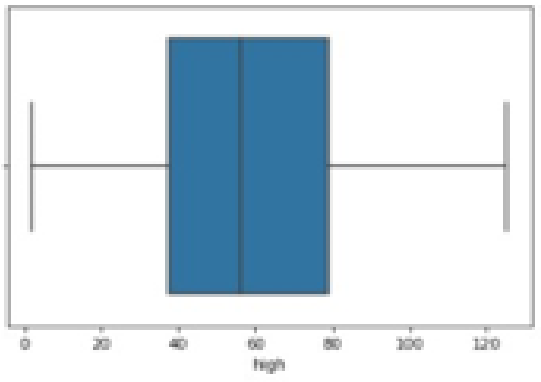
```
125.32500000000002 94.755
```

```
df1 = df[df['high']<upper]
df1.shape
```

```
(533191, 7)
```

```
sns.boxplot(df1['high'])
```

```
<AxesSubplot:xlabel='high'>
```



Low

```
Q1 = df['low'].quantile(0.35)
Q3 = df['low'].quantile(0.65)
print(Q1,Q3)
iqr=Q3-Q1
iqr
```

```
47.98 78.1

30.119999999999997
```

```
upper = Q3+1.5*iqr
lower = Q1+1.5*iqr
print(upper,lower)
```

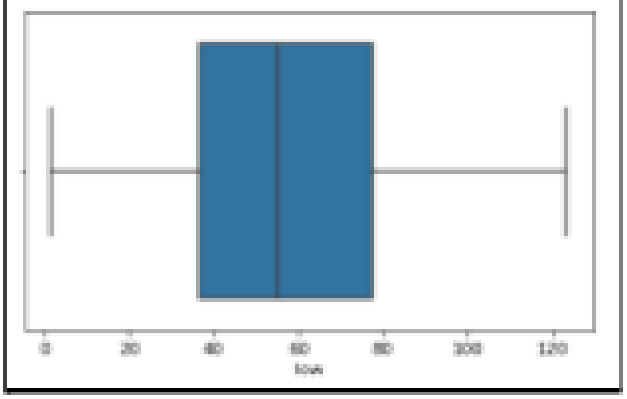
```
123.27999999999999 93.16
```

```
df1 = df[df['low']<upper]
df1.shape
```

```
(533387, 7)
```

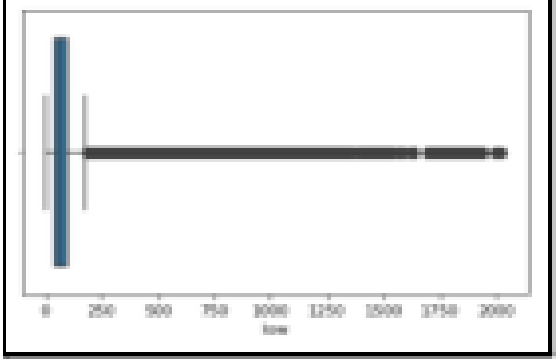
```
sns.boxplot(df1['low'])
```

```
<AxesSubplot:xlabel='low'>
```



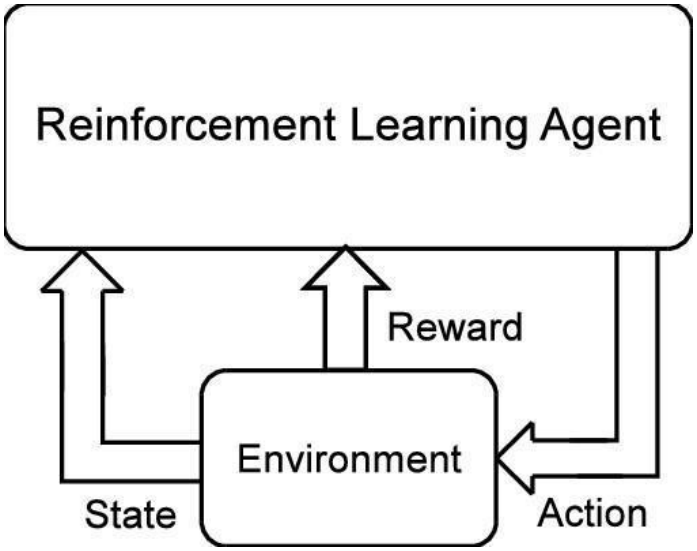

```
sns.boxplot(df['low'])
```

```
<AxesSubplot:xlabel='low'>
```



CREATE AN ENVIRONMENT

In the reinforcement learning problem, the environment is a critical component. It's critical to have a solid grasp of the underlying world with which the RL agent will interact. This aids us in developing the best design and learning strategy for the agent.



The reinforcement learning task is intended to be a simple framing of the challenge of learning from interaction in order to accomplish a goal. The agent is the learner and decision-maker. The environment is the object it interacts with, and it consists of everything outside the agent. The agent chooses actions, and the environment reacts to those actions, presenting the agent with new scenarios. The environment also produces rewards, which are unique numerical values that the agent attempts to maximise over time.

```
class Environment1:

    def __init__(self, df, history_t=90):
        self.data = df
        self.history_t = history_t
        self.reset()

    def reset(self):
        self.t = 0
        self.done = False
        self.profits = 0
        self.positions = []
        self.position_value = 0
        self.history = [0 for _ in range(self.history_t)]
        return [self.position_value] + self.history # obs

    def step(self, act):
        reward = 0

        # act = 0: stay, 1: buy, 2: sell
        if act == 1:
            self.positions.append(self.data.iloc[self.t, :]['close'])
        elif act == 2: # sell
            if len(self.positions) == 0:
                reward = -1
            else:
                profits = 0
                for p in self.positions:
                    profits += (self.data.iloc[self.t, :]['close'] - p)
                reward += profits
                self.profits += profits
                self.positions = []

        # set next time
        self.t += 1
        self.position_value = 0
        for p in self.positions:
            self.position_value += (self.data.iloc[self.t, :]['close'] - p)
        self.history.pop(0)
        self.history.append(self.data.iloc[self.t, :]['close'] - self.data.iloc[(self.t-1), :]
['close'])

        # clipping reward
        if reward > 0:
            reward = 1
        elif reward < 0:
            reward = -1

        return [self.position_value] + self.history, reward, self.done # obs, reward, done
```

```
env = Environment1(train)
print(env.reset())
for _ in range(3):
    pact = np.random.randint(3)
    print(env.step(pact))
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13.52
9999999999998], -1, False)
([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 13.52999
9999999998, 7.380000000000003], -1, False)
([3.8999999999999986, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 13.529999999999998, 7.380000000000003, 3.8999999999999986], 0, False)
```

```
Q, total_losses, total_rewards = train_dqn(Environment1(train))
```

```
/opt/conda/lib/python3.7/site-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning:

Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuple
s-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you
must specify 'dtype=object' when creating the ndarray
```

5	0.0999999999999992	4715	-36.4	1269.1880589734762	58.75414991378784
10	0.0999999999999992	9430	68.8	106.02257219804451	62.61797070503235
15	0.0999999999999992	14145	92.4	88.00889271469786	63.70128417015076
20	0.0999999999999992	18860	105.2	81.62211509905755	64.25300574302673
25	0.0999999999999992	23575	130.8	67.72250651605427	65.3246157169342
30	0.0999999999999992	28290	128.8	64.84192206077277	64.26934337615967
35	0.0999999999999992	33005	109.2	37.99149908022955	65.57610964775085
40	0.0999999999999992	37720	102.0	30.748561142757534	66.24323463439941
45	0.0999999999999992	42435	100.0	30.852891511656345	67.29631352424622
50	0.0999999999999992	47150	132.0	39.443355267494915	65.53461694717407
55	0.0999999999999992	51865	137.6	38.289829121623185	65.3227026462555
60	0.0999999999999992	56580	161.4	42.372054285090414	64.54252576828003

```
plot_train_test_by_q(Environment1(train), Environment1(test), Q, 'DQN')
```

DQN: train s-reward 229, profits 7800, test s-reward 57, profits 2399

MODEL TESTING & TRAINING:

```
import pandas as pd
import numpy as np
import pickle as pk

# Load the Q-table
q_table = pk.load(open("pickl.pkl", 'rb'))

# Function to simulate trading using the trained Q-table on the test data
def test_trading(stocks_test, q_table):
    port_value = 1000
    num_stocks = 0
    buy_history = 0
    net_worth = [1000]

    for dt in range(len(stocks_test)):
        long_ma = stocks_test.iloc[dt]['5day_MA']
        short_ma = stocks_test.iloc[dt]['1day_MA']
        close_price = stocks_test.iloc[dt]['close']
        next_close = 0

        if dt > 0:
            past_close = stocks_test.iloc[dt - 1]['close']
        else:
            past_close = close_price
        t = trade_t(num_stocks, net_worth[-1], close_price)
        state = get_state(long_ma, short_ma, t)
        action = np.argmax(q_table[state])
```

```
        if action == 0:  # Buy
            num_stocks += 1
            buy_history = close_price
            to_append = net_worth[-1] - close_price
            net_worth.append(np.round(to_append, 1))

        elif action == 1:  # Sell
            if num_stocks > 0:
                num_stocks -= 1
                to_append = net_worth[-1] + close_price
                net_worth.append(np.round(to_append, 1))

        elif action == 2:  # Hold
            to_append = net_worth[-1] + close_price
            net_worth.append(np.round(to_append, 1))

    return net_worth

# Apply the testing model on the test dataset
final_net_worth = test_trading(stocks_test, q_table)
print (final_net_worth)
```

TRAINING MODEL EVALUATION

Model evaluation is a step in the model creation process that is often overlooked. This is the stage where the model's performance is determined. As a result, it's important to think about the model's results in terms of every conceivable assessment technique. Using various approaches can result in a variety of viewpoints.

- Training Error: This measures how well a model fits the training data. It's computed by evaluating the model's performance on the same data it was trained on. However, it's important to note that a low training error doesn't guarantee good performance on unseen data.
- Overfitting vs. Underfitting: These are common issues in machine learning. Overfitting occurs when a model learns noise and patterns specific to the training data but doesn’t generalize well to new data. Underfitting, on the other.

TEST MODEL EVALUATION:

- Test Error: This is the performance of the model on new, unseen data (the test set). It's essential for understanding how well the model generalizes to real-world scenarios.
- Validation Set: Sometimes, in addition to the training and test sets, a validation set is used to fine-tune model hyperparameters. The model is repeatedly trained on the training set and evaluated on the validation set to select the best hyperparameters. Then, the model's final perform.

KEY COSIDERATION:

- Data Quality: The quality and representativeness of the data play a significant role in model evaluation.
- Model Complexity: Balancing model complexity to avoid both underfitting and overfitting is crucial.
- Generalization: The ultimate goal is to have a model that generalizes well to new, unseen data. By evaluating models on both training and test sets, practitioners gain insights into the model's performance characteristics and its ability to generalize beyond the training data.

PATHWAY FOR FUTURE BETTERMENT

o Data Enhancement:

i. Alternative Data Sources: Explore additional data streams beyond traditional market data (news sentiment, social media sentiment, economic indicators) to capture more comprehensive market signals.

ii. Feature Engineering: Experiment with new features or combinations of existing ones that could better represent market dynamics and patterns.

- Model Development and Optimization:
  i. Advanced RL Algorithms: Experiment with state-of-the-art RL algorithms or novel variations that might improve learning efficiency or model performance.
  ii. Ensemble Methods: Combine multiple RL models or strategies to create an ensemble approach that leverages the strengths of each individual model.
  iii. Hyperparameter Tuning: Fine-tune model hyperparameters systematically to optimize performance. Techniques like grid search or Bayesian optimization can help in this aspect.

- Risk Management and Control:
  i. Dynamic Risk Control: Implement more sophisticated risk management techniques within the RL framework to dynamically adjust risk exposure based on changing market conditions.
  ii. Portfolio Diversification Strategies: Introduce methods to optimize portfolio diversification to reduce risk and enhance stability.

- Adaptive Learning:
  i. Transfer Learning: Explore the applicability of transfer learning techniques to leverage knowledge from one market or asset class to another.
  ii. Continuous Learning: Implement mechanisms for continuous learning, allowing the model to adapt to changing market regimes and evolving patterns.

- Evaluation and Validation:
    i. Extended Backtesting: Extend backtesting to cover more diverse market conditions, validate the strategy's robustness across different market scenarios, and reduce overfitting.
    ii. Real-time Simulation: Create more realistic real-time simulation environments that closely mimic live market conditions for better performance evaluation.

- Ethical and Regulatory Considerations:
    i. Fairness and Bias Mitigation: Ensure algorithms are fair and unbiased in decision-making to comply with ethical standards.
    ii. Interpretability and Transparency: Enhance model interpretability to understand and explain the rationale behind trading decisions.

- Incorporating Human Expertise:
    i. Human-in-the-Loop Approaches: Integrate human expertise into the learning loop to guide and improve the model's decision-making process.
    ii. Feedback Mechanisms: Establish mechanisms for receiving feedback from traders or domain experts to refine and adjust the model's strategies.

- External Factors:
    i. Geopolitical Analysis: Consider geopolitical events and global economic factors that could impact markets and incorporate this analysis into the model.
    ii. Market Microstructure Analysis: Dive deeper into market microstructure to capture and exploit short-term inefficiencies.