University of BRISTOL

DEPARTMENT OF ENGINEERING MATHEMATICS

Analysis of XAI algorithms for Black-Box Reinforcement Learners

Xu He 2029883

Supervisor: Pro.Jonathan Lawry

A project plan submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Saturday 11th September, 2021

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author. It is important to note that the first two chapters of this thesis are based on the author's own previously submitted project plan. This is a re-use of author's own words from the project plan. It is permitted. Please ignore these statements if they are marked in the duplicating-word checking session.

Xu He 2029883 Saturday 11th September, 2021

Contents

List of Figures

List of Tables

List of Algorithms

Abstract

Nowadays, Explainable artificial intelligence (XAI) is shown to bring greater transparency as well as interpretability to dynamic systems, which plays a non-negligible positive role for the further application of AI techniques in the future. Under this background, I analysed the XAI algorithms of the Black Box Reinforcement Learner and illustrated the differences between the algorithms. In my analysis, I used the dynamic simulation system provided in Open AI Gym as a black box environment to demonstrate the explanatory effects of different XAI algorithms. The improved Frozen Lake environment and the Lunar Lander environment were chosen as the experimental platforms. These 2D environments can simulate dynamic systems well. From previous experiments and related literature, I found that XAI algorithms could make the operation of black box reinforcement learners more transparent and understandable. These algorithms could explain the black box policy to some extent. In conjunction with these literature, I expected to use the rule-based XAI algorithms, Greedy Rule List, CN2 algorithm and Rulefit algorithm to achieve interpretability of black box reinforcement learners in these simulated environments. By comparing the results of different algorithms, the algorithms that could provide the best interpretability in each environment were found. From the results, I concluded that the more complex environment would bring more difficult task of the XAI algorithm and the worse final results. A efficient XAI algorithm needed to balance both interpretability and model performance maintenance. After deriving the results for each of the different algorithms, I found that the Greedy Rule List, by its greedy strategy, did not provide a clear explanation in either environment. Rulefit algorithm worked best in a simple environment with few features. However, the CN2 algorithm, which could handle multiple feature relationships, performed best when the variety and number of features increases. By analysing the results of the different algorithms, I had made conjectures about the interpretability and performance of the various algorithms that appear in this situation. By summarising the shortcomings and the problems that occurred throughout the project, I presented further research plans and ideas for future improvements, such as improving the accuracy of existing models and dealing with more continuous dynamic systems with more features in more dimensions. Finally, I concluded the whole project by presenting the various take-aways from the project.

Keywords: Explainable AI; XAI. Dynamic Systems; Reinforcement Learning;Black Box Policy. Rule Learning. Simulation; Open AI; Frozen Lake; Lunar Lander.

Acknowledge

Firstly, I would like to thank Professor Jonathan Lawry for all his help during the project, he always gave me fruitful advice when I needed it most. He guided me step by step through the final project. I was also able to complete the project thanks to Professor Lawry's friendly and enthusiastic help and timely advice.

Secondly I need to thank Professor Dave Cliff for designing and guiding the whole process of the project. In the first year of the Data Science program, the final project designed by Professor Dave Cliff were very useful and positive for my own improvement. I believe that Professor Cliff's control of the timeline for my final project further motivated me to complete my project successfully and in a timely manner.

Finally, I would like to thank everyone who helped me during the project, including my family, friends and school staff. Without their help, I would not have been able to complete the project. Their helps have been invaluable during this difficult year and have made it all the more touching.

Notation and Acronyms

There are a number of symbols and abbreviations used in this thesis. All the abbreviations and symbols are grouped as follows:

AI	:	Artificial Intelligence
AQ learning	:	Algorithm quasi-optimnal learning
CNN	:	Convolutional Neural Network
DDPG	:	Deep Deterministic Policy Gradient
DQN	:	Deep Q Network
Foil	:	First-Order Inductive Learner
Lasso model	:	L1 regularised linear model
MDLP	:	Minimum Description Length Principle
RMSE	:	Root Mean Square Error
RL	:	Reinforcement Learning
XAI	:	Explainable Artificial Intelligence

Code storage Link and Oral Link

All relevant code and datasets used in this project were stored in a private Github repository. The link to the private Github repository was as follows: DSPR-2029883.

Because of the repository is private, the personal account and corresponding password should be provided: Account: XuHeyr20678 Password: Hexu960129

The oral video was submitted into the GitHub repository as the zip file. And it also was submitted into the YouTube. The Link of YouTube was: DSPR-oral-2029883.

Chapter 1

Introduction

In this section, the background, the reasons, the details of the plan and the final objectives were presented in order to give a clear and concise overview of the project process. Some words in this chapter were from the project plan. The re-use was permitted.

1.1 Background

Early speculations about thinking artificial beings such as Frankenstein have been treated as imaginative myths, but this was not the professional Artificial Intelligence (AI) in the true sense of the word. [64] But from these mystical conceptions, attempts to create human-like intelligence have never ceased.

It was not until Turing proposed the hypothesis, 'machines could think.' and the famous Turing Test that the concept of AI in the modern sense became known. [14] As shown in Figure 1, the Turing Test demonstrated that a computer could simulate a human verbal conversation for the purpose of communicating with a human assessor through the communication between the human assessor and the computer. The implementation of the Turing Test displayed the existence of artificial intelligence in Figure 1.1.

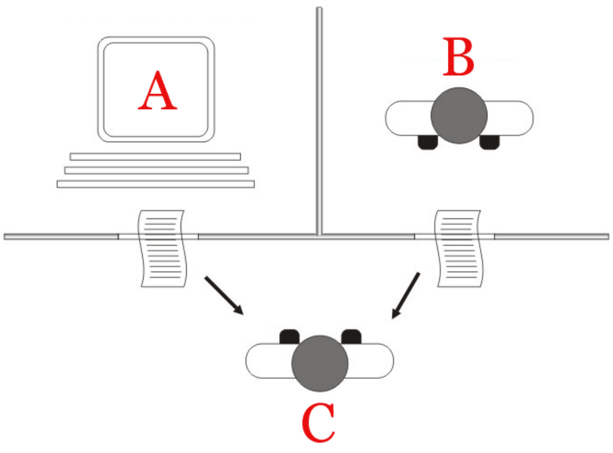


Figure 1.1: The Model of Turing Test [63]

Based on Turing's theory, the formal design of Turing-complete "artificial neurons" by McCullouch and Pitts in 1943 marked the first work of AI. [66] The "artificial neuron" was a pioneering use of mathematical models to simulate the workings of biological neurons. The process of receiving input signals and outputting feedback signals was similar to that of a human response. It therefore demonstrated an attempt by artificial intelligence to simulate human behaviour. In terms of research areas, John McCarthy coined the term 'artificial intelligence' to distinguish AI theory from control theory. This also represented the creation and study of AI as a separate field of study. [51] Since then 'artificial intelligence' has moved beyond the framework domain of control theory and its development has expanded towards humanoid and intelligent.Over the next few decades, AI has undergone several development booms as well as troughs and evolved into the current situation.

At present, the term ”AI” is now commonly used to describe machines that mimic the ”cognitive” functions associated with human thought, such as machines that autonomously recognize and solve problems. [66] The whole process of working with these machines is based on the imitation of human thought processes. In the field of computer science, AI research is defined as the study of ”intelligent agents”: any device that perceives its environment and acts to maximise the chances of successfully achieving its goals. [58] A more detailed definition describes AI as the ability of a system to correctly interpret external data, learn from that data, and use that learning to achieve specific goals and tasks through flexible adaptation. [36]

In terms of life applications, AI is already playing an important role in various fields. In business, AI has successfully revived expert systems. [47] The term expert system actually refers to a class of intelligent computer program systems with specialist knowledge and experience. Artificial intelligence uses a knowledge base and reasoning systems to simulate human experts in making judgement decisions. In the present business, AI is available to be used to make many business choices and decisions as if it were a business expert. Users can obtain business solutions that predict the greatest returns from the expert systems of artificial intelligence. This significantly reduces the risk of business investment failure and enhances the profitability of business practices.AI also plays an important role in improving efficiency in areas such as logistics, data mining, medical diagnosis, and smart manufacturing. [66] [38] [53] Most of these industries are involved in risk detection, workflow planning or pattern mining. Artificial intelligence is often able to plan the best solutions in these areas in the shortest possible time. At the same time, the ’tireless’ nature of artificial intelligence allows it to work actively in real time. This often allows it to discover minute details that humans may have missed. In many cases, these small details are potential risks or major discoveries. So these industries can be more efficient and less risky than ever before with the help of AI. At the same time, the introduction of emerging algorithms such as deep learning has enabled AI to help practitioners of all kinds to handle more difficult tasks with larger amounts of data more quickly. [30]

The field of artificial intelligence at present covers a wide range of different algorithms. It is these algorithms that improve the researches of AI. Among these many algorithms reinforcement learning is increasingly playing an important role. So-called reinforcement learning is the implementation of the interaction between an intelligent agent and its environment. [67] Through hundreds or thousands of training sessions, the agent finds the most profitable solution in the environment. Reinforcement learning is ’greedy’ essentially. Given a reward mechanism, through reinforcement learning, the agent learns to choose an optimal solution in order to obtain the most cumulative reward. This technique has been widely used in areas such as game design and unmanned vehicles.

While AI is developing across the board, it has some flaws of its own that cannot be ignored, and these flaws can often be a hindrance to the process of AI development. Learning, Perception and Reasoning are three of the most important aspects. [66] [32] [55] [35] As the amount of tasks given to AI grows, the tasks become more and more complex. The working part of AI is no longer a simple input-output process. Throughout the work process, AI needs to perceive all kinds of changes in a complex environment to learn the correlation and meaning of all kinds of inputs and to reason about all kinds of complex decisions. All of these processes revolve around learning, perception and reasoning. At the same time, how to make these three aspects better understood by humans is also something that AI research needs to consider. After all, just like human thought processes, AI thought processes cannot be fully articulated. These are also the three areas that need to be considered for this project. How to correctly observe unknown behaviour in a real dynamic system and give correct responses in learning is a hot topic for further development of AI at this stage.

How can people better understand artificial intelligence in reverse? Many researchers have made various attempts to address this question. One of the applications that has come to fruition is Interpretable AI. Interpretable AI aims to solve the problem of how to make AI more understandable to humans and to make the whole AI learning process more transparent. In short, Interpretable AI is a special kind of AI where the models themselves and the solutions they present can be more easily understood by humans after being processed by Interpretable AI techniques.

Two classifications of interpretable AI exist in terms of definition. One is Interpretable AI, which is used to further analyse and explain ’white-box processes’ that are already observable by humans. It can be seen as an additional interpretation of the white-box process. The other one is Explainable AI (XAI). This is an interpretable AI technique that is specifically designed for ‘black-box processes’ that cannot be directly understood and observed by humans. It aims to approximate black-box processes in a way that humans can understand them so that they can be made 'visible' to some extent.

The concept of interpretable AI technology originated in the 1970s and 1990s when various institutions were exploring symbolic reasoning systems. [21] As a result of this exploration, rule-based reasoning systems were increasingly used in a variety of industries. However, these reasoning systems were complex and difficult to understand. [66] In order to make complex AI techniques such as deep learning and genetic algorithms understandable to users and to achieve transparency and fairness in AI, interpretable AI technology was born. [1] [15] At present, interpretable AI has developed many new techniques to make models more interpretable. [45] [54] A growing number of researchers are also engaged in XAI research.

Figure 1.2 shows how the number of publications on interpretable AI technology (including Interpretable AI, XAI and other Expalainable AI) has changed in recent years, demonstrating how hot interpretable AI technology research has become in recent years. [3]

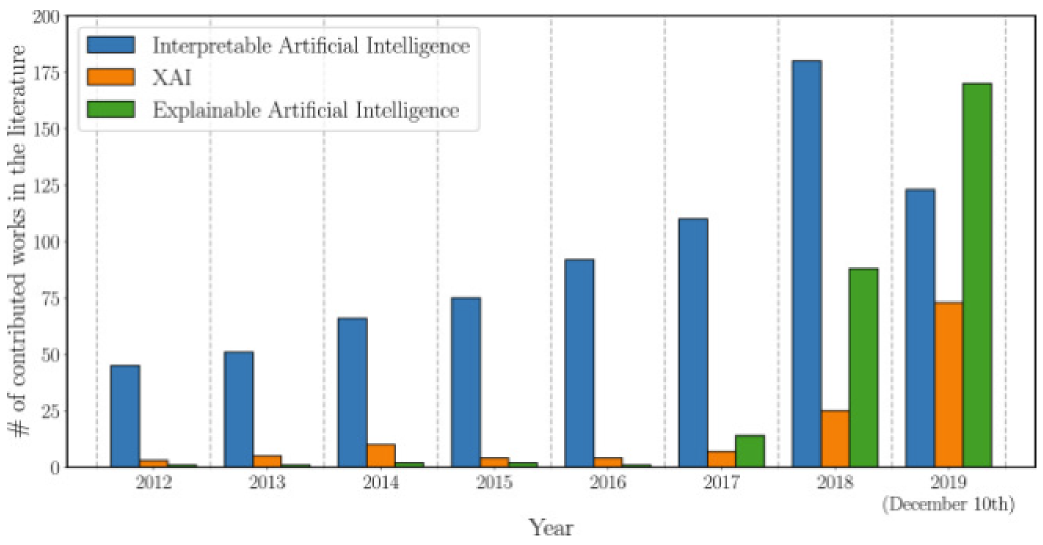


Figure 1.2: The numbers of Publications about XAI Technologies [3]

As the goal of this project was to explore the XAI algorithm used to explain black box strategies in dynamic systems. Therefore, the project focused on XAI techniques.

The goal of an ideal XAI is to explain what has been done, what is being done, what is to be done next, and to reveal the information on which the action is based. In this way, XAI can analyse a product or service to improve the user experience by helping the end-user trust that the AI is making the right decision. [28] At the same time, many dynamic systems are real-time, risky and incomprehensible. The AI used in them needs to be more reliably interpretable, transparent and fair. The analysis of XAI in dynamic systems is therefore an important aspect of productive life now. It is vital for many areas such as automatic vehicles, intelligent energy management, medical diagnostics and high-frequency trading.

Although XAI has the potential to play a significant role in many areas, there are also non-negligible drawbacks to XAI technology according to the research available. As the complex processes of artificial intelligence are sometimes disorderly, it is not possible to fully explain its entire working process using XAI techniques. The incomplete explanation of the working process, especially the black-box learning process, can affect the performance of the model output. This has been confirmed by numerous studies. Models with XAI techniques tend to have poorer performance compared to AI models alone. The performances of basic XAI technologies between the interpretability and the prediction accuracy are shown in Figure 1.3. The figure shows that the rules are the most explanatory, but at the same time the accuracy of the predictions they demonstrate is considerably lower. Complex neural networks, on the other hand, bring the highest prediction accuracy, but their working process and decision making judgements are difficult to understand. The focus of this project was on the rule-based XAI algorithm. Based on the above graph, it can be hypothesised that the rule-based XAI algorithms would provide high explanatory power but would perform much less well than the black box model. Combined the present developing direction, XAI techniques are significantly positioned as interpreters of AI models rather than substitutes in this process. In some aspects of AI models where the need for transparency is particularly important, the use of XAI can give better transparency to the model based on a local approximation of the initial model. This can make it easier for users to understand the entire model, including its working process and output, without compromising on fundamental performance. The most sensible way to make the best use of XAI technology to achieve interpretability is to strike a balance between the transparency gains and performance losses that XAI brings.

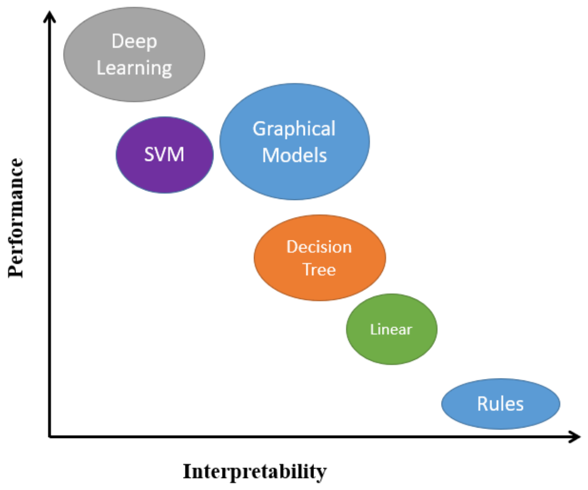


Figure 1.3: The Performance Vs Interpretability of XAI technology

In addition to ensuring a balance between performance and interpretability, the interpretation process of the so-called XAI technology needs to be further improved to meet a variety of new challenges. Like a bridge, XAI technology needs to be designed with security, usability and stability in mind. If these principles and conditions of use are ignored, XAI technology can be used in a variety of ways, including failure, violation of the law and even significant damage to the user's property. Following these principles and meeting the challenges in Figure 1.4 will allow XAI to be optimised for future development.

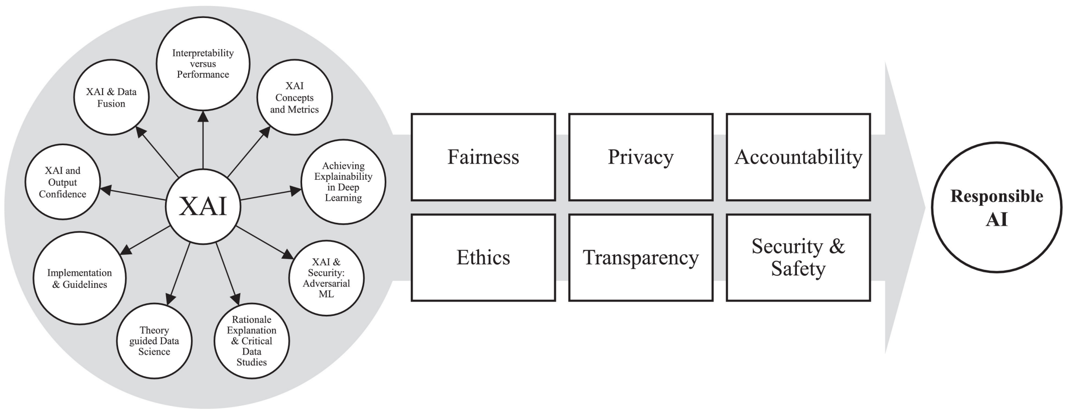

XAI
Interpretability versus Performance
XAI Concepts and Metrics
Achieving Explainability in Deep Learning
XAI & Security: Adversarial ML
Rationale Explanation & Critical Data Studies
Theory guided Data Science
Implementation & Guidelines
XAI and Output Confidence
XAI & Data Fusion
Fairness
Privacy
Accountability
Ethics
Transparency
Security & Safety
Responsible AI

Figure 1.4: The challenges and rules for XAI development [3]

XAI contrasts with the 'black box' concept of machine learning, in which even its designers cannot explain why an AI makes a particular decision. [62] Black-box learning processes, on the other hand, are mostly found in dynamic systems. After comparing the respective properties of XAI and black-box learning, the researchers found that XAI has great potential for dynamic systems. The fundamental reason why XAI is able to play an active role in dynamic systems is that it is thought to be able to better explain black box policies.In contrast to a white box, a black box often refers to a system that can be viewed on the basis of its inputs and outputs (or transmission characteristics) and whose internal working processes are unknowable. [7] Because the internal processes are not known, the black box often need to represent the causal relationship between input and output values according to the 'principle of interpretation'. [27] At present, many dynamic systems include the black box environments. The XAI is able to better explain the internal processes in these black box environments, which can make the black box more transparent and explainable.The differences between the traditional black box model and the black box model optimised by XAI technology are summarised in Figure 1.5.

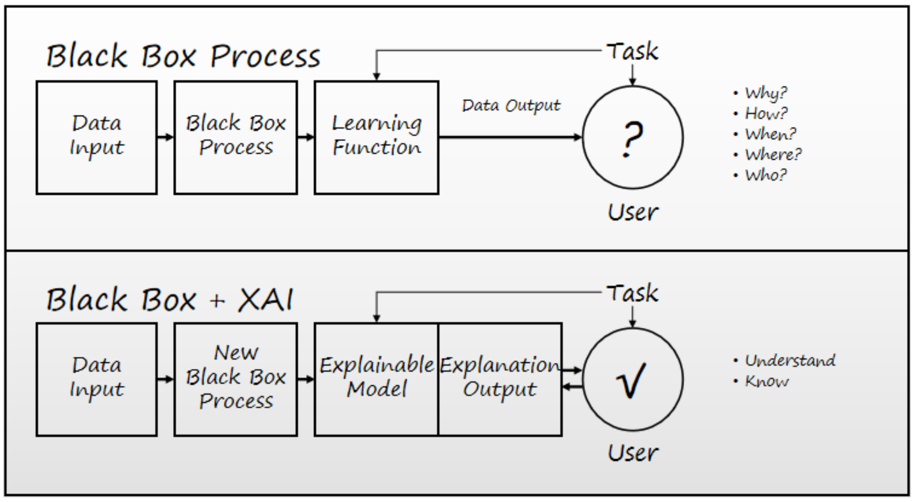


Figure 1.5: The Model of classical XAI for Black Box Process

The figure shows that the XAI technology provides a degree of explanation of the black box model that was previously incomprehensible. The user moves from the initial confusion of ”why?” and ”how?” to ”understand” and ”know”. These transformations show that XAI technology is an important contributor to black box learning. The study of how XAI technology can be applied to black-box learning is of great importance and potential at present. Because of this, the project focuses on the analysis of XAI algorithms in black box reinforcement learners.Further more, both in the course of the project and in the combination of black-box learning and XAI techniques, the contradiction between performance and transparency mentioned above and the challenges and design principles faced by XAI techniques remain a prerequisite for the design of XAI techniques, which are not intended here to replace black-box learning, but rather to simulate it in a local approximation and attempt to explain black-box learning strategies. With this clarity, the exploration of XAI techniques in the field of black-box learning is meaningful.

As the XAI algorithms studied in this project were all rule-based learning, the rule-based learning was the one that needed to be understood. Rule learning is a rule that discriminates between unrecognised instances in the form of 'IF, Then'. It is essentially an unsupervised learning method that is subsumed in classification. With rule learning, the user wants to obtain rule sets that can cover as many samples of instances as possible. The process of obtaining rules is often based on the sequential coverage strategy, whereby the instances are removed from the dataset after the rule covered these instances was generated. The remaining instances are then trained to obtain the next rule. This process is repeated until all instances in the entire dataset have been removed. As this method only processes a portion of the data at a time, it is also known as a partitioning strategy.

As the basis of this background, the project direction was to explore the performance of XAI algorithms for the Black-Box Reinforcement Learners, which is in line with the needs of the times and industry.

1.2 The Plan of the project

The focus of this project was to explore the role of XAI in a black box reinforcement learner. Based on this key point, the whole project proceeded in the following steps.

Firstly, the Open AI Gym platform was studied to understand the platform on which the project was experimenting, from which the classic games Frozen Lake and Lunar Lander were chosen as the simulated black-box reinforcement learning environment. The Frozen Lake environment requires the agent to start from the starting point and avoid the holes on the way to the goal point. This process simulates dynamic movement in a 2D flat space. Frozen Lake is essentially a classical 2-dimensional grid world path planning problem. While the Lunar Lander game is a more complex simulation of a lander landing on the moon. Through the game, the agent must learn to fly and make the lander score the highest in the rules. As the lander is dynamic throughout the game and there are no pre-made tags, it is a very suitable object for testing the effects of XAI. Also in comparison to Frozen Lake, Lunar Lander has no pre-defined obstacle points as well as grid areas, so its complexity is higher. And a higher level of complexity tends to simulate realistic situations more realistically. Lunar Lander is a more realistic attempt to build on the Frozen Lake experience, which was the basis for this project.

Both the Frozen Lake and Lunar Lander problems are essentially dynamic problems that occur in a 2D environment. In the project, after reinforcement learning, the agent found an optimal trajectory in both Frozen Lake and Lunar Lander. This optimal trajectory allowed the agent to obtain the most reward at the end of the game. In the Frozen Lake environment, a trajectory was made up of coordinate points. For each point, the agent had an action to perform when it was at that point. Each pair (position, action) that made up the best trajectory was recorded as data in the dataset. By recording all possible trajectories in the environment into the dataset, the dataset was created for use by XAI. In the Lunar Lander environment, the similar approaches were used. Unlike the process in the Frozen Lake environment, in the Lunar Lander, the final optimised trajectory included 8 status features and 4 actions. the data pair (status, action) which represented each points included 8 status features and 1 action. The 8 status features formed status while 1 action was the predicted action for next status point. Again by recording the trajectories with high final rewards in the environment into the dataset, the dataset of Lunar Lander was created for use by XAI algorithms.

Based on the datasets, the project planed to test different rule-based learning XAI algorithms in two experimental settings, including Greedy Rule List, CN2 algorithm and Rulefit. The status features in the dataset were used as X and the corresponding actions as Y were brought into the specific interpretable algorithm for analysis. Then the performance of the different algorithms was planned to be demonstrated by different representations, such as whether the rules were easy to interpret, whether the rules were accurate, whether the rules were general and whether the rules are critical. The most applicable XAI algorithm was obtained by comparison. The specific performance judging methods used were:

- Dividing the dataset into a train set and a test set. The test set was used to verify the performance of the XAI algorithms.
- The trained rule-based XAI algorithms were used as a controller for the agent. The performance of the agent under the control of the XAI was used to demonstrate the effectiveness of the XAI. Specifically whether the agent reached the target point under the control of the XAI algorithm in Frozen Lake and the final reward of the agent in Lunar Lander would be used to determine the performance of the model under the control of the XAI algorithm.

The strengths and weaknesses of the three XAI algorithms in use were planned to be demonstrated through the determination of the two methods.

For future research, the project would focus on more sophisticated XAI techniques in combination with other more complex black-box learning environments. These future work aimed to use XAI techniques to better approximate black-box learning environments in a more simulated context in order to explain them more rationally. The direction of further development were based on the following some components:

- The Frozen Lake problem is a simple 2D grid world environment and Lunar Lander is a more complex dynamic simulation environment. On the basis of these two, future investigations could focus on the interpretability of black-box strategies in 3D realistic situations.
- Future research will also require optimisation of interpretable algorithms. Further research will build on the original rule learning as well as the simple foil algorithm. Attempts are made to use more complex rule learning to more accurately approximate the black-box strategy locally to achieve better interpretation.
- In the meantime, future work will focus on this part and try to improve the performance of the algorithms used in this project, in view of the problems, flaws and conjectures that emerged in this project.

The above aspects of the research environment and the interpretable algorithm were used to plan future research in order to optimise the project in a more comprehensive and deeper way.

Finally, a summary of the whole project's experiments was shown in Conclusion part. Also details related to the project, the general time planning of the project and the risk analysis of the project were documented in an appendix. A final presentation would conclude this phase of the project.

1.3 The motivation for the project

The selection and planning of the research for the whole project followed multiple considerations and reasons from multiple perspectives. the huge potential applications of XAI in the real world and the huge development possibilities that exist in its current research component were important motivations for this project.

The project was related to XAI techniques and highlighted the role of different rule-based XAI techniques for black-box reinforcement learners in selected different environments. In a practical context, this XAI research was particularly relevant for a smarter and more humane future life. Firstly, XAI technology could meet legal or regulatory requirements in a number of industries. For example, in financial high-frequency trading, XAI could explain the complex black-box environment so that some people do not take advantage of the opacity of the black box to trade illegally. Secondly, XAI could give users a better level of trust. It provides users with more intuitive explanations. With these explanations, even non-expert users could understand how the system works and thus trust it more. At the same time, XAI could be very useful for error analysis. With the explanations given by XAI, some accidents or errors that occur in a black box situation can be easily analysed. A typical example is the analysis of errors in smart car driving tests. Finally, XAI could also be used as a design tool. With the explanations it gives, designers can better understand the specific factors that influence the effectiveness of the tool. By using these factors, designers can then design better tools. Although at this stage, XAI technology is still in the problem of how to perfectly balance model performance and transparency, the changes it has brought to artificial intelligence, to machine learning, are undoubtedly significant. Not only does it open up a new direction for the future of artificial intelligence in terms of "human-machine mutual understanding". It also makes it possible to use AI in a wider range of scenarios because of the user trust that comes with transparency and interpretability. It is therefore clear that XAI has great potential and value for all aspects of life and production in the future. It makes sense to explore its foundations.

On the other hand, many researchers have published a large amount of literature on XAI technology in recent years. Many attempts and studies on XAI technology have also appeared in different fields. The directions covered by XAI techniques in the literature are in healthcare, finance, law and so on. However, it is worth noting that the research on rule-based XAI techniques in the area of black-box reinforcement learning in dynamic systems is not very well developed. Black-box reinforcement learners in dynamic systems tend to emerge from the reinforcement learning process. Its many times occurs in dynamic systems such as unmanned vehicles, flight vehicle simulations and other directions. In contrast to applications where there are clear rules to follow, such as the rise and fall of currencies and the violation of laws, where there are clear constraints, the laws that humans can understand or the factors that influence reinforcement learning are not very ready to be shown while being well understood in black box reinforcement learners. This feature has resulted in rule-based XAI algorithms that have not been extensively tested and validated for use in black-box reinforcement learners. Given this state of research, the project planed to further analyse the application of XAI techniques in specific experimental settings and hopes to demonstrate their beneficial optimisation for the system as a whole. This could provide a more comprehensive introduction to the application of XAI techniques to dynamic systems, and in particular black-box reinforcement learning. To some extent, this project could bridge the gap in the broad understanding of the use of rule-based XAI techniques in black-box reinforcers in dynamic systems.

The strong motivation for this project was not limited to the current state of real-world applications and the current state of research, but also came from the motivation of the intended purpose of the project itself.

In terms of the project itself, the initial aim of the whole project was to verify the effectiveness of different rule-based learning XAI algorithms on black-box reinforcement learners in different dynamic environments. The expectation of the project was to try to demonstrate that rule-based learning XAI algorithms can explain Black River strategies well. Specifically by rule-based XAI algorithms, their models would locally approximate the effect of black-box reinforcement learners as much as possible while giving intuitively understandable rule explanations to the observer. The initial aim of the project was considered to be achieved if the rule-based XAI technique could act as an interpreter rather than a substitute for the black-box reinforcement learner in the project, giving the model better transparency while maintaining a certain level of performance. Also, as the project gave a variety of XAI algorithms, it was considered an additional purpose to compare the effectiveness of different algorithms in order to find the most suitable XAI algorithm for the current environment and the black-box reinforcement learner. As a beginner in the field of XAI technology, these two aims presented a great challenge to the project implementers while providing a constant and strong motivation.

At the same time, the project was a meaningful exploration of rule-based learning of XAI algorithms. The project was also significant for the enlightenment of this class of XAI techniques. The most immediate significance of this project was that participants are able to gain a comprehensive understanding of this class of XAI techniques. Different kinds of rule-based learning XAI algorithms were compared and investigated by testing them in specific black-box reinforcement learning problems - Frozen Lake and Lunar Lander. This provided participants with a dialectical and diverse understanding of some of the algorithms currently available in XAI and gave participants a suitable grounding in the XAI field. This provided positive inspiration for future research directions should participants still wish to pursue XAI-related work or research in the future. At the same time, outside of the project itself, XAI technology is a very promising area of research that needs more attention from non-specialists at this stage. The project, as a Master's graduation project, further stimulated a strong academic interest in the field of XAI among the participants. By focusing on the participants, the project would be able to reach out to a wider audience of lay people, thus allowing XAI to gain more attention and further development. All in all, this was a very interesting and forward-looking project at this stage of the emergence of XAI technology.

1.4 The Challenges of the project

After stating the motivation for the project, the challenges of the project could not be ignored. Looking at the project as a whole, the challenges of the project existed in three main parts:

- The project needed to address the black box reinforcement learning problem. It also required experimentation on the new Open AI Gym platform. This was a daunting challenge for the project implementers who had never been exposed to the field before. It took time and effort to figure out how to get the best trajectory for the agent in reinforcement learning, and how to find the right data in the best trajectory to import into the database for later use in interpretable AI algorithms.
- Secondly, the project focused on using rule-based XAI algorithms to explain the process of black-box reinforcement learners. The learning process of a black-box reinforcement learner was complex and invisible. It would be a great challenge to select the right training data to obtain more effective and interpretable rules.
- Finally, the current reality of life was also a great challenge for the project. The current trend of pandemics was not over. This could make the original school life and project learning more or less affected. At this particular time, uncertain realities also posed a challenge to the project.

1.5 Summary of project

Overall, this project was an experiment in XAI technology. It was a meaningful attempt to explore XAI technology, a new direction of artificial intelligence with great potential for practical application and great research value. Throughout the history of artificial intelligence and the development of XAI technology, the unique significance of this project and the meaningful exploration that it aims to bring about is the main motivation for the project. However, because of the current state of the pandemic and the difficulties that the project presented to those who are starting out, the project still required a great deal of time and experience to overcome the various challenges. Overcoming the challenges and completing the project step by step was not only the meaning of the whole process, but also a comprehensive basis and prerequisite for further research by the project implementers. The whole project had been meaningful and rewarding.

Chapter 2

Literature Review

Various types of prior research as well as literature existed in the areas covered by this project. The project plan could be enlightened to the greatest extent possible by reading the relevant literature. Previous research and literature summaries also be useful for avoiding the risk of misdirecting the research process and making a lot of ineffective work. The extensive literature reading was used as academic background and as a basis for this project. Some words in this chapter were from the project plan. The re-use was permitted.

2.1 Explainable AI (XAI)

To study XAI in the context of the project, it is first necessary to understand what is meant by XAI in the general context. Previous research has examined and analysed in detail its specific definition and role in various industries.

Increasingly, researchers are arguing that explainable artificial intelligence (XAI) is once again an emerging multidisciplinary research field. [5] [11] [68]. In the face of the diverse and complex interpretable AI requirements and representations in different domains, Markus Langer and his research team constructed a model to show the main concepts and relationships that need to be considered when aggregating interpretable AI work in different domains. Their model allowed XAI researchers from different domains to be used as a common denominator, highlighting the interdisciplinary potential that XAI showed in various domains. [41] Figure 2.1 showed the model created by Langer.

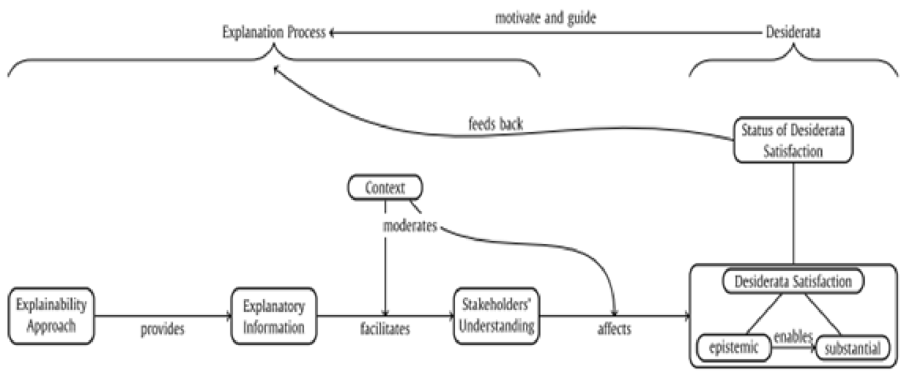


Figure 2.1: The Model of how the XAI approaches work [41]

Langer's research could guide XAI researchers in the design of research in their specific field, while to some extent the model created by Langer could also inspire the development of future XAI methods, thus furthering XAI research on stakeholder satisfaction with needs. This had far-reaching implications. In the case of this project, however, Langer's research only provided guidance and ideas for designing specific XAIs in this project, but not much in the way of detailed ideas.

Given this situation, previous specific applications of XAI techniques in other fields needed to be investigated to find project specific methodological references from them.

Chen's attempt to use XAI techniques to interpret convolutional neural networks (CNNs) in the field of bearing fault diagnosis was an example of research that provided a methodological reference. In his research, the vibration signal was converted into an image signal through short-time Fourier variation so that it could be more easily identified and classified and interpreted by the XAI technique. The final fault signal was displayed in a highlighted form. [29] Although the subject of this study was different from the project. However, the method of mining the raw information for data that could be easily processed by XAI technology could be applied to this project. Figure 2.2 showed Chen's novel approach to fault diagnosis decision making with XAI technology.

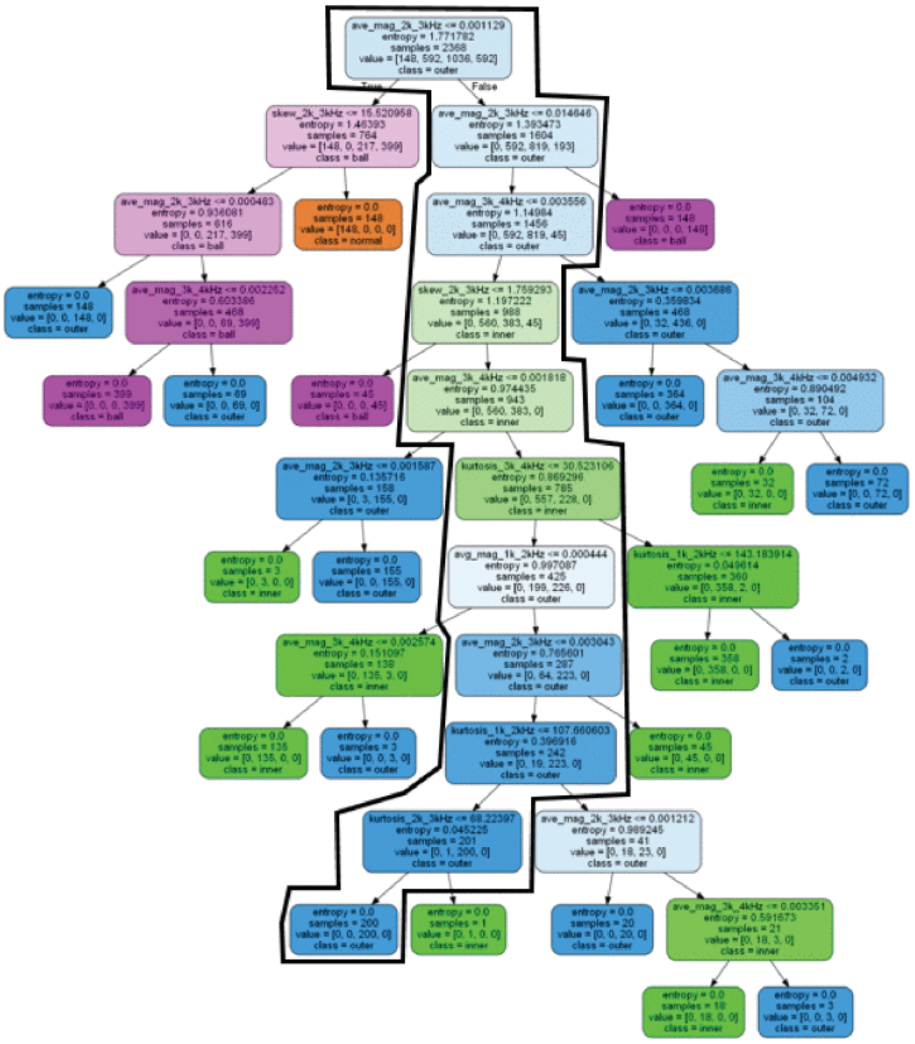


Figure 2.2: The Model of how the XAI technology work in Chen's research [29]

Kuzlu's research focused on the interpretation of power forecasts for solar power using XAI technology. In his research, he used various interpretable algorithms such as LIME, SHAP and ELI5 to make predictions on electricity generation. Although Kuzlu's research was in a different field to the one covered by the project. However, generation forecasting itself was a black box process and its setting was similar to that of the project. Similar black box processes could guide the research for this project. Also Kuzlu's research approach of using multiple interpretable algorithms for comparative analysis could be used in this project. [39] The table 2.1 below showed the final results of the three interpretable algorithms applied by Kuzlu in his research. The effect was evaluated in terms of a specific value of Root Mean Square Error (RMSE).

At the same time, Rich summarised the practical applications of XAI technology in different areas. Although his summary was not fully representative of all situations, the idea that 'XAI techniques need to balance both model performance and interpretability' was valuable. [8] Rich's research has guided the design of the project, which was followed by the principle that 'XAI technology is an interpreter, not a

Model	Features removed	RMSE (Percentage)
w/all features	-	7.236
LIME	TCWL,TCC	7.228
SHAP	SP,TCWL	7.216
ELI5	TP,TWCL	7.235

Table 2.1: Performances of different algorithms in Kuzlu's research [39]

replacement'.

In conjunction with the above previous research, a process for the design and application of a generic XAI technology could be broadly constructed. However, this project focused on the study of XAI technology in a specific context, so more specific prior research needed to be understood as well.

2.2 Reinforcement Learners

Having understood the current stage of research in generic XAI technology, previous research in the area to be covered by this project also needed to be further understood. The project was an exploration of interpretable AI techniques based on reinforcement of learners. Therefore, prior research on reinforcement learning needed to be understood as an academic reference before the research related to the project could be specifically understood.

Li summarised the main areas of application for reinforcement learning at this stage, such as healthcare, robotics, energy, dynamic systems and so on. He also presented a summary of the areas of application and success stories similar to the one in this project. [74] From his research, project specific reinforcement of the learner's environment could be constructed with reference. Sindhu summarised the use of reinforcement learning in dynamically changing environments. This research built on the Li's research and was more relevant to the environment involved in this project. Also Sindhu proposed future enhancements in his research. [56] The future enhancements also gave some insight into the future outlook of this project. For specific reinforcement learning algorithms, Timothy's work on deep Q learning for continuous control [43] and Volodymyr's work on Q learning in Atari games [52] both provided some code design inspiration and reference for this project.

Through the above-mentioned literature, the reinforcement learning techniques involved in this project, as well as the generic XAI techniques, were further understood. The vision of XAI techniques for reinforcement learners in dynamic systems, which was the focus of this project, required more specific prior research as academic support.

2.3 XAI for the reinforcement learners in dynamic systems

XAI at this stage has specific research and analysis in many dynamic areas and many Black-Box enviroments. Much of the research literature could inspire the specific implementation of this project.

In the dynamic field of high-frequency trading, Kirilenko verified that XAI could bring better interpretability and transparency to high-frequency trading by studying the impact of high-frequency trading on electronic trading. This helped to maintain fairness and stability in electronic trading markets. [37] In the field of dynamic medical diagnosis, researchers had also made medical diagnosis cheaper, more efficient and more satisfying for patients through various XAI technologies. To a certain extent, this led to enhanced medical diagnosis. [31] [40] [69] For dynamic systems in constant motion, XAI is also coming into play. The most typical area is self-driving cars, and Li's latest research showed the importance of XAI in self-driving car technology. For non-AI-expert users such as drivers, efficient interpretability was what increases customer satisfaction with AI in self-driving cars. [42] This was also important in the design of XAI algorithms in this project.

The working process in the above domains was often not very transparent and its working process could involve a black box environment. The use of AI techniques such as reinforcement learning in these black box environments could make the whole process very complex. In order to understand this complex black-box reinforcement learning, XAI was used to explain the whole reinforcement learning process. At this stage, much research had focused on this aspect.

In her research, Erika Puiutta investigated interpretable reinforcement learning and found that black-box reinforcement learner with XAI algorithms could simplify the complexity of the models, which facilitated better understanding of the process by non-expert users. At the same time, Puiutta also found that achieving interpretability of reinforcement learning for non-expert users required interdisciplinary research in order to obtain an optimal form of explanation. [59] Jeff Druce found that XAI could be used in autonomous systems for deep reinforcement learning to improve user reliability, and that while the black-box reinforcement learning systems in general were relatively brittle, black-box reinforcement learning became more understandable when explained through XAI processing. Figures 2.3 and 2.4 displayed the workflow of Jeff's experiment and the final results of different experimental modes.

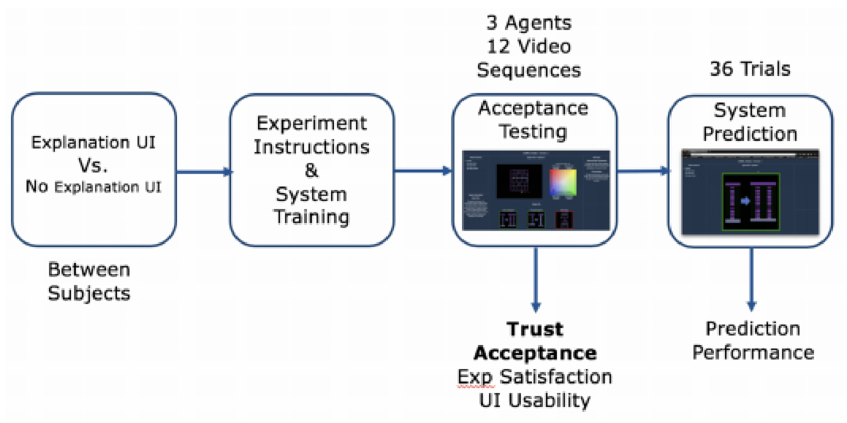


Figure 2.3: The workflow in Jeff's experiment [20]

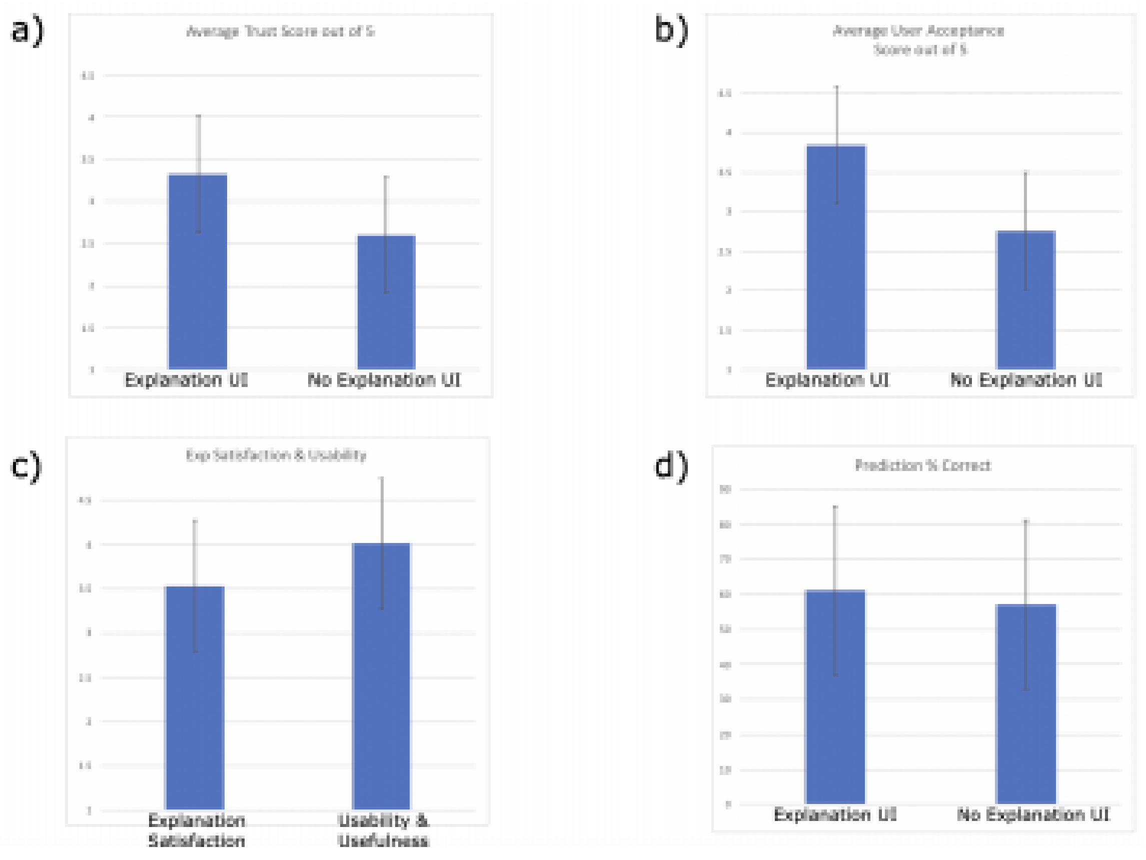


Figure 2.4: The results of different experimental modes [20]

For different modes, The XAI technology could bring the positive influences. These positive results encouraged users and improved the overall efficiency of the system. [20]

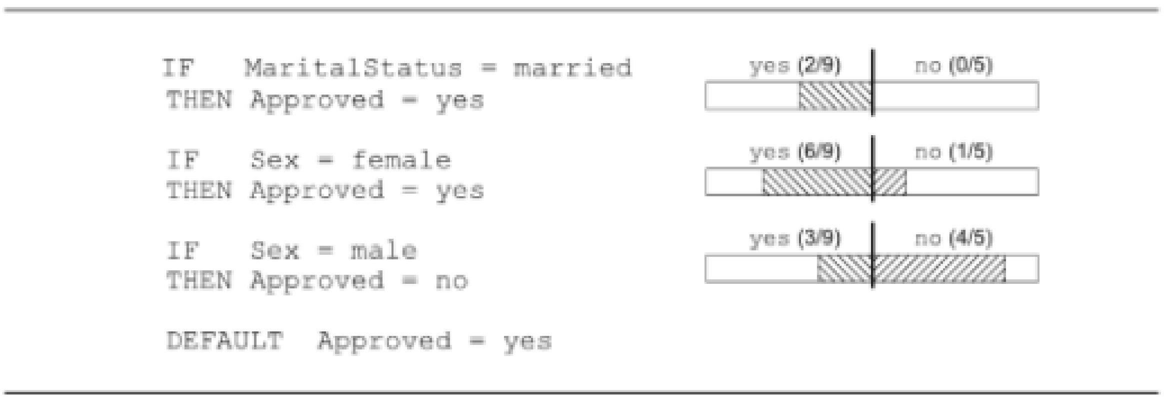
Also XAI could bring a more immersive visual interpretation to black box reinforcement learning. These could increase the transparency of the whole black-box reinforcement learning process. [19]

Combined with the current stage of XAI's application in various dynamic systems and its specific performance in black-box reinforcement learning, researchers could gain a comprehensive understanding of the general working mechanism of XAI. On the basis of this understanding, the planning of this project could be gradually conceived. At the same time, although the above research encompasses many areas where XAI was applicable, there was no literature on dynamic systems based on black-box reinforcement learners in dynamic systems specifically in a direction similar to the one required for this project. Most of the literature did not provide a detailed comparative analysis of the specific performance of different rule-based XAI algorithms in dynamic systems. Therefore, the comparison of different rule-based XAI algorithms in this project could be explored and analysed as a focus. To some extent, this also added a theoretical dimension to the application of XAI in dynamic systems at this stage.

2.4 Rule Learners

As the XAI algorithms for this project were based on rule-based learning. Therefore, the relevant literature on rule-based learning could also serve as inspiration for this project.

Rule learning, also known as rule induction, is a field of machine learning that is based on a working model of extracting formal rules from a set of observations. The extracted rules may represent a complete scientific model of the data or only local patterns in the data. Many researchers have attempted to improve XAI techniques through rule-based learning. A simple rule base for describing a database is shown in Figure 2.5. [23]


```
IF   MaritalStatus = married
THEN Approved = yes

IF   Sex = female
THEN Approved = yes

IF   Sex = male
THEN Approved = no

DEFAULT  Approved = yes
```

Figure 2.5: The sample ruleset describing the data [23]

Gilpin's research could show that rule-based learning can improve the transparency and interpretability of AI. This was exactly what XAI needs to achieve. [26] However, it was also important to note that although Gilpin's study demonstrated the effectiveness of rule-based XAI, often the explanations provided by these XAIs were not systematic and standardized, which was the pain point of rule-based XAI technology at this stage and the direction of future development. Jasper's recent study used diabetes self-management decision making as the base task, and compared two of the most classic approaches to XAI, rule-based and exemplar-based, to show the differences in performance between their systematic understanding, persuasiveness and task performance. Through evaluation tests, Jasper and his team concluded that rule-based XAI could lead to some positive effects on system understanding. [71] But this impact was relatively small, because the rule-based explanation was only concerned with individual decision details, and it did not show a positive effect on the whole potential causal relationship. This was one of the problems faced by XAI techniques that apply rule-based learning at this stage.

In response to these phenomena, Arun Das had summarised the evolution of XAI technology between 2007 and 2020. Following the summary Das also evaluated the results generated by eight different XAI algorithms on the data and discussed the associated limitations. Many of these XAI algorithms were rule-based designs. It was clear from this literature that rule-based XAI algorithms have been studied and used to some extent. [17]

XAI algorithms based on rule-based learning have been greatly developed by many researchers. There are a wide variety of rule-based learning algorithms that are widely used now. The most common ones used are Decision Tree, Rulefit, Foil, CN2 algorithm and many more. Each of these algorithms has its own advantages and disadvantages.

In combination with this literature, some of the academic underpinnings of rule-based learning XAI techniques could be accessed to refine the project plan. In the project, the basic rule-based learning algorithms, Greedy Rule list, CN2 algorithm and Rulefit would be applied to explain the work of the AI in the black-box reinforcement learning environment.

2.5 Open AI and Project Environment

Finally, as this project was tested experimentally by using a specific Frozen Lake environment in Open AI Gym as well as the Lunar Lander environment. Understanding Open AI Gym and the two environments was fundamental to this project. There was already a large body of literature as well as previous research available for this purpose.

Greg Brockman wrote a white paper on the use of Open AI Gym, a toolkit for reinforcement learning research, and introduced each of its components. [6] At this stage Open AI Gym had been used for a variety of reinforcement learning research, such as robotics research, computational disease investigation, urban development design research and more. [75] [16] [72]

Once Open AI Gym was clear, the first thing that needed to be delved into is the research on Frozen Lake environment. As an introductory test to the Open AI Gym, the research on Frozen Lake was also considered to be the basis for using the Open AI Gym.

The idea of reinforcement learning in a situation, where performance was imperfect, was presented in the study by Gao et al. via the Frozen Lake problem. From his research, it was clear that the Frozen Lake problem itself had some uncertainty, which could lead to an agent potentially exhibiting imperfect performance in the environment. [25] Although the study by Gao et al. failed to take into account the relatively fixed division of points in Frozen Lake in their analysis, it could give guidance to the Frozen Lake part of the project. The study by Zhao et al. instead used the Frozen Lake problem as a typical case study. They proposed that the use of asynchronous reinforcement learning methods for path planning in small discrete spaces such as Frozen Lake could avoid local minima to a certain extent. [73] This served as a reference for the design of the project algorithm. It also served as an inspiration for future enhancements to the project.

Meanwhile Matteo et al. in their study proposed the introduction of inductive controllers in problems such as the Frozen Lake problem similar to the position of a teacher in human societies to supervise and teach the agent to learn a better path. [70] Although the idea of a 'teacher' in this experiment was only used as a monitor, there were inherent similarities to this project. This project also required the introduction of a new component, but instead of a 'monitor', it was an 'interpreter'. Both, however, were judged by the performance of the agent. The explainer in this project was used to generalise rules in order to explain black box policies. The Matteo et al. study, on the other hand, incorporated a monitor to analyse behaviour in order to reset it. Both had similar strategies for observing agent behaviour. To a certain extent, Matteo et al.'s study could also be used as a reference for the Frozen Lake part of this project. A visualisation of the Frozen Lake environment was shown in Figure 2.6.

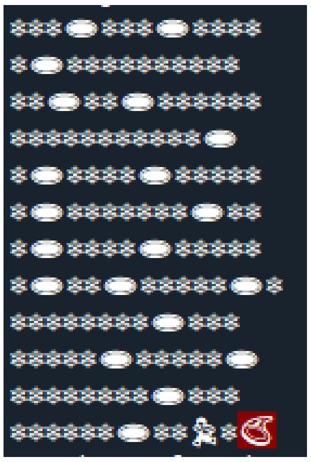


Figure 2.6: The game interface visualisation of Frozen Lake

In this project, Lunar Lander, a game that simulates the moon landing, was used as another experimental platform to study XAI algorithms under black-box reinforcement learning. A wide range of research literature on Lunar Lander detailed this dynamic simulation system and the associated reinforcement learning research in its environment.Figure 2.7 showed the game interface of Lunar Lander.

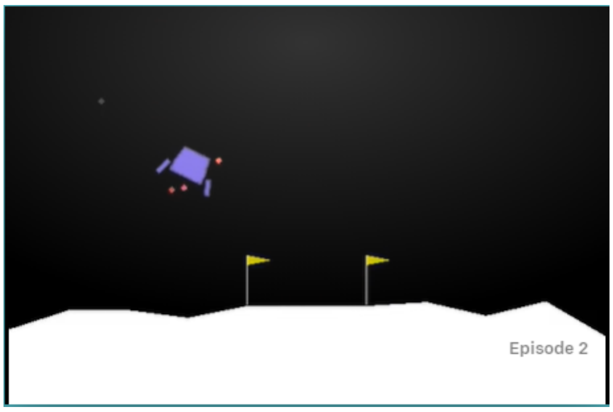


Figure 2.7: The game interface of Lunar Lander [6]

Abhijit Banerjee analysed the evolutionary neural network topology variations of the evolutionary algorithm (JADE) and the meta-heuristic algorithm (PSO) in three settings: simple policy gradient, A2C and deep deterministic policy gradient (DDPG), in order to find an improved policy gradient to evolve the topology of the neural network. In testing this new scheme, Lunar Lander was used as a continuous action model. The researchers concluded that optimising the DDPG has the advantage of fast convergence and minimal forgetting. This could lead to better neural network topology, resulting in better results for the Lunar Lander model. [4] This study showed that optimisation strategies gradient reinforcement learning could help optimise topology and eliminate the need for unnecessary parameter tuning. However, the real Lunar Lander problem involved uncertainty in evolutionary neural networks. This aspect was not adequately considered in Abhijit's study.

The same optimisation improvements had been seen in the work of other researchers, with Lu using a control model-based approach to learning the optimal control parameters rather than traditional dynamics. [46] This approach improved the efficiency of the overall vehicle model but ignored the uncertainties and contingencies caused by the dynamics in real-life situations. Peters then explored the optimisation of spiking neural networks for the Lunar Lander problem. [57] Spiking neural networks could fine-tune almost all small details on Lunar Lander, but their drawbacks are also obvious. For large, complex models of a high order, spiking neural networks needed to be further optimised to make machine understanding and processing easier and more like real human processing.

All three of these methods optimised Lunar Lander's performance to some extent. However, they had not given much thought to the uncertainty. On this basis, Soham and the other researchers used Sarsa and Q-learning two algorithms to tackle the Lunar Lander problem and then compared and analysed the effects brought about by the different techniques while verifying the robustness of each technique using the additional uncertainty. This made the whole study more comprehensive and adequate. Through experiments, Soham found that deep Q learning in the Lunar Lander environment resulted in more average rewards. Both Sarsa and deep Q-learning overcame the additional uncertainty and achieved a consistently positive average reward. Taken together, deep Q learning delivered better results. [24] But Soham's study was still somewhat flawed. His experiments took into account uncertainty but did not do much in the way of interpretability. In the real world, a lunar landing is a complex process involving large and variable amounts of data as well as the activities of the astronauts in the vehicle. Even machines are sometimes unable to understand the process and the results in the face of complex algorithmic forms. The interpretability and transparency of the algorithms were crucial in the high-risk lunar landing process, which directly involved the lives of the astronauts. So a more optimised design should reflect better interpretability.

High-risk dynamic systems with black-box reinforcement learning like the Lunar Lander require interpretable artificial intelligence to improve their transparency and ensure a fairer and safer way of operating for users. This is exactly the area that this project aims to address.

The literature summarised above provides academic support for the various components involved in the overall project as well as providing inspiration for specific implementation methods.

Chapter 3

Methodology and Materials

Following an introduction and a summary of the literature, relatively adequate contextual and academic support provided ideas for the design process of the project. In this section, details about the project process were presented. The details included mainly the methods and relevant materials used in the project.

3.1 Overall project program

Although this project covered different parts of the study based on two different environments, Frozen Lake and Lunar Lander. In these two environment, three different rule-based XAI algorithms were planned for this project. The overall research approach was consistent in different environments.

The flow chart of the overall research strategy was as follows in Figure 3.1:

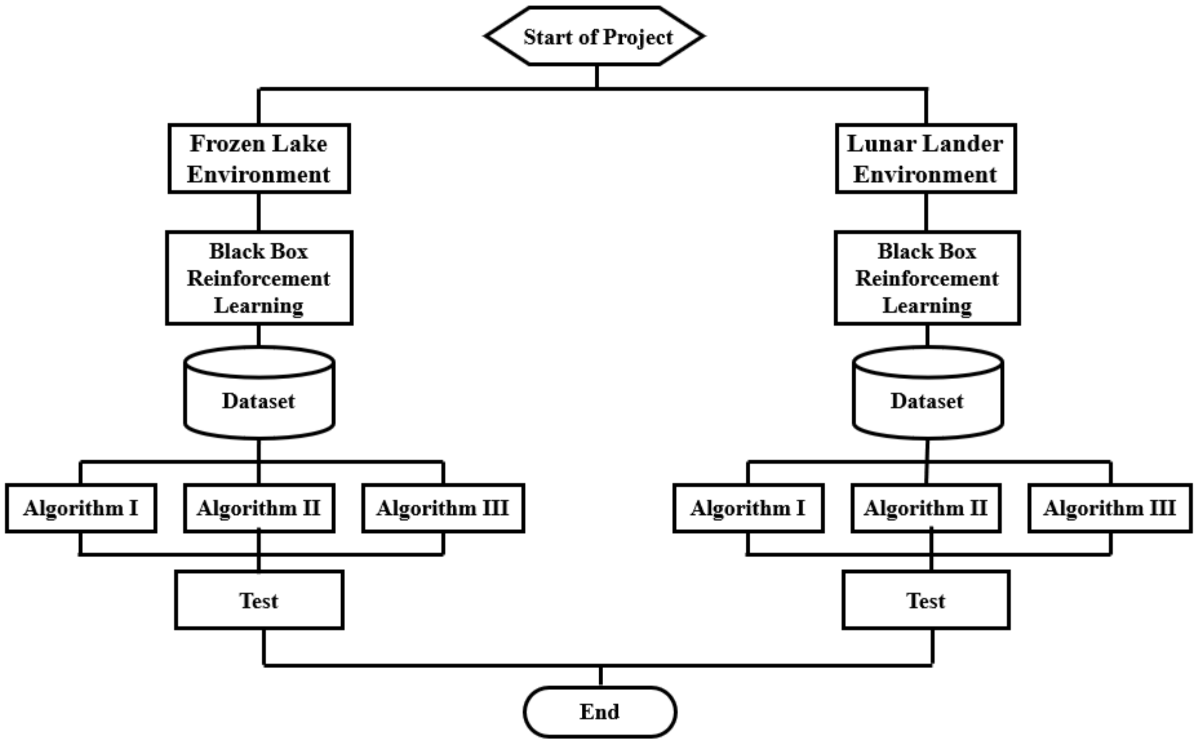


Figure 3.1: The flow chart of the overall project

Taken as a whole, the entire project followed a quantitative research approach. Firstly the hypothesis was formulated. XAI algorithm could provide a degree of explanation for the black-box reinforcement learning process of the dynamic systems involved in the project. Through rule-based learning explanations, humans could better understand the working process of black-box reinforcement learners. At the same time the new model processed by the XAI technique was predicted to exhibit relatively poor performance of the effect. This was in line with the paradoxical nature of XAI technology between performance and transparency. After formulating the hypothesis, the whole process of reinforcement learning with XAI was implemented in concrete code. The effects of the XAI technique were shown in relation to the data recorded throughout the experiments. The hypothesis was further tested by the results presented. The whole part of the study followed 'A Priori' Theory of quantitative research.

Next, comparative analysis and interaction analysis were used to compare the effects of different algorithms in the specific experimental validation process. The results obtained by the different algorithms were quantified on a uniform scale and compared in the same environment and under the same conditions. Such comparative analysis provided a more intuitive picture of the effectiveness of the different algorithms. At the same time, the rules obtained by the XAI algorithm were used as a controller for the agent to mimic the black-box strategy, and the performance of the agent after this interaction analysis was a more intuitive reflection of the effectiveness of the XAI technique for the local simulation and interpretation of the black-box strategy. Taken together, both the comparative and interaction analysis provided a more intuitive presentation of the project.

As the basis of the general methodology described above, the project was planned to be divided into two environmental components, Frozen Lake and Lunar Lander, in each of which the corresponding three XAI algorithms would be applied to explain the black-box strategy of the environment. In this section, each environment would be treated as one part. The reinforcement learning approach and the XAI approach adopted for the corresponding environment would be presented in each part. Descriptions of the construction for the environments and the corresponding improvements were also presented in this chapter.

3.2 The Program for Frozen Lake

Frozen Lake is one of the simpler simulation environments in the Open AI Gym, a maze-like pathfinding game environment in a 2D grid world. This relatively simple environment can be used as a basis for testing XAI algorithms. To implement XAI algorithms in this environment for this project, certain environmental improvements, suitable reinforcement learning methods and the corresponding XAI algorithm composition were essential. Attempts on this environment were based on the following approach and use the following materials in this project.

3.2.1 Environment Construction of Frozen Lake

In this project, a new Frozen Lake environment was architected based on the original Frozen Lake environment provided by the Open AI Gym platform in order to ensure that a sufficient amount of data could be trained by XAI algorithms. The original Frozen Lake environment was presented before the introduction of the changed environment.

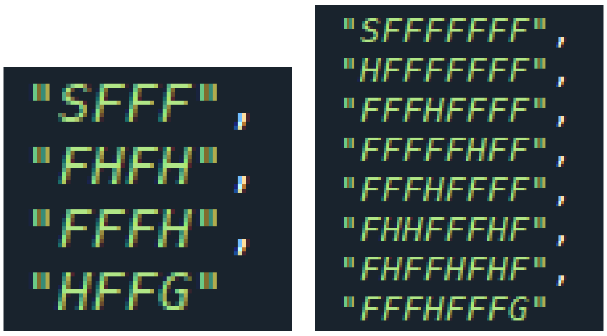
3.2.1.1 Original Frozen Lake Environment from Open AI Gym

Essentially Frozen Lake is a problem similar to a chess game in a grid world, which is described in the Open AI Gym platform as: 'You and your friends were tossing a Frisbee in the park one winter day when you dropped it in the middle of a lake. Most of the lake is frozen over, but there are a few holes where the ice has melted. If you step into one of the holes, you will fall into the lake. You have to get the frisbee. However, the ice is slippery, so you won't always move in the direction you intended.' [6] For the visual representation of this problem, different letters were utilised to represent regional states. Table 3.1 presented the descriptions of different letters.

Letter	Description	Status
S	Starting Grid	Safe
F	Frozen Gird	Safe
H	Hole Grid	Fall to doom
G	Goal Grid	Get the frisbee

Table 3.1: The descriptions of different grids in Frozen Lake

When the respective grids had been defined, the most basic 4x4 and 8x8 grid maps given by Open AI Gym could be represented as follows in Figure 3.2.



```
"SFFF",
"FHFH",
"FFFH",
"HFFG"
```

```
"SFFFFFFF",
"HFFFFFFF",
"FFFHFFFF",
"FFFFFHFF",
"FFFHFFFF",
"FHHFFFHF",
"FHFFHFHF",
"FFFHFFFG"
```

(a) 4x4 map (b) 8x8 map

Figure 3.2: The original Frozen Lake Grid Maps

After the state of each location grid had been specified, the agent's action state also needed to be specified. In the Frozen Lake environment, the agent was specified as being able to perform four random actions, Up, Down, Left and Right. To facilitate later processing of the relevant actions, the relevant actions were represented as specific numbers. In this way, in the future prediction action session, the specific predicted actions could also be represented by the values shown in the corresponding numbers. The use of numbered values instead of actions also facilitated the calculation of the algorithms. The actions could be described in Table 3.2.

Action	Number
Left	0
Right	1
Down	2
Up	3

Table 3.2: The descriptions of different actions in Frozen Lake

For specific status displays and action displays, the corresponding presentation settings were structured in Frozen Lake's environment code. The number of states and actions and how they form data pairs were shown in Table 3.3.

(a) Numbers of States

Number of Action	Number of State
4	$nrow * ncol$

(b) The different state pair

Number of Action
(s, a) ,where s in nS and a in nA

Table 3.3: Status setting related codes

Sub-table (a) in the above figure indicated that there were 4 different actions numbered in this environment, Up, Down, Left and Right. Also each grid was set to a state. Multiplying the number of rows by the number of columns displayed that there were 16 state numbers corresponding to the corresponding grids in a 4x4 map. Similarly, there are 64 states in an 8x8 map. Sub-table (b) showed the pairing of actions numbers and grid states formed a kind of new two-dimensional arrays, meaning the data obtained by taking a particular action in a particular state, i.e. a tuple of (transfer probability, next state, reward, completion marker).

In order to better guide the movements and define the planning, Frozen Lake had been introduced with a reward mechanism. The mechanism stipulated that only one point was awarded when a frisbee was obtained, whilst no points were scored or deducted for any other action.

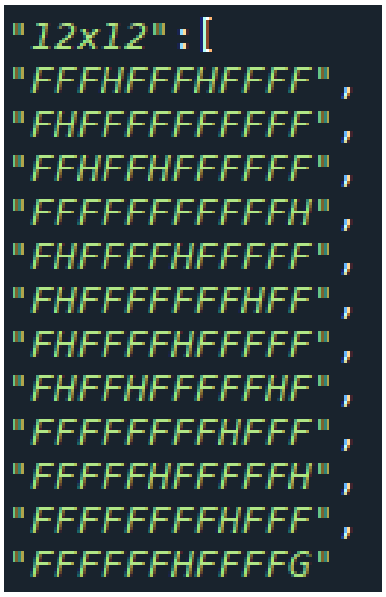
3.2.1.2 Improved Frozen Lake Environment

The original Frozen Lake environment provided by the Open AI Gym platform was a simple toy test environment. The following drawbacks in the analysis of the interpretability of its reinforcement learning affected the proper conduct of the experiment:

- The original Frozen Lake environment was only available in 4x4 as well as 8x8 maps for testing. As these two maps only provide 16 and 64 grids of state respectively, the maps needed to be further extended in order to obtain a sufficient amount of data for the next part of the XAI algorithm.

- In the original environment, the position of the frisbee, your starting position and the position of the hole were fixed on the map and were shown in the visual grid world as 'S', 'H' and 'G ' are all fixed positions. This would result in only one optimal route trajectory for agent after RL training. This would result in a small-size final data output. The small amount of data was not sufficient to support the needs of Explainable AI algorithms at a later stage.

As the basis of two points above, the newly extended map was a 12x12 grid world map.The starting point was randomly generated on any grid on the entire map. The visualisation of this improved environment in this project was shown in Figure 3.3.



```
"12x12":[
"FFFHFFFHFFFF",
"FHFFFFFFFFFF",
"FFHFFHFFFFFF",
"FFFFFFFFFFFH",
"FHFFFFHFFFFF",
"FHFFFFFFFHFF",
"FHFFFFHFFFFF",
"FHFFHFFFFFHF",
"FFFFFFFFHFFF",
"FFFFFHFFFFFH",
"FFFFFFFFHFFF",
"FFFFFFHFFFFG"
```

Figure 3.3: The improved Frozen Lake 12x12 map

In the improved maps, holes were placed in fixed positions. There were 20 holes in the 12x12 map. These holes would be used as barriers to restrain the agent's movements during the test.

3.2.2 Reinforcement Learning Process of Frozen Lake

Once the map was constructed, the black-box reinforcement learner was used to train the agent to plan an optimal path to its destination in improved Frozen Lake environment. This involved a reinforcement learning process based on the glacial lake part of this project.

3.2.2.1 Q learning algorithm

In the previous chapters, reinforcement learning (RL) was introduced. The Deep Q Network (DQN) was used as a reinforcement learner to train the agent for the two dynamic environments of this project.DQN is a novel approach that combines Q-learning algorithms with neural networks. It is based on the Q-learning algorithm. Before introducing DQN, the main ideas of Q-learning, as the basic part of DQN, need to be presented clearly.

Q-learning is a reward-based reinforcement learning algorithm. It is very popular in the field of reinforcement learning at present. [50] Its property of maximising successive or aggregate rewards allows it to plan optimal paths in problems based on reward mechanisms.There are four basic components of the

Q-learning algorithm, namely the Q table, action selection, environmental feedback and environmental update. Among these four part, Q table is the most important component.Q-table are used as the core of Q-learning to reflect the expected reward for actions taken in a given state. Its performance function is:

$$Q : S \times A \Rightarrow R$$

Where S meant the set of agent's states, A meant the set of agent's actions, R meant the reward function influenced by S and A.

Since the agent was in a decision making situation throughout the Frozen Lake environment where it needed to perform successive actions, the whole decision making and planning of the optimal path could be seen as a grid world problem with relatively independent states but with a continuous action selection process. The Q table allowed the agent to choose the next action. And the updated action caused a change in the state of the environment. This let the environment feed the agent with a completely new Q-table. After this, the agent made a decision about the next action in the updated environment based on the new Q-table. Throughout this process, the four components of Q learning mentioned above always influenced each other.

A classical flowchart of Q-learning was presented below in Figure 3.4:

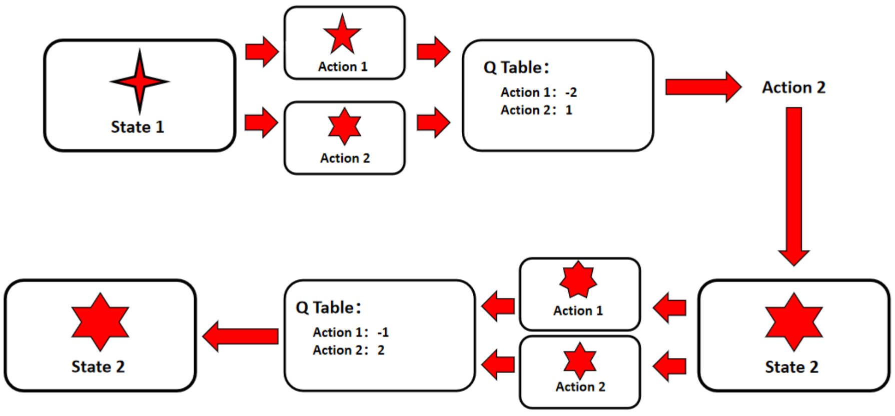


Figure 3.4: The classical flowchart of Q-learning

The algorithm used for a complete Q learning was based on an iterative process of values for the Bellman equation. This was not only influenced by state and action, but also by other parameters that affect the performances of Q learning model. The weighted average of the Q values of the previous state and the Q values of the new state was used. The specific expression equation was:

$$Q^{new}(s_t, a_t) \Leftarrow Q^{old}(s_t, a_t) + \alpha \cdot [r_t + \gamma + \max Q(s_{t+1}, a) - Q^{old}(s_t, a_t)]$$

Where s_t meant the agent's state at time t, a_t meant the agent's action at time t. $Q^{new}(s_t, a_t)$ meant the new Q table at time t. $Q^{old}(s_t, a_t)$ meant the old Q table at time t. α meant the learning rate $(0 \leq \alpha \geq 1)$,which represented the efficiency of the impact of each step state and action change as it was transferred to the next step. r_t meant the reward observed at time t. γ meant the discount factor $(0 \leq \gamma \geq 1)$, which represented the extent to which the value of the back-end reward was less declining than the value of the front-end reward. γ was also often used to show how well a start was made and the probability of success at each subsequent step. $\max Q(s_{t+1}, a)$ meant the maximum reward Q that could be obtained from the next state (time t+1) in the ideal case.

Analysis of the formula showed that the new Q table for a particular state$Q^{new}(s_t, a_t)$was influenced by a mixture of three components below:

- $Q^{old}(s_t, a_t) - \alpha \cdot Q^{old}(s_t, a_t)$: This part showed the current true value of the Q table under the weighted influence of the learning rate α. If the learning rate α was larger, the current value of the Q table had less influence on the new state value of the Q-table. In terms of running speed the larger the learning rate α, the changes in the Q-table would be more rapid.

- $\alpha \cdot r_t$: This part meant the real reward influenced by the learning rate α after making the action a_t at time t.

- $\alpha \cdot \gamma \cdot \max Q(s_{t+1}, a)$: This part showed the maximum reward value of Q obtained at the next time step $t + 1$ (The learning rateα and the discount factor γ affect the reward).

In a complete process, the starting Q was initialized to a fixed value and for the final time t_end, the $Q(s_end, a)$ at this time was usually never updated and was taken to be 0 in most cases.

It was also important to emphasise that in this project greedy strategies were used to seek to maximise the rewards of different actions for the current state. A greedy strategy was a strategy that determined how likely the current sampling was to be based on the Q value generated by the current training network. The purpose of this decision was to ensure that the agent explores as many possible grid positions as possible and did not keep going with the solution that had the highest success rate. According to the greedy strategy, there was initially an initial value of epsilon, which then decayed at a certain rate depending on the number of iterations. As the number of iterations increased, the epsilon decreased. The agent could explore as many regions as possible in a short period of time and had a better training effect because of the large epsilon at the beginning of the training period. Later on, as the epsilon decreased, the effect of the greedy strategy diminished. The parameters associated with the greedy strategy were the initial value of epsilon ϵ, the decay rate, the maximum iteration period and a limit on the overall number of agent steps.

As the focus of this project was to explore the interpretability of XAI algorithms to black-box reinforcement learners, the efficiency of reinforcement learning was not the most important aspect to observe. Therefore, in this project, the learning rate α, the discount factor γ, the initial epsilon ϵ, the decay rate of epsilon, the maximum iteration period and the maximum steps of the agent were set as default values. The specific values were shown in Table 3.4.

Parameter	Number
α	0.1
γ	0.9
ϵ	1
$Decayrateofepsilon$	0.001
$Maximumnumberofepisodes$	5000000
$Maximumnumberofsteps$	1000

Table 3.4: Q-learning parameter settings for Frozen Lake

Followed by the greedy strategy and the Q-learning algorithm, the basic algorithmic flow of the agent action for the Frozen Lake part of this project was as follows in Figure 3.5.The coordinates shown were the exact location of the agent in the Frozen Lake environment.

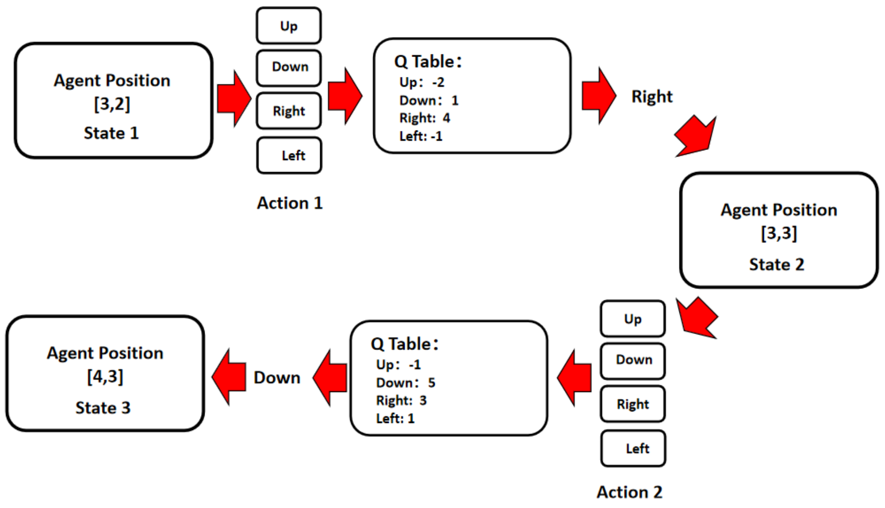


Figure 3.5: The basic algorithmic flow of Q-learning in Frozen Lake

The Q-learning used in this project was based on dynamic planning for value-based reinforcement learning, as could be seen from the process analysis above. However, Q-learning alone was not very efficient at showing the effects of reinforcement learning. Therefore, the Deep Q Network DQN algorithm, which was a combination of the neural networks, had been applied to Frozen Lake's path planning reinforcement learning.

3.2.2.2 Deep Q Network

The so-called DQN algorithm added the process of neural networks to the Q-learning algorithm in order to handle more complex situations. Instead of Q-tables, neural networks in DQN stored data and performed faster searches on the stored data. The entire reinforcement learning process was unstable and diffuse because of the use of a non-linear function approximator, the neural network, to represent Q. Small updates to Q could cause more significant changes in data distribution and agent planning strategies during this diffuse learning process. This feature also allowed the DQN algorithm to have better sensitivity and better performance. The biggest improvements of DQN over regular Q-learning could be summarised in two points:

- Neural networks with experience replay replaced the Q tables.Neural networks had a recursive hierarchy. In the process of passing from layer to layer, a portion of the data from each layer was passed as a sample to the next layer as a new target value to update the overall weights. The combination of target policy and behaviour policy could be used to take advantage of the off-policy model to plan better paths.

- The Q target was fixed by the neural network. The DQN algorithm had two similar sets of neural nets, one for predicting Q estimation and one for predicting Q target. The new parameters generated by the Q estimation neural network after training were passed into the Q target neural network. This process was repeated throughout the DQN algorithm. This resulted in a relatively stable Q target neural network. A more stable Q target neural network also made the whole agent learning process easier to fit. By doing so, the DQN algorithm also made it easier for the agent to learn the best plan.

Because of the introduction of the above-mentioned neural network structure with empirical playback and a fixed Q Target, the formula of the new DQN algorithm had a corresponding change compared to the formula of Q-learning. The weights w were added, resulting in the following new formula:

$$Q^{new}(s_t, a_t, w) \Leftarrow Q^{old}(s_t, a_t, w) + \alpha \cdot [r_t + \gamma + \max Q(s_{t+1}, a, w) - Q^{old}(s_t, a_t, w)]$$

Where $\triangle\ w$:

$$\triangle\ w = \alpha \cdot (r_t + \gamma + \max Q(s_{t+1}, a, w) - Q(s_t, a_t, w)) \cdot \nabla_w \cdot Q(s_t, a_t, w)$$

In order to make the working process of the DQN in this project more clearly presented, the flow of the DQN algorithm applied in the Frozen Lake part of this project was as follows in Figure 3.6.

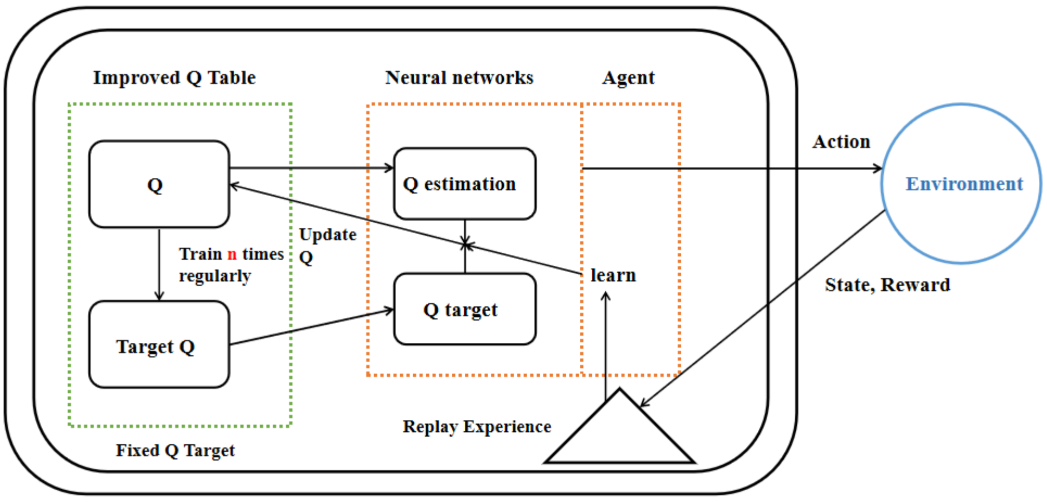


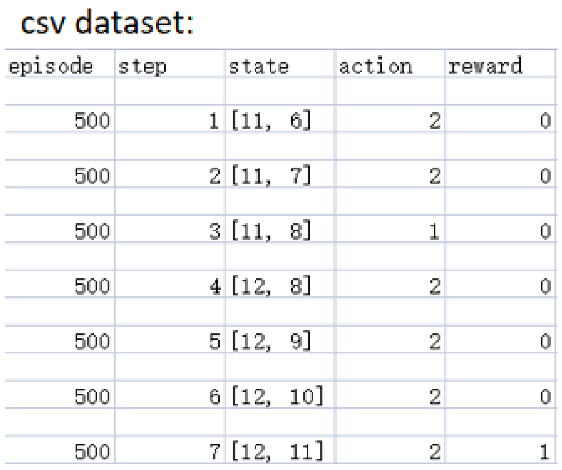
Figure 3.6: The DQN algorithmic flow for the project

3.2.2.3 Data generation and Data Cleaning

In this project, the DQN followed the specific steps mentioned above to train the agent in the Frozen Lake environment for reinforcement learning. During the process of reinforcement learning, the agent could generate the best plan after several iterations. The final optimal plan in this problem was a specific path that allowed the agent to successfully get from the start point to the goal point. This path could be seen as a trajectory on a visual map. In order for the XAI algorithm to have sufficient data to train its model afterwards, the starting point of the Frozen Lake environment in this project was not fixed, as described in the previous section. It would be set sequentially to all the Frozen Space ('F') in the Frozen Lake environment. According to such a plan, when the complete reinforcement learning phase was over, the agent would be given an exclusive optimal path, also known as a trajectory, in the case of each starting point. Specifically, in a 12x12 map with 20 holes, agent would end up with 123 trajectories.

Since this project investigated rule-based XAI algorithms, Only trajectories alone did not uncover the relevant rules. So the data corresponding to the relevant data during the training of the agent by the DQN algorithm needed to be attended to. Firstly, in the grid world of Frozen Lake, the reinforcement learning DQN algorithm took the state number of each step which the agent was in, and the number of the four actions, Up, Down, Left and Right as the action. So the (s, a) to which the agent belonged in each step was represented in this environment as (state number, action number). According to the basic principle of rule learning, state was often taken as the decision of the rule and action was often taken as the prediction by the rule. If the one-dimensional state number was used as the decision of the rule, the rule would not perform well in terms of accuracy. Therefore, the grid world of Frozen Lake was added to the 2D X-Y coordinate system. The corresponding state number became a 2-dimensional coordinate shaped as (x, y). This way the rule could be determined by focusing on both x and y. In the end, rule learning would be more effective.

As mentioned above, after DQN reinforcement learning, the agent could obtain the best path trajectory with all possible initial points. And each trajectory in the grid world was connected by a specific grid location. Combined with the above description of the DQN generated (state, action) plan at each grid location, each trajectory could be seen as a combination of the (state, action) dataset contained by the agent at each step. The state was represented by 2D coordinates and the action was represented by an action number. Each trajectory was thus a sequential record of all (coordinate position, action numbers) under the current trajectory. In order to check the accuracy of the entire agent activity, the number of iterations, the number of steps and the current reward value for each step were also counted in the trajectory. After these operations, the final trajectory data was stored in separate csv files and npy files for later use in the XAI algorithm. Figure 3.7 showed an example of the form in which the data within any track was stored.


csv dataset:

episode	step	state	action	reward
500	1	[11, 6]	2	0
500	2	[11, 7]	2	0
500	3	[11, 8]	1	0
500	4	[12, 8]	2	0
500	5	[12, 9]	2	0
500	6	[12, 10]	2	0
500	7	[12, 11]	2	1

Figure 3.7: The data in a simple csv file for the 12x12 map

Analysis of the generated datasets showed that some of the same datasets (states, actions) had some overly repetitive data as well as some incorrect data. The cleaned dataset, by cleaning these incorrect or overlapping data, was more representative of the agent's movement. In the next section of the XAI algorithm, the cleaned dataset were used to train the XAI algorithm model.

3.2.3 XAI Explanation Process of Frozen Lake

In the improved Frozen Lake environment, the agent was constantly exploring the path. The state of each grid location during exploration was discrete, but the process was continuous. The rule-based models employed for this situation were usually divided into classifiers and regressors.To make this project more comprehensive, a rule-based classifier, the Greedy Rule List classifier, and a imporved classifier, CN2 Rule induction algorithm, as well as a fit model combined with linear regression, Rulefit were used in this project. Theses three different models attempted to explain the reinforcement learning process of the DQN process in Frozen Lake. The results of these algorithms were also analysed by the comparative analysis to derive the most suitable rule learner for this type of problem. Specific explanations of these three algorithms were given in the following two sections.

3.2.3.1 Greedy Rule List Classifier algorithm

The Greedy Rule List Classifier was a special kind of classifier based on CART approach and selected according to a greedy strategy. It was further adapted from imodel package source code on Github. [65] The basic principle followed a greedy strategy, where the whole process was to 'greedily' segment one feature at a time along a single path and eventually generalise to the corresponding rule list. For the Greedy Rule List, the input data was an array of x-axis coordinates and y-axis coordinates from the reinforcement learning dataset provided in the Frozen Lake environment. The label predicted by the Greedy Rule List model was the specific number of the action.

This classification algorithm used the BaseEstimator and ClassifierMixin of the sklearn base class, the so-called BaseEstimator, which was an estimator. It provided the Greedy Rule List model with the ability to obtain parameters and adjust them. The $fit()$ function set the internal parameters of the Greedy Rule List and received the two key parameters of the training dataset and the corresponding class. Another function, $predict()$, was used to predict the category of the test set. The corresponding parameter to be processed was the testset. In this project, the data received and output via BaseEstimator was in the form of numpy arrays. The ClassifierMixin was used to classify multi-labelled data. In the Frozen Lake experiments of this project, there were four different labels presented the actions to be the predictions of the corresponding state. The ClassifierMixin could evaluate the accuracy against these four labels. Its score function could produce an average accuracy for a given test data and label to help the classifier better classify the data.

With these two basic models in sklearn, the basic framework of the Greedy Rule List was constructed. In this classification model, the decisions could be based on the information entropy, the Gini index and the positive and negative binary partition variables. As the decision classification in this project was multi-labelled, the binary partitioning variable was not applicable. The Gini index was similar to the information entropy and did not involve too many exponential operations. The Gini index could make the model run faster while ensuring certain effects were presented. In this project, the Gini index was used as the index for classifying the best features in the Greedy Rule List. The Gini index represented the impurity of the model, reflecting the probability that two randomly selected samples from the dataset would have inconsistent feature labels. Therefore, smaller the Gini index was, higher the purity of the dataset was. The Gini index favoured features with more feature values in decision making. This was similar to information gain. It took values in the range $0 \leq Gini \leq 1$. The formula below showed a concrete representation of the Gini index.

$$Gini(D) = \sum_{k=1}^{K} \frac{\mid C_k \mid}{\mid D \mid}(1 - \frac{\mid C_k \mid}{\mid D \mid}) = 1 - \sum_{k=1}^{K} (\frac{\mid C_k \mid}{\mid D \mid})^2$$

$$Gini(D \mid A) = \sum_{i=1}^{n} \frac{\mid D_i \mid}{\mid D \mid} Gini(D_i)$$

Where K meant the features.

As the basis of the calculated Gini index, the Greedy Rule List algorithm selected the segmentation point with the best criterion value for node segmentation according to a greedy strategy. The process was based on a maximisation strategy, where the feature with the highest probability of occurrence was used as the outcome of the decision. The segmentation point was used as the critical condition for the decision. If this condition was exceeded, the feature with the highest probability of occurrence was used as the type of action. The number of data that meet this rule was also counted. The new dataset after the exclusion of the data matching the rule was used for the next node splitting and rule learning. Through a similar process layer by layer, all the data would eventually find the corresponding rules and actions according to the specific classification. It was important to note that the whole process introduced Gini index and node splitting, so it was not advisable to simply use the respective integer numbers for different action numbers. Throughout the decision-making process, the average of the action numbers was used to select the best criterion segmentation point. The average of the feature values with the highest probability at each split node was then rounded off. The integer after processing was then used as the specific action determined after rule learning. Although rounding did not guarantee correctness in all cases, it did give a more correct representation of a macroscopic action selection. The hyper parameters in this Greedy Rule List model were displayed in Table 3.5.

Hyper parameters	Setting
Maximum numbers of rules	5
Criterion	Gini index
Strategy	Max

Table 3.5: The hyper parameters in the Greedy Rule List Classifier

After testing, the maximum number of rules required to generate the Greedy Rule List for this project was set to 5, which was a suitable value. This value did not increase the workload of the model but at the same time ensured that all rules were discovered. As mentioned above, the Gini index was introduced as the Criterion, and the model strategy was based on the greedy principle of choosing the "Max" strategy, i.e. the side with the higher classification risk as the choice of the leaf node splitting.

After going through the code for node splitting as described above, each data sample was assigned the corresponding rule. The entire flowchart of rule generation was shown in Figure 3.8.

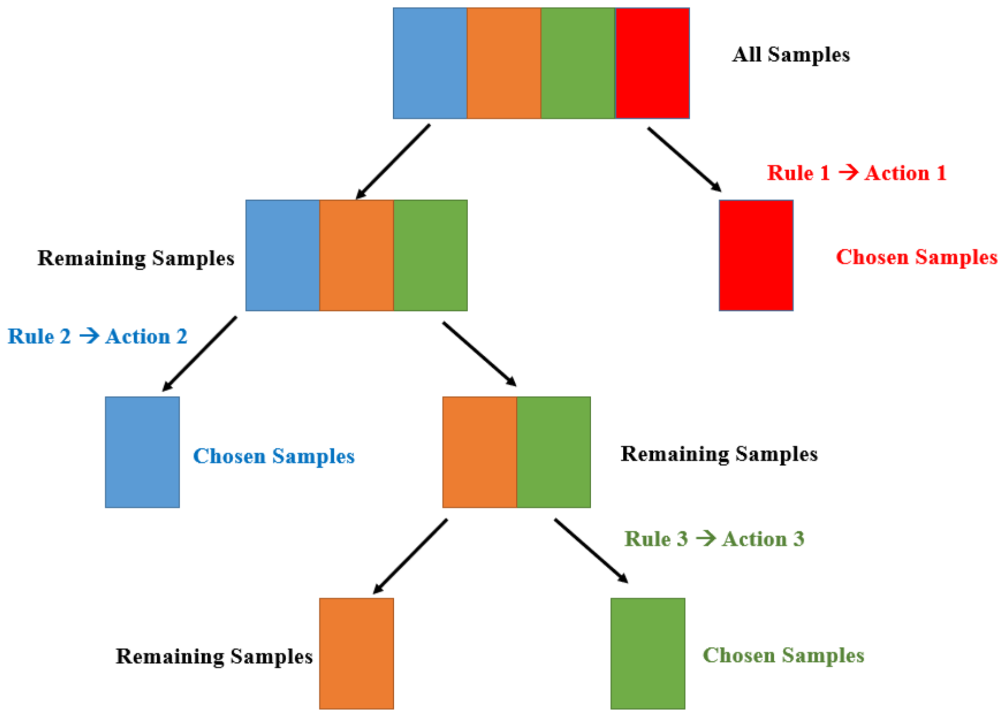


Figure 3.8: The flowchart of rule generation in the Greedy Rule List

By following the algorithmic process described above, rules encompassing all samples were generated.

Through further visualisation, the final rules were output as the form of Algorithm 3.1.

```
Data: Dataset ($State\ position(x, y), Action\ number$)
$k$ means the mean value of all feature, $j$ means the number of data
Mean value $k$ ($j$ pts)
$x/y$ means the specific coordinates in state, $n$ means the value splitting point
if $m \geq x/y \geq n$ then
    Then $a$ ($i$ pts)                    // $i$ means the number of the chosen data
    Repeat the above steps until all data is divided within the rule.
end
Data: Output: The rule list based on Greedy policy
```

Algorithm 3.1: Th rule form of Greedy Rule List

The Greedy Rule List was a classifier to generate single-feature rules. Its predicted accuracy for the overall data was not very satisfactory because of the limited number of split leaf nodes and the strategy of 'max' ,which just split one feature at one time, but it was still effective for predicting the state of a relatively independent position. Another classifier model, CN2 Rule Induction algorithm, was also used in this project in order to provide a more comprehensive algorithmic attempt at XAI algorithms for reinforcement learning in the Frozen Lake environment.

3.2.3.2 CN2 Rule Induction algorithm

The so-called CN2 induction algorithm is a rule induction learning algorithm, based on the ideas of the Algorithm quasi-optimnal(AQ) learning and the ID3 algorithm, which can handle noisy and messy data sets and obtain rule sets similar to the AQ algorithm. [12] The first step in understanding the CN2 induction algorithm is to understand the AQ algorithm and the ID3 algorithm.

The ID3 algorithm is a method invented by Ross Quinlan for deriving relative decision trees from datasets. [60] The algorithm performs leaf node splitting based on the calculated information entropy of each feature in the dataset and the corresponding information gain. It then classifies each sample in the dataset using the leaf nodes that are continuously split during the iterative process. Nowadays, the ID3 algorithm is widely used in machine learning and natural language processing, and the information entropy and corresponding information gain used in the ID3 algorithm for classification are calculated as follows.

$$H(s) = \sum_{x \subseteq X} -p(x) \log_2 p(x)$$

Where $H(s)$ means the measure of the amount of uncertainty in the dataset for a feature s. it also could be called as its information entropy value. X means the sets in S. $p(x)$ means the proportion of the number of elements in x.

$$IG(S, A) = H(S) - H(S \mid A)$$

Where $IG(S, A)$ means the information gain of S under the condition A. $H(S)$ means the entropy of feature S. $H(S \mid A)$ means the entropy of feature S under the condition A. It also could be called as the conditional entropy of feature S.

Followed by the above idea, the ID3 algorithm can generate into decision trees from the dataset. But the CN2 algorithm covers more than just this algorithm. Its more important ideas are derived from the AQ algorithm. The AQ algorithm is the basic rule learning algorithm in machine learning. Its job is to find consistent and complete rules in a dataset. [9] The core part of its work is to divide the dataset into positive and negative examples and find a formula that holds consistently from all the positive examples. The workflow of AQ algorithm is illustrated in Figure3.9.

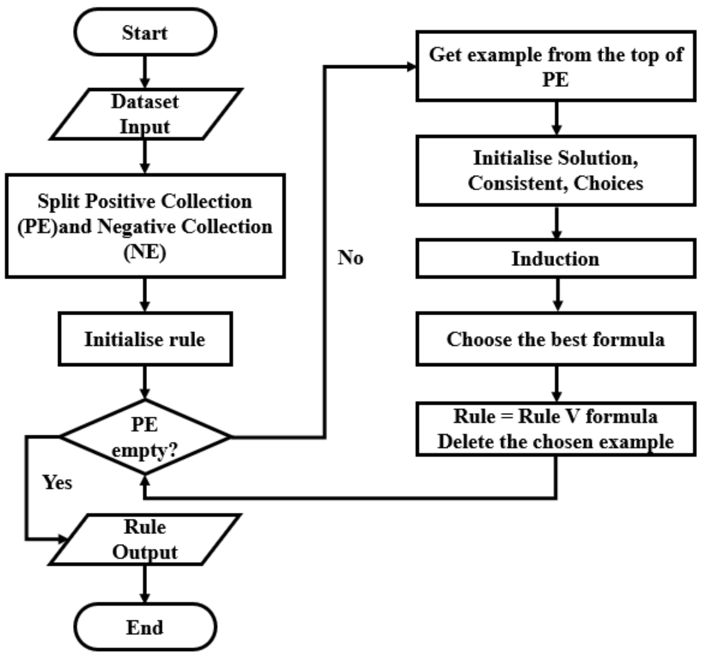

Start
Dataset Input
Split Positive Collection (PE)and Negative Collection (NE)
Initialise rule
PE empty?
Yes
No
Rule Output
End
Get example from the top of PE
Initialise Solution, Consistent, Choices
Induction
Choose the best formula
Rule = Rule V formula
Delete the chosen example

Figure 3.9: The flowchart of the AQ algorithm

The induction link in the AQ algorithm is an important factor in the ability of the AQ algorithm to produce rules. In the induction session, the formula undergoes a series of determinations before the final rule can be derived. The entire formula determination process is shown in Figure 3.10.

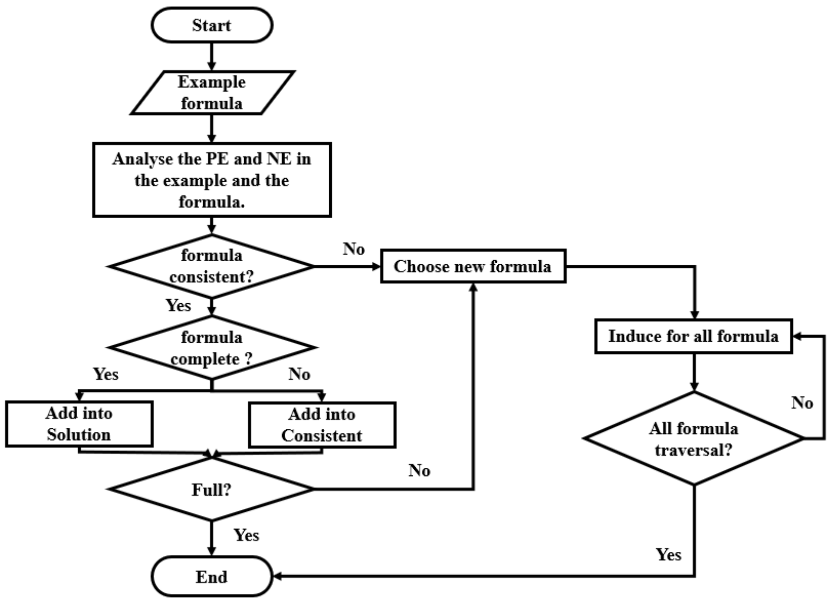


Figure 3.10: The working process of the AQ algorithm Induction part

Combining the advantages of the ID3 decision tree algorithm and the AQ algorithm above, the CN2 algorithm can be used as a decision tree to sense rules for the dataset according to the information entropy as a determination method. The CN2 algorithm model in this project was based on the CN2 classifier posted on Github by Alessia Mondolo and adapted for this project. [49] The model's hyperparameters were also set by reference to Mondolo's default values for the given model.

The CN2 algorithm was based on the AQ algorithm and the working principle of the ID3 decision tree. The main parts involved in the operation of the algorithm model were presented in workflow order.

1. Firstly, the conditions for determining the rules were related to the different determiners. The specific determiner was a feature and its corresponding value (for example, gender = male).

2. Secondly, unlike the greedy strategy of Greedy Rule List, CN2 had a mixed decision in the decision of individual rules. Mixed adjudication meant that there may be more than one adjudicator in a rule (for example, gender = male AND age = old).

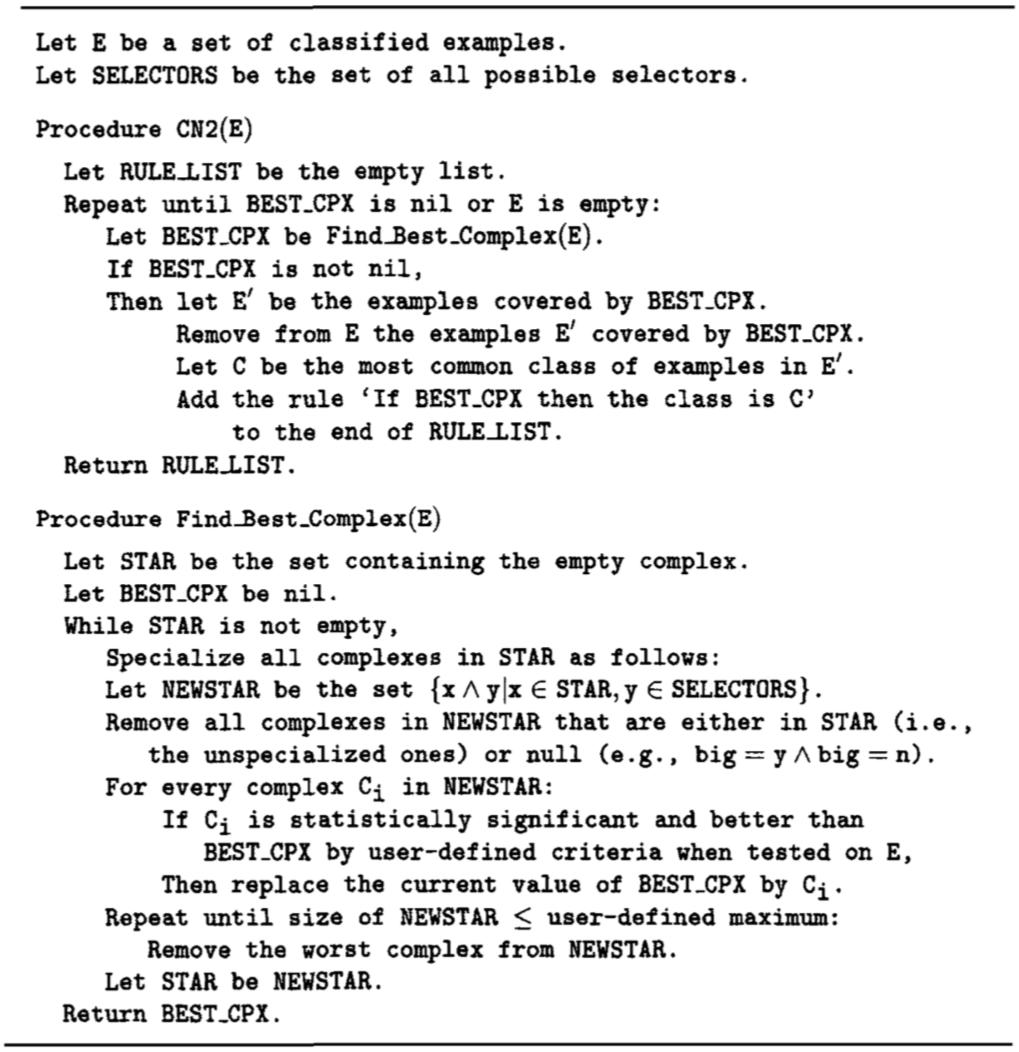
3. The CN2 algorithm used a heuristic search algorithm to continually search for the best combination of determiners. The search algorithm used could be adjusted by the user to the best K complex found, and the CN2 algorithm used the ID3 decision tree method of information entropy as a calibrator for the selection process.

4. After the selection of the complex, significance was applied to the evaluation of the portfolio as another enlightening algorithm. The higher the significance, the stronger the association between attributes and corresponding values between the combinations. Significance was expressed as a likelihood statistic. It was expressed by the following formula:

   $$Significance = 2\sum_{i=1}^{n} f_i \log(f_i/e_i)$$

   Where f_i was the frequency distribution of the observed instances that satisfy the given class. e_i was the expected distribution of the same number of instances given a random selection. The lower the value of the final significance the less relevant the law was proven to be.

After an introduction to the important parts of the CN2 algorithm workflow, a specific description of the CN2 algorithm could be presented. Figure 3.11 below showed the initial CN2 algorithm as first proposed by Clark and then by Nibblet.



```
Let E be a set of classified examples.
Let SELECTORS be the set of all possible selectors.

Procedure CN2(E)
  Let RULE_LIST be the empty list.
  Repeat until BEST_CPX is nil or E is empty:
     Let BEST_CPX be Find_Best_Complex(E).
     If BEST_CPX is not nil,
     Then let E' be the examples covered by BEST_CPX.
          Remove from E the examples E' covered by BEST_CPX.
          Let C be the most common class of examples in E'.
          Add the rule 'If BEST_CPX then the class is C'
              to the end of RULE_LIST.
  Return RULE_LIST.

Procedure Find_Best_Complex(E)
  Let STAR be the set containing the empty complex.
  Let BEST_CPX be nil.
  While STAR is not empty,
     Specialize all complexes in STAR as follows:
     Let NEWSTAR be the set $\{x \wedge y | x \in STAR, y \in SELECTORS\}$.
     Remove all complexes in NEWSTAR that are either in STAR (i.e.,
        the unspecialized ones) or null (e.g., $big = y \wedge big = n$).
     For every complex $C_i$ in NEWSTAR:
         If $C_i$ is statistically significant and better than
            BEST_CPX by user-defined criteria when tested on E,
         Then replace the current value of BEST_CPX by $C_i$.
     Repeat until size of NEWSTAR $\leq$ user-defined maximum:
        Remove the worst complex from NEWSTAR.
     Let STAR be NEWSTAR.
  Return BEST_CPX.
```

Figure 3.11: The algorithm process of the original CN2 algorithm model [12]

Followed by the above theoretical underpinnings, the CN2 algorithm would be applied to the dataset obtained after black-box reinforcement learning in the Frozen Lake environment. The related hyper parameters would be set as the default values. The details would be displayed in Table 3.6

Hyper parameter	Value
Maximum Star Size	5
Minimum Significance	0.5

Table 3.6: The hyper parameters for CN2 algorithm model

From Table 3.6, Maximum Star Size meant the maximum feature number in a rule. 5 could meet the requirement in two environments of this project. And the Minimum Significance meant the allowed minimum significance. These hyper parameters were suitable for this project.

As the CN2 algorithm was a classification XAI method for the classification of features. Each feature was supposed to be a specific string of a fixed category. The two-dimensional array used in the previous Greedy Rule List could not be used in the CN2 algorithm. Therefore, the initial dataset needed to be adapted before the model could be trained with the CN2 algorithm. The array representing the two-dimensional coordinates would be split into two features, namely the X-axis coordinates and the Y-axis coordinates. At the same time, the action numbers in the original dataset were converted into specific action types in order to improve interpretability and visualisation. As the 12x12 map did not contain many locations, the x-axis and y-axis coordinates did not need to be further classified in this problem. At the same time, as this project analysed all dynamic systems. In dynamic systems, security was of paramount importance. Only using the agent's state and action quantities in a dynamic environment as data allowed the rules generated later to be directly related to the properties of the agents themselves. This also helped humans to understand the rules directly and to utilise them in subsequent dynamic systems. Based on this principle, derived features that were based on initial data but were more difficult to interpret were not necessary to add into the dataset. It was the most initial data that guaranteed the interpretability of the XAI algorithm in dynamic systems and the security of the corresponding interpretability.The dataset after a series of processes was shown in table reftable:14. The transformed dataset was split into training set at 70% for later training. The remaining set was the test sets.

x_{axis}	y_{axis}	Action
11	10	Right
12	10	Up
9	4	Right
11	10	Right

Table 3.7: The example of the Frozen Lake transformed datasets for CN2 algorithm

Also, unlike the rules generated by the other two algorithms, the rules generated by CN2 were for specific feature types and do not compare values. The presentation of its final rules Pointed to a specific feature types. Examples of specific rules are represented in Algorithm 3.2.

```
Data: Input: Dataset ($x_{axis}$, $y_{axis}$,Action)
$m$ and $n$ mean the specific coordinates
if $x_{axis}$ is $m$ and/or $y_{axis}$ is $n$ then
    Then a                                          // a means the predicted action
    Repeat the above steps until all data is divided within the rule
end
Data: Output: The rule list and the accuracies of every rules
```

Algorithm 3.2: The rule form of CN2 algorithm

After applying the two classifiers, the Rulefit algorithm combined with the linear regression model, Lasso model would also be applied. Finally a comparison of the three would also show the difference in the impact of the classification task and the regression task on interpretable rule learning.

3.2.3.3 Rule Fit Regressor algorithm

The Rulefit algorithm used in this project was proposed by Friedman and Popescu in 2008. [22] The Rulefit algorithm used in this project was proposed by Friedman and Popescu and based on the Imodel package on GitHub. [65] It differed from a simple decision tree and from a simple linear regression model. It was an innovative combination of the two. By first classifying by some decision trees and then fitting the rules in a regression model, it solved a more complex problem and brought a higher accuracy. Specifically, the Rulefit algorithmic model brought together the rules obtained from multiple decision trees in a generalised additive model. The representative rules obtained in the multiple decision trees were combined with the original feature variables to form new variables, and the new rules formed by training the generalised additive model kept the interpretability of the rules formed in the decision trees and the ease of understanding of the generalised model. The sparse linear model used in Rulefit was the L1 regularised linear model, also known as the Lasso model. The final results by Rulefit algorithm showed higher accuracy and better performances than a generalised decision tree because the Lasso model could remove some duplicate rules by weights.

For data with multiple features in the dataset, the Rulefit algorithm model worked by first training multiple decision tree models and generating multiple initial rules from these multiple decision tree models. The rules were then fitted as variables into a linear regression model. The formula could be expressed as:

$$F(x) = a_0 + \sum_{k=1}^{K} a_k r_k(x)$$

Where r meant the different rules, a means the correspondence coefficients. K meant the total numbers of the different rules.

The estimate of the penalty factor coefficient in the above equation was obtained by minimising the loss function containing the penalty term. The specific formula was:

$$(\widehat{a}_k)_0^K = arg\min \sum_{i=1}^{N} L(y_i, a_0 + \sum_{k=1}^{K} a_k r_k(x_i)) + \lambda \cdot \sum_{k=1}^{K} \mid a_k \mid$$

By using the above formula, the exact workflow of Rulefit could also be summarised. The first step of the Rulefit algorithm was to train n decision trees. These decision trees were used as base learners to generate the initial rules. The second step was to introduce the Lasso linear model. In this linear model the previously generated rules were combined with the corresponding penalty coefficients to form new variables. For the new variables, the model was linearly fitted to them. The accuracy of the rules after the linear fit was further improved. Based on the process described above, a flowchart of how the Rulefit model works was shown on Figure 3.12 .

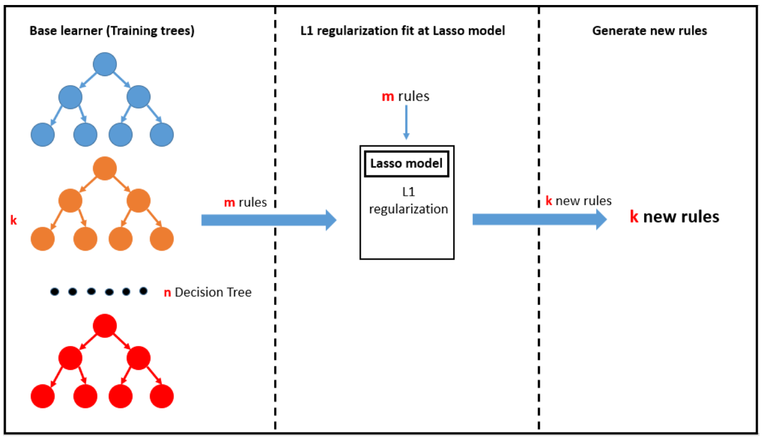


Figure 3.12: The flowchart of the Rulefit model

The first step in the workflow was to use multiple decision trees based on information entropy to generate the initial rules. The specific rules were the classification conditions implied by the root endpoint to each leaf node point on the decision trees. The second step was to add the initial rules as variables to the Lasso linear model to create new variables. This process allowed for the deletion of some of the rules that were covered in the first step by weighting them. This enhanced the final interpretability and predictive accuracy of the overall model.

As the basis of the above process, the Rulefit algorithm could be trained on the current dataset in a step-by-step manner. The Rulefit algorithm specified that the input dataset must be in the form of an array containing only values, and the output predictions in the form of a corresponding label. The dataset generated in the Frozen Lake environment fulfilled this requirement. The x-axis and y-axis coordinates in the 2D space representing the state could be formed into arrays, while the specific numbers of the actions could be used as labels. Similarly, because of security considerations for the XAI algorithm, the features employed in the Rulefit algorithm are the agent's own attributes in the environment. Additional derived features are not applicable in this project.

The Rulefit algorithm model allowed for the prediction of features after the entire workflow as described above. In this project, this was the prediction of the type of action. At the same time, the rules about the dataset were grouped into lists. In this project the rules would focus on the constraints of the two-dimensional coordinates of the locations. Some specific indicators of the rules were also presented in the list. The penalty coefficients for each rule, the proportion of the total sample contained in each rule (support) and the importance of each rule were all recorded in the list. One indicator that needed to be highlighted was the importance metric. It was a measure of the influence of each rule on the results, obtained by integrating the rule and feature variables. The higher the importance, the greater the influence of the corresponding rule on the overall model performance.The specific calculation process for importance was as follows:

- A standardised predictor finds the importance of the original features for the linear item.The formula is:

  $$I_j = \mid \widehat{\beta_j} \mid \cdot std(l_j(x_j))$$

  Where β_j means the weight of the Lasso model for the linear item and $std(l_j(x_j))$ means the standard deviation of data in the linear item.

- For the rule terms, the importance could calculated with the formulas below:

  $$I_k = \mid \widehat{\alpha_k} \mid \cdot \sqrt{s_k(1 - s_k)}$$

  $$s_k = \frac{1}{n} \sum_{i=1}^{n} r_k(x^(i))$$

  Where r_k means the rule. s_k means the support of the corresponding rule, which show the percentage of data belongs to the corresponding rule. α_k means the weight of the Lasso model for the rule item.

- After combining the linear and rule terms, the importance of the composite for the independent predictions could be expressed by the following equation

  $$J_j(x) = I_j(x) + \sum_{x_j \subseteqq r_k} I_k(x)/m_k$$

  Where I_j means the importance of the linear item, I_k means the importance of the rule item when the x_j appears in r_k, m_k means the number of features constituting the rule r_k.

Although the rules generated by Rulefit included specific qualifications for specific features, they did not directly generate the corresponding predictions. To address this situation, in this project, the results would be predicted for specific rules. This was done by extracting all the samples belonging to a specific rule and averaging and rounding the results of these samples. The final integer result was the predicted outcome of the rule, which was represented in this project as the predicted action, and by combining the metrics and rules on the list and the predicted action of the corresponding rule, the strategy and learning direction of the agent in the black-box learning process was explained to some extent.

After the above plan, the final Rulefit presented the rule determination similar to that presented by the Greedy List Rule. Both were numerical comparisons of specific feature. The algorithm used by Rulefit to form the rules was shown in Algorithm 3.3. Using the resulting final rules and the corresponding overall predictions of the rules as well as the metrics of rule support and importance, the Rulefit algorithm allowed for a deeper interpretation of the overall black box strategy.

```
Data: Dataset ($State\ position(x, y), Action\ number$)
$x$ and $y$ mean the specific coordinates in state, $n_1$,$m_1$,$n_2$ and $m_2$ mean the rule condition values
if $m_1 \geq x \geq n_1$  and  (or)  $m_2 \geq y \geq n_2$ then
    Then $a$        // $a$ means the corresponding properties (Support, Importance and coefficient)
    Adding the overall predictive action of the rules
    Repeat the above steps until all data is divided within the rule.
end
Data: Output:  The rule list with corresponding properties based on Rulefit
```

Algorithm 3.3: Th rule list form of Rulefit

Many of the hyper parameters of the Rulefit model were adjustable. The hyper parameters of the Rulefit model used in this project were shown in Table 3.8.

Hyper parameters	Setting
prediction task	Classification
Number of estimators	100
Tree size	4
Maximum numbers of rules	2000
Memory parameter	0.01
liner trimming quantile	0.025
Exp-random tree size	True

Table 3.8: The hyper parameters in the Rulefit algorithm model

The specific parameters in the table above all played an important role in the final model. The prediction task was used to confirm whether the final task performed by Rulefit would be regression or classification. In this project, the task was chosen to be 'classification' as the ultimate aim was to classify specific states and thus predict actions. In the first step of Rulefit, multiple decision trees formed a forest-like structure, and the number of estimators represented the number of decision trees. In this project, the number of estimators was set to 100, i.e. there were 100 decision trees in the first step. The tree size determined the average final endpoint of the Rulefit decision tree. From another perspective, the tree size determined the average depth of the decision trees. In this project, the tree size was set to 4. Under this setting condition, the structure of the decision tree showed that the depth of the decision tree in this project was 2-3 levels. This parameter determined the number of conditions contained in the final rules. The more conditions there were in the same rule, the more complex the interpretation and the greater the probability of contradictory conditions. When using the Rulefit algorithm, 2-3 conditions per rule was generally the best choice. The maximum number of rules defined the final number of rules. As there were 100 decision tree generation rules, the number of rules generated would be relatively large. In order to save all the rules, the maximum number of rules was limited to be as large as possible. In this project, the maximum number of rules was set to 2000. The memory parameter was the shrinkage factor for each new tree in the sequential induction method. In this project, the memory parameter was set to 0.01, a default value. The liner trimming quantile was a quantile in the pre-normalisation construction process. in this project, this parameter was set to 0.025. in this project, the Exp-random tree size was set to True, which meant that each tree can have a different maximum number of nodes. The different number of nodes would ensure that the resulting rules were more comprehensive. Based on the above parameter settings, the Rulefit algorithm could be used to some extent to explain the black-box learning process of the agent.

3.2.3.4 Methods for evaluating the performance of algorithms

With the three different XAI algorithms described above, the black-box reinforcement learning process of the agent in the Frozen Lake environment was explained to some extent. However, how to evaluate the effectiveness of these two algorithms was a matter of both interpretability and model performance. At the beginning of this project, the principles of XAI technology were introduced, not as a 'replacement' for black-box reinforcement learning, but as an 'explainer' or 'explainable imitator'. Its training performance could not be 100 percentage the same as that of a black-box model. However, the aim of using explainable AI was to maintain a certain level of performance while some of the strategies or working processes of the black box model that could not be made intelligible explained to some extent. Based on this principle, the methods used in this project to evaluate all algorithms, Greedy Rule List, CN2 algorithm and Rulefit, took both aspects into account.

Firstly, all algorithms provided corresponding rule lists that could be understood by humans. The explanation of the action prediction method and the agent black-box reinforcement learning simulation process mined from the rule lists could be taken as a manifestation of interpretability. Since these explanations were intuitively understood by humans, it was difficult to use a scalar to judge their merit. However, the difference in effectiveness in terms of interpretability could also be seen through the accuracy of the rules and the amount of information presented.

Secondly, XAI algorithms needed to ensure that acceptable performance of the model still existed. Whereas the performance of black-box reinforcement learning was often determined by how well the agent ultimately performs in the environment. For testing the performance of the agent under the XAI model, this project took two approaches to validation.

- The first approach was often for general classification regression problems. Starting from a dataset, a portion of the dataset was used to train the XAI algorithm model. The other part of the dataset was used as a testset to test the model. With the data input from the testset, the trained XAI algorithm model could predict the corresponding actions and serve as the output data. These predicted actions were then compared to the real actions. The accuracy shown could be used as a measure of the performance of the XAI algorithm model.

- The second approach started with the performance of the agent in the environment. The XAI algorithm model was trained using the dataset. Then the trained XAI algorithm model was returned to the environment and used as a controller for the agent. The success rate of the agent in this way could be indirectly derived by counting the number of times the agent can finally reach the target point from the starting point under the XAI algorithm model in the Frozen Lake. The higher the success rate of the agent, the better the performance of the XAI algorithm model was proven to be.

The performance of the XAI algorithm model could also be visualised by the above two methods. The combined effect of the algorithm could be demonstrated by analysing the interpretability of the algorithm model and the performance of the retention.

3.3 The Program for Lunar Lander

The action of the agent in the Frozen Lake environment had the character of a grid world. Within each time step, the agent advanced one frame in one of four directions: Up, Down, Left and Right. This made the two-dimensional coordinates of the agent's position an integer in any given case, and the number of actions performed an integer. This type of movement made the task simple when processed by XAI algorithmic models. The simplicity of the task made it easier to demonstrate the effects of interpretability. In the real world, however, the motion of dynamic systems was often more complex. The 'agent' in a real dynamic system would often move in any direction and the pattern of movement would not follow a step-by-step pattern. Therefore, in order to bring this project closer to real-world dynamics, a more complex Box2D environment, Lunar Lander, from the Open AI Gym platform, was added to this project to investigate the XAI algorithm.

3.3.1 Environment Construction of Lunar Lander

Lunar Lander is a Box2D environment based on the open source Open AI Gym platform. The so-called Box2D environment is an open source physics-based game engine. It introduces motion and collisions with 2D rigid objects. A large number of physics mechanics and kinematics calculations are integrated in Box2D and the physics simulation is encapsulated in the object. By simply calling objects or functions that are revealed in the engine, real-life acceleration, deceleration, parabolic motion, gravity, collision bounce and various other realistic physics movements can be simulated. Based on this environment, Lunar Lander simulates the landing of a lunar lander on the moon. Through this environment, the various situations encountered during a lunar landing can be simulated to a certain extent.

Lunar Lander simulates the environment of a lander landing on the Moon, involving the different states of the lander after ignition of the different engines and the final outcome of the landing. The details of the environment build in this project were as follows:

- The Lunar Lander environment, based on the Open AI Gym platform, still used 2D coordinates to reflect the exact location of the lander. The lander's landing point was fixed at (0, 0). This was similar to the fixed target point of Frozen Lake, which was the subject of a previous part. The position of the lander at each moment would also be represented using 2D coordinates. The 2D coordinates at each moment were determined by the first two numbers in the lander's state vector.
- In order to measure the performance of the lander in the overall Lunar Lander environment, a bonus mechanism was introduced in this environment. The performance of the lander was determined by the reward score. Different states or results would result in a change (increase/decrease) in the total bonus score. When moving from the top of the environment to the landing point, the lander was rewarded with between 100 and 140 points depending on the landing speed (the lower the landing speed the higher the reward). If the lander did not eventually land at the landing site, the lander would not receive the landing bonus to which it was entitled. If the lander crashed on final landing because of rigid body motion, an additional 100 reward points would be deducted and the process would end. If the lander finally landed successfully, the lander would be awarded 100 points, regardless of whether it landed on the landing site or not. As for the lander itself, it would receive a reward of 10 points for each leg that touches the ground. As the objective of the test was to land the lander with the minimum amount of fuel, the use of fuel was limited. Each frame used by the lander's engine would cost the lander 0.3 reward points. A test in the Lunar Lander environment was considered successfully completed if the final lander ended the process with a total reward of 200 points.
- At each moment throughout the simulated lunar landing the lander would show the state of its position. Each moment of state contained eight different features, including (1) horizontal coordinates (i.e. x-axis coordinates) (2) vertical coordinates (i.e. y-axis coordinates) (3) horizontal velocity (i.e. velocity on the x-axis) (4) vertical velocity (i.e. velocity on the y-axis) (5) angle (6) angular velocity (7) contact of the first foot ('1' means contact, '0' means no contact) (8) Contact of the second foot (represented in the same way as the first foot). With these eight lander specific attributes, the state of the lander at each moment could be observed by the user.
- The tests in this environment were designed to allow the lander to learn how to fly in an intensive learning environment. Therefore, it was necessary to set certain rules for the lander's movements. Firstly it was permissible for the lander's manoeuvres to result in the lander landing outside the landing site. Secondly, the lander's fuel was by default unlimited. This ensured that the process did not end prematurely because of lack of fuel. Finally the lander had a choice of four discrete actions: (1) action number 0: no engine be fired (2) action number 1: fire left orientation engine (3) action number 2: fire right orientation engine (4) action number 3: Fire the main orientation engine.

With the environment set up as above and the physics laws of object movement that come with Box2D, an environment suitable for this project regarding the moon landing was built. Based on this environment, a black-box reinforcement learning model would be trained. The DQN algorithm would be used again during the training process to train the agent to learn to fly.

3.3.2 Reinforcement Learning Process of Lunar Lander

3.3.2.1 DQN process for Lunar Lander

For the Lunar Lander environment, the algorithm used for reinforcement learning in this project was still the DQN algorithm. This also ensured consistency in the use of reinforcement learning models in both environments. However, unlike the DQN model in Frozen Lake, the DQN model in Lunar Lander was more complex. This was followed by the complexity of the Lunar Lander environment itself.

The algorithm for Lunar Lander referred to Nikolai Karasov's model for RL on Github and was adapted for this project. [34] As the movement of the agent in the Lunar Lander environment was much more complex than in the previous Frozen Lake, its entire reinforcement learning model would have to handle a much larger task. Since a normal model would not be able to handle the Lunar Lander part of the task quickly enough. This project used the existing neural network model in the Tensor module to train the agent for reinforcement learning and transfer the variables to the 'cuda' gpu for computation if allowed. In this form, the reinforcement learning process of the agent in the Lunar Lander environment could be efficiently implemented. The basic flow of the DQN algorithm in the Lunar Lander environment for this project could be found in the flowchart above Figure 3.6.

Based on a process similar to the DQN algorithm model in the Frozen Lake section, the agent in the Lunar Lander environment could show better flight results after reinforcement learning training. The specific hyper parameters of the model used in this process were summarised in Table 3.9.

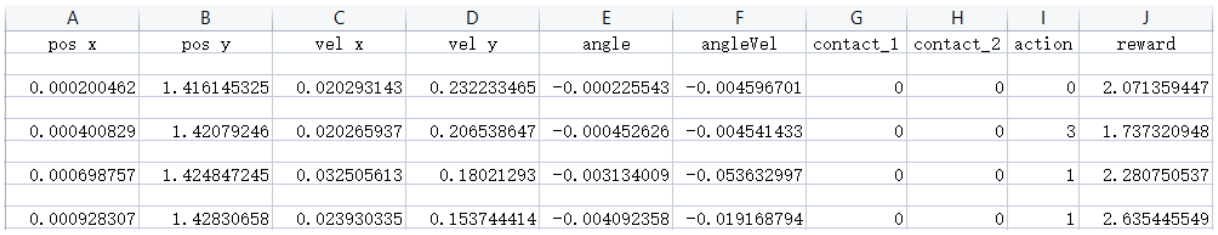
Hyper parameters	Setting
Discount Factor γ	0.99
Target update rate	1000
device	'cuda'
Learning rate	0.00003
Batch size	64
epsilon	1
Capacity	100000

Table 3.9: The hyper parameters in the DQN models for Lunar Lander

The discount factor γ was set to 0.99 in the model setting of the DQN algorithm for Lunar Lander, and a higher γ value meant that the agent was more future-focused. Although a discount factor value of 0.99 may cause the program to run relatively slowly, it was useful for solving complex Lunar Lander problems and improving the performance of the agent. The Target update rate was set to 1000 for this project, which had a positive effect on the tuning of the Target network. The Tensor service type was chosen to be a 'cuda' GPU, which allowed the model to have more computing power. The learning rate of the DQN model was set to 0.00003, a relatively low learning rate that was useful for handling the large and complex learning samples in this environment and ensured that the final results were effectively converged. The epsilon greedy policy was also used in Lunar Lander's DQN algorithm learning process. The initial value of epsilon was set to 1 as in the Frozen Lake environment, while the batch size was set to 64 in the Lunar Lander environment for the structure of the DQN network, which represented the amount of data used in each update using the optimiser. A value selection method of 2 to the nth power tended to make the updates work best. As could be seen from the introduction of the DQN algorithm above, the memory playback component was an important factor in the effectiveness of the DQN algorithm. During the Lunar Lander reinforcement learning process, the maximum memory storage was set to 100000, which could provide a large enough memory storage space for a large amount of complex data. After the above hyper parameters were set, the DQN algorithm model used in the Lunar Lander environment was structured to train the agent on the fly.

3.3.2.2 Data Generation

After training with the DQN algorithm model, the agent showed efficient flight performance in Lunar Lander. By selecting the final sets of flight trajectories with high reward, the data could be recorded. In this project, 11 trajectories which could get over 250 rewards were recorded as the output data. The environment architecture above showed that at each moment in time the lander had a state consisting of 8 features, the specific feature types were described in the environment architecture above. In this project, these 8 features were used as the data set to be trained. Similar to the Frozen Lake phase, the number of the upcoming action of the lander on each state was treated as the result of the dataset in the Lunar Lander environment. In summary, the final output of the black-box reinforcement learning in the Lunar Lander part was (state, action). where state was an array of 8 feature values and action was the specific numbered value corresponding to the action type. The (state, action) for all moments of the selected trajectory were pooled together to form a dataset for use in the XAI algorithm. The final rendered dataset would be represented in a csv file in the form shown in Figure 3.13.



A	B	C	D	E	F	G	H	I	J
pos x	pos y	vel x	vel y	angle	angleVel	contact_1	contact_2	action	reward
0.000200462	1.416145325	0.020293143	0.232233465	-0.000225543	-0.004596701	0	0	0	2.071359447
0.000400829	1.42079246	0.020265937	0.206538647	-0.000452626	-0.004541433	0	0	3	1.737320948
0.000698757	1.424847245	0.032505613	0.18021293	-0.003134009	-0.053632997	0	0	1	2.280750537
0.000928307	1.42830658	0.023930335	0.153744414	-0.004092358	-0.019168794	0	0	1	2.635445549

Figure 3.13: The dataset form of Lunar Lander output

3.3.3 XAI Explanation Process of Lunar Lander

3.3.3.1 XAI algorithms for Lunar Lander

Similar process to Frozen Lake's XAI algorithm were used in Lunar Lander part. For Lunar Lander's black-box reinforcement learning process, the GreedyRule List and CN2 algorithm as well as Rulefit were also used to provide some explanation of its black-box strategy. The use of the same algorithms in two different contexts demonstrated the generalisability of the XAI algorithm used. By comparing the performance of the algorithms in the two different environments, it was also possible to analyse the impact of simple versus complex environments on the XAI algorithms.

The specific implementation of the two XAI algorithms for Lunar Lander was similar to the process for Frozen Lake. However, because of Lunar Lander's larger data volume and more complex data types. The maximum number of rules was set to be larger to accommodate the more complex rule forms in the algorithm hyperparameters. Additional hyperparameters were also adjusted as necessary to accommodate the larger data volumes and more complex data forms.

As for the performance demonstration of the XAI algorithm, the methods used were similar to those used in the Frozen Lake process. One approach was still to perform a test set of accuracy measurements on the dataset. The other approach was to use the XAI algorithm as a controller to control the actions of the lander. Ultimately, the probability of the lander successfully landing under the control of the XAI algorithm reflects the lander's flight performance. If the lander performs better, the XAI algorithm model would perform better. the interpretability of the XAI algorithm could be determined by direct observation of the rules generalised to the XAI algorithm.

3.4 Summary of the Methodology and Materials

These were the specific models, algorithms and methodological strategies used in this project. Through these processes, the relevant results were documented. The corresponding results would be analysed in the following chapter. The hypothesis analysis of the reasons for the differences in the results between the different algorithms would be explained. For possible errors, the causes of the errors were also explained.

Chapter 4

Result Analysis

After the above three XAI algorithms had been applied to the Frozen Lake and Lunar Lander environments, the performances of the corresponding rules and individual algorithmic models were demonstrated. In this chapter, all results were presented and analysed. Also, differences in the effects of the different results, the analysis of the causes and the analysis of possible errors would be included in this chapter.

4.1 The result analysis for Frozen Lake

Each of the three XAI algorithms was used in the Frozen Lake environment to derive rules that could be understood by humans. The performances of these three algorithmic models were also measured by the two methods described in Chapter 3. In this section, the results of the Frozen Lake environment were presented one by one. Also, results from different algorithms would be compared and analysed.

4.1.1 The results from Greedy Rule List

The greedy rule list algorithm could generalise the corresponding rules from the dataset generated during the black box learning process. These rules could serve to explain the black box strategy to some extent. Meanwhile, the rules generated by Greedy Rule List after training on the dataset were presented in 4.1. As could be seen in Table 4.1, the rules generated by the Greedy Rule List conformed to its working principle of disaggregating one feature at a time and generating one rule at a time. All generated rules were qualified for only one of the x_{axis} and y_{axis}.

Rules of Greedy Rule List
If $x_{axis} \geqslant 12$, then Right
If $y_{axis} \geqslant 8$, then Right
If $x_{axis} \geqslant 7$, then Down
If $y_{axis} \geqslant 10$, then Down
If $y_{axis} \geqslant 11$, then Down

Table 4.1: The derived rules by Greedy Rule List for Frozen Lake

Without considering potential problems, the generated rules mostly conformed to the agent's motion policy. From the five sets of rules generated above, the agent moved to the right when the position was at the bottom boundary, i.e. $x_{axis} = 12$. The coordinates of the position of the target point were (12, 12). Then when $x_{axis} = 12$, the rule-controlled agent moved to the right in accordance with the motion policy. Similarly, when $y_{axis} = 12$, the rule-controlled agent moved to the down, which was also consistent with the motion policy.

However, there was also a very significant drawback to these rules. This class of rules was significantly less inductive as it contained only one feature. This results in specific rules that were often very broadly qualified. A very wide range of rules for a single feature could be very ambiguous for the interpretation of a black box policy. Also, as only one feature could be generalised at a time, the algorithm ignored the influence of another feature in the generalisation process. This could lead to errors in the final generalised rules.For example, when $x_{axis} \geqslant 7$, the rule gave a policy of moving downwards. This policy was suitable for $11 \geqslant x_{axis} \geqslant 7$, because downward action could make the agent closer to the target point when the agent in the range above. However, because the rule was over-qualified, continuing to move down when $x_{axis} = 12$ within the rule limits would cause the agent to leave the map. This caused an error to occur. Another example in the table was that the predicted action was Down when $x_{axis} \geqslant 12$. However, based on the map setting, the Down action would be executed when the agent exceeded the map boundaries and was stuck in place, leading to an error. Also from the rule itself, the inclusion of areas outside the map size in the rule itself reflected the inaccuracy of the rule because of the map size limitation. In summary, the rules generated by the Greedy Rule List were inaccurate and highly partial because of the greedy policy of the algorithm for a single feature. The human observation of these rules did not provide a clear and deep understanding of the agent's specific strategy in the black-box learning process.

The impact of the Greedy Rule List algorithm on the performance of the model was also shown after the analysis of the interpretable results. The two specific methods were described in the section of Chapter 3.(3.2.3.4) The performance of the Greedy Rule List model by the two methods was presented in Table 4.2.

	Performances of Greedy Rule List algorithm
Prediction accuracy	46%
Controlling performance	Fail

Table 4.2: The performances of Greedy Rule List for Frozen Lake

The results in Table 4.2 could again prove the previous conjecture. Because of the greedy policy of the Greedy Rule List's single feature, the Greedy Rule List presented a poor interpretability and did not provide a very accurate and intuitive explanation for the black box strategy. The Greedy Rule List's prediction accuracy was not very high when predicting the testset data. This demonstrated that in general, the rules generated by the Greedy Rule List did not make valid predictions about the agent's actions. This was contrary to the results achieved by black-box reinforcement learning. Therefore, it could be seen that the Greedy Rule List algorithm did not accurately explain the black-box policy in the Frozen Lake environment. A similar proof came from the controller's attempt. If the Greedy Rule List was used as the controller for the agent, the agent would get stuck at some point and thus the motion process got stuck in a dead loop, regardless of the model used in the dataset. The specific reason for the deadlock was consistent with the previous rule conjecture that the agent would come to the boundary of the map and kept executing commands to move out of the map. The performance results showed the same trend as the interpretability results, with both results demonstrating that the Greedy Rule List was not a valid and accurate explanation of black-box reinforcement learning in the Frozen Lake environment.

In summary, the Greedy Rule List did not provide an effective interpretation of black box strategies in the Frozen Lake environment because of the shortcomings of its own algorithm. In order to obtain a better interpretability, two other algorithms, the CN2 algorithm and the Rulefit algorithm, were applied in the Frozen Lake environment. The results were presented in the following sections.

4.1.2 The results from CN2 algorithm

The CN2 algorithm was a classifier though, like the Greedy Rule List algorithm. However, it was predicted to be better than the Greedy Rule List in terms of effectiveness. In this section, the performance of the CN2 algorithm in terms of interpretability and accuracy in terms of performance would be shown one by one.

Similarly, the dataset from DQN was applied to the CN2 algorithmic model. Its interpretability was expressed in terms of the obviousness of the rules generated. The generated rules, as well as the number of examples covered by the specific rules and the corresponding accuracy, were shown in Table 4.3 .Because of the excessive length of the list of rules, the complete list could not be shown in the text. Only a few representative examples were shown in the two tables. A specific and complete list could be found in Appendix A.1.

Rule	Coverage	Precision
If x=12 and y=2, then Up	2.63%	100%
If x=12 and y=3, then Right	0.39%	100%
If x=12 and y=6, then Up	5.26%	100%
If y=12 and x=9, then Left	20.4%	100%
If x=12 and y=11, then Right	10.04%	91.7%
If x=12 and y=10, then Right	7.28%	84.2%
If x=12, then Up	2.63%	50%
Default: Right	100%	46.1%

Table 4.3: The rule list of CN2 algorithm for Frozen Lake

As could be seen from the table above, the overall effect of the rules generated was acceptable. With the exception of the last few sets of rules, all the rules basically achieved 100% prediction. Since there were only two features in the Frozen Lake environment, the x-axis coordinates and the y-axis coordinates. And both coordinates have only 12 different values. This made it difficult to generate rules for specific ranges. During the project, the x-axis and y-axis coordinates were divided into top-middle-bottom and left-middle-right ranges. However, the CN2 algorithm was unable to generate effective rules for the dataset with this division. The final results were all in the form of Default. Therefore, in the end, only the specific values of the x-axis coordinates and the y-axis coordinates could be considered as different classifications. The table above listed the rules for different positions for the case $x_{axis} = 12$. For example, in the case of $x = 12$ *and* $y = 2$, the action made was to move up. This type of data represented 2.63% of the overall dataset and the rule predicted its action with 100% accuracy. The rest of the rules behaved similarly to the rules mentioned in the example. Combining these rules allowed the analysis of the agent's strategy for avoiding the hole. The map showed that the position (12, 7) was a hole, and all the previous positions with an x-coordinate of 12 had been chosen by the agent to move upwards rather than to the right according to the rules. This prevented the agent from falling into the hole (12, 7). Similarly (11, 9) was also a hole, and the rule at (12, 9) did not choose to move upwards according to the rule. This also prevented the agent from falling into the hole. In other cases, according to the single-feature rule with $x = 12$, the agent mostly acted to the right. This was also consistent with the optimal strategy of reaching the target point at (12, 12) the fastest.

However, because of the insufficient number of data and features in Frozen Lake, the final CN2 algorithm model was not able to generate rules for all locations. The presence of undecidable Defaults would be determined by uniformly performing the rightward action. The prediction accuracy of the rules for this Default case was only 46.1%. This made it possible for the agent under the control of the CN2 algorithm to make incorrect action decisions.

Using the same method as above (3.2.3.4), the performances of CN2 in the testset and when acting as a controller were shown in Table 4.4.

	Performances of CN2 algorithm
Prediction accuracy	60%
Controlling performance	52%

Table 4.4: The performances of CN2 algorithm for Frozen Lake

As could be seen from the table above, the CN2 algorithm did not perform very well in the testset. This may be followed by the fact that each point was analysed as a sample throughout the training process. This may have resulted in some particular data points carrying their own particular circumstances being treated as general patterns. This could lead to overfitting of the model and thus compromised the accuracy of the testset. In addition, the control of the CN2 algorithm over the agent's actions in Frozen Lake led to a failure in action prediction because of the presence of Default in the rules generated by the CN2 algorithm. the CN2 algorithm always predicted that the agent should move to the right when entering the 'default' region, which was not bounded by the rules. This could also lead to the agent not reaching the final goal point at certain locations. This made CN2 algorithm not very successful as a controller. Combining the two performance results showed that the CN2 algorithm could not perform on its own in this simple environment where there were only two features and the specific value of each feature was determined as a rule condition. This resulted in a significant reduction in model performance compared to reinforcement learning while achieving black-box policy interpretability. This was also in line with the contradiction between the interpretability and performance of the XAI algorithm described in Chapter 3.

For a dataset with a small number of features from the Frozen Lake environment, the common classifier Greedy Rule List and CN2 algorithms did not achieve efficient results. The Rulefit algorithm, which combined a classifier and a regressor, was used to compare the results with the other two algorithms.

4.1.3 The results from Rulefit algorithm

After improving the classifier, Rulefit was applied to the Frozen Lake environment as a hybrid XAI algorithm, combining decision trees with a linear regression Lasso model. This algorithm took a rule-based interpretability process for black-box strategies by first classifying the rules and then linearly fitting the rules to form new rules. As with the above approach, the rules generated from the provided dataset and the specific effects of the algorithmic model would be presented and analysed in this section.

The rules generated from the dataset would be displayed in Tables 4.5. Because of the large number of rules that Rulefit could bring to the table, it would be impossible to show all the rules. Therefore, only the top five most important rules were shown in Tables 4.5. The predicted action for each rule was also shown in the table using the prediction method mentioned above. All rules regarding the generation of Rulefit would be fully presented in Appendix A.2.

Rules	Predicted action	Importance
$11 \geqslant x_{axis} > 7$ and $y_{axis} \leqslant 8$.	2 (Right)	0.182
$10 \geqslant x_{axis} > 7$ and $y_{axis} \leqslant 9.5$.	2 (Right)	0.150
$x_{axis} \leqslant 10$ and $y_{axis} > 10$	1 (Down)	0.102
$x_{axis} > 8$ and $y_{axis} > 11$	1 (Down)	0.085
$x_{axis} \leqslant 10$	1 (Down)	0.077

Table 4.5: The explainable results with top 5 importance by Rulefit for Frozen Lake

The analysis of the specific rules generated by the dataset led to the following conclusions:

- In contrast to the Greedy Rule List, Rulefit generated rules that contained more than just one feature. Some of the rules were bounded by the x-axis coordinates as well as the y-coordinates. In contrast to the CN2 algorithm in the Frozen Lake environment, Rulefit generated rules that were not specific to a single location, but were restricted to a region.
- And depending on the specific rules, interpretable presentations could be easily understood. For example, the first rule was 'If $11 \geqslant x_{axis} > 7$ and $y_{axis} \leqslant 8$, then Right'. This rule located the range in which the x-axis coordinates were in the range of (7,11] and the y-axis coordinates were less than 8. Within this range, the average case of predicted actions made by Rulefit was identified as being Right. Also this region was identified as being the part of the overall dataset that had the greatest impact. This was all in line with the map setting and the activity strategy of the agent movement. The area bounded by the first rule was the area that would be covered by almost every agent that set off from the starting point. So it was logical that the rules within this region played the most important role for the overall action. Also since the target point was located at (12, 12), the right movement was closer to the target point for the region bounded by the first rule. The prediction of action was also logical. The interpretation of the black box strategy involved in the other rules was similar to that of the first rule. In summary, the specific rules generated by Rulefit were not perfect for predicting specific actions at each location, but they could provide humans with a clearer understanding of the regions that the agent focused on in the reinforcement learning plan and the general actions that the agent would take in each region, which was useful for further understanding of the agent's activity.
- Looking at all rules globally, the most important rules were generally located at the bottom and right side of the map. This was consistent with the environmental setting. Since the target point was fixed in the lower right corner of the map, the agents should all move in the two main directions to the right and the down in the map after training. The rules involved in the right and lower regions of the map should also be important in the overall interpretation. This was verified in the rules generated for the dataset.

- Although the rules generated by Rulefit worked well in the Frozen Lake environment. However, it also had some problems. In response to these problems, related conjectures were made. Since the overall activity of the agent moved towards the target point, it was the normal for the target point in the lower right corner for the agent to make downward and rightward movements. Similarly, (state, action) would appear frequently in the dataset. This had an impact on the final rule. When certain similar datasets were too numerous, the rules were judged with a weight bias. This may lead to some rules focusing on a single feature, such as the example 'If $x_{axis} \leqslant 10$, then down'. As discussed in the Greedy Rule List section above, rules with a single feature had too much influence, which may lead to their interpretation of black box strategies being very ambiguous. Also, a very broad rule could affect the prediction of actions. Further validation of this conjecture would be required in future work. Also how to solve the problems arising from the conjectures would be an important part of the future work.

- Overall, Rulefit exhibited a largely acceptable performance in terms of interpretability. Apart from some possible errors, Rulefit allowed humans to intuitively observe the rules that encompassed all features and the types of actions predicted under the corresponding rules. It also allowed humans to understand the areas of focus of the black box strategy in the Frozen Lake environment by the introduction of the importance metric. This was a unique advantage of Rulefit compared to the other two algorithms.

After interpretable analysis of Rulefit, the performance of the model also needed to be further validated. The two methods of validation were the same as those described above.(3.2.3.4) Since the Rulefit algorithm performed a linear fit to the rules derived from the classification, the final predicted value for each action step was not an integer. In order to better compare the predicted values with the actual values, the predicted values were rounded to an integer to represent the corresponding action, but as the predicted value from each Rulefit algorithm model was not a constant value. The final result displayed was also not a constant value. For the result analysis, the performance of the model could be roughly demonstrated by the best results from multiple tests. The performance results were presented in Table 4.6.

	Performances of Rulefit
Prediction accuracy	75%
Controlling performance	67%

Table 4.6: The performances of Rulefit algorithm in Frozen Lake

In the training set, Rulefit achieved a best prediction of about 75%. This was a very high accuracy rate, but at the same time Rulefit still had some false predictions that reduced the accuracy rate. Looking at the table of rules generated by Rulefit, it could be seen that some of the rules contain only one feature, which led to a very ambiguous decision range. Following these rules to make action predictions led to some errors. There were also problems with rules that exceed the map boundaries, which could prevent actions being predicted correctly for particular locations in the testset. The same phenomenon occured in the performance of the controllers. Since the success rate of reinforcement learning was not 100%, the success rate of about 67% for the Rulefit control could also be considered as a local approximation to the black-box learning strategy.

Overall, Rulefit maintained efficient performance of the model in both methods. This also validated the introduction of the Rulefit algorithm, which adds a linear regression lasso model to the interpretable classification algorithm. This approach did allowed the model to maintain better performance results. The improvement of Rulefit over the other two algorithms was particularly noticeable in terms of the impact of the local approximation black box strategy.

After analysing the interpretability results as well as the performance results in Frozen Lake environment, Rulefit was shown to exhibit better results both in terms of interpretability and model performance maintenance. Some conjectures made during the analysis in relation to the results would need to be further validated in future work.

4.1.4 Comparative analysis of results

Combining the performance of the three algorithms in the simulation of the Frozen Lake black box learning process showed that the Greedy Rule List, as an algorithm that could only select one feature at a time for rule generation, generated rules with wide ranges which made the accuracy of the rules very low. The prediction accuracy was already very low in the uncomplicated Frozen Lake environment. The rules generated by the CN2 algorithm could contain multiple features. However, because of the small number of features in Frozen Lake and the inability to range feature values, it was not possible to make accurate predictions for individual features and ultimately for motion prediction. The one rule per location did not allow for the full performance of the CN2 algorithm. In contrast, the Rulefit algorithm, which was able to range multiple features and reflect the importance of specific rules, performed best among three algorithm. With this algorithm, humans could visualise the area covered by the rule and the prediction of the action. At the same time, humans could understand the importance of specific rules to the overall black box strategy. This gave humans a deeper understanding of the black box strategy. At the same time, Rulefit approximated the black box strategy as locally as possible in terms of model performance, both on the testset and in terms of controller performance. The results of Rulefit were broadly acceptable.

4.2 The result analysis for Lunar Lander

Following the interpretability analysis of the dataset derived from reinforcement learning in the Frozen Lake environment, another task arose from the analysis of the complex dataset derived from the black-box learning of the lander in the Lunar Lander environment. Again, the Greedy Rule List, CN2 algorithm and Rulefit would all be applied to this more complex dataset and their respective effects would be compared with each other in this section.

4.2.1 The results from Greedy Rule List

In this section, the Greedy Rule List was applied to Lunar Lander's dataset. The method used was similar to that used in Frozen Lake. The resulting rules were shown in Table 4.7.

Rules	Action
$Velocity_y \geqslant -0.015709255$	Fire the main orientation engine
$Angle\ Velocity \geqslant -0.128028587$	Fire the left orientation engine
$Angle\ Velocity \geqslant 4.96e^{-5}$	Fire the left orientation engine
$Velocity_y \geqslant -2.8e^{-5}$	Fire the left orientation engine
$Position_y \geqslant -0.002390411$	Fire right orientation engine

Table 4.7: The derived rules by Greedy Rule List for Lunar Lander

As could be seen from the table above, the Greedy Rule List still followed the Greedy policy when faced with a more complex dataset with eight different features. This algorithm generated rules by filtering the samples over and over again. Each rule selected only one of the features and limited the rule to a range of feature values greater than or equal to that conditional value. The rules generated by this strategy did not lead to clear interpretability. The disadvantages of explanations of Greedy Rule List were highlighted when analysing the individual rules generated. For example, the rule was 'If $Velocity_y \geqslant -2.8e^{-5}$, then fire the left orientation engine'. According to this rule, the lander should fire the engine to the left if the lander's velocity on the y-axis was down but the velocity value cannot exceed $2.8e^{-5}$ or if the velocity was toward up. This rule was just confusing in its relevance. The action to control the y-axis velocity should either do nothing or turn on the main engine. The action of opening the engine to the right, as predicted by the rule, will not change the y-axis velocity. This made no sense at all for explaining the black box strategy. It may even confuse pre-existing human understanding. Other rules had similar problems. Firstly, the single feature decision made the rule too broad for humans to understand the black box strategy in the complex Lunar Lander environment accurately. At the same time, the features in the rules were significantly not intuitively related to the predicted actions, which not only did not lead to an explanation of the black-box strategy, but also influenced human judgement to some extent. The drawbacks of the Greedy Rule List in the complex Lunar Lander environment were exposed more clearly than in the simple Frozen Lake.

After analysing the rules generated by the Greedy Rule List, the performance of the model was also presented according to the two methods described above. It should be noted that the specific performance of the controller in the Lunar Lander environment was visualised as the number of times that the lander successfully achieved a reward of 200 out of 100 flight tests. The performance was shown in Table 4.8.

	Performances of Greedy Rule List
Prediction accuracy	6%
Controlling performance	Fail

Table 4.8: The performances of Greedy Rule List in Lunar Lander

As could be seen from the graph above, the Greedy Rule List performed very poorly in the Lunar Lander environment. In the testset, its accuracy was only 6%. In the controller session, the lander under the control of Greedy Rule List was completely out of control and did not complete the given task. This result also demonstrated that Greedy Rule List performed worse in complex environments. Its algorithmic strategy of limiting only one feature in a rule was not at all interpretable. The generated rules affected the normal operation of the lander to some extent. This made the lander's performance ineffective. In summary, the Greedy Rule List was almost completely ineffective in the Lunar Lander environment. With this in mind, two other algorithms were applied to Lunar Lander in the hope of getting better results.

4.2.2 The results from CN2 algorithm

The CN2 algorithm was considered to be an excellent classification method that could handle multi-feature dataset. Applying it to the large dataset of multi-features generated by reinforcement learning in Lunar Lander was a planned and feasible solution.

The initial training set needed to be processed before the CN2 algorithm process, which associated features with different eigenvalues and gave a table of rules containing predictions. The initial Lunar Lander dataset contained 4465 different sets of data from 11 trajectories. The eight different features consisted of six physical quantities with specific values (x-axis coordinates, y-axis coordinates, x-axis velocity, y-axis velocity, angle and angular velocity) and two binary determinants (whether the first leg touched the ground or not and whether the second leg touched the ground or not). Treating all values of a feature as a class of that feature would result in a dataset with too many features. The model may suffer from overfitting. Therefore each feature needed to be grouped. As each project discussed the most basic XAI algorithm for reinforcement learning in dynamic systems, it was representative that the most basic dataset was used to train the corresponding algorithmic model. Based on this, each feature quantity containing a specific value was divided into ten groups. The feature quantities containing binary determinations were divided into two groups. All features containing special values were grouped based on the Minimum Description Length Principle (MDLP). The MDLP method was an information dispersion method based on information entropy. This method allowed for more efficient discretization of unevenly distributed continuous datasets. This property determined its suitability for this project. The final predicted actions were used as label classes. The group numbers of different features were shown in Table 4.9. Because the length of the specific data division tables were too long, the tables had been placed in Appendix A.1 for reference purposes.

Pos_x	Pos_y	Vel_x	Vel_y	Angle	Angle Angvel	$Contact_1$	$Contact_2$
18	13	12	10	8	9	2	2

Table 4.9: The group number of different features with values

After the range was divided as above, the new dataset was applied to the CN2 algorithm model. Over 200 rules were eventually obtained. Because of space limitations, it was not possible to show all the rules, and some typical examples were shown in Table 4.10 to explain how the rules generated by the CN2 algorithm could be interpreted for black-box strategies. A more detailed form could be found in the Github repository.

Rule	Coverage	Precision
If $Pos_x = (0.0332, 0.0333]$, then No action.	0.11%	100%
If $Vel_y = (0.211, 0.455]$ and $Pos_x = (-0.0884, -0.0496]$, then No action.	0.22%	100%
If $Vel_y = (-0.368, -0.183]$ and $Angle = (-0.601, -0.416]$ and $Vel_x = (-0.168, -0.0044]$, then Fire down engine.	0.13%	100%
If $Angle = (-0.601, -0.416]$ and $Vel_y = (-0.183, -0.0059]$ and $Pos_x = (-0.526, -0.3794]$ and $Angvel = (-0.303, -0.167]$, then Fire Down engine.	0.13%	100%
Default,Fire down engine.	100%	50%

Table 4.10: The examples of rule list by CN2 algorithm for Lunar Lander

The examples in the table above showed that the CN2 algorithm was significantly more effective than the Greedy Rule List in classifying such complex dataset with multiple features. The number of features covered by the rules generated by the CN2 algorithm was variable. From the typical example, it could be seen that the generated rules could be associated with one feature or with multiple features. In the examples, there was a maximum of one rule qualifying up to four features.

Also through specific rules, humans were able to gain a deeper understanding of the black box strategy in the Lunar Lander environment. As an example, the second rule was 'If $Vel_y = (0.211, 0.455]$ and $Pos_x = (-0.0884, -0.0496]$,then No action.' This rule was a correlation between the velocity on the y-axis and the coordinates of the x-axis. When the y-axis velocity was in the range $(0.211, 0.455]$ and the x-axis coordinate was in the range $(-0.0884, -0.0496]$, the rule predicted that the lander's next action was No action. This prediction was logical in a realistic decision. Since the lander needed to land safely at (0, 0), the coordinates of the lander's position needed to be close to (0, 0). At the same time, in order to achieve a smooth landing, the lander needed to land with the lowest possible velocity. The x-axis coordinate was almost close to 0 in the first rule limited case and the velocity of the y-axis was in the opposite direction, upwards. What the lander needed to do in this case was to reduce the velocity in the upward direction of the y-axis. Therefore, it was logical for the lander to take the No action in this case. This proved that the rule could logically explained the agent's action planning.

Another example was the third rule, 'If $Vel_y = (-0.368, -0.183]$ and $Angle = (-0.601, -0.416]$ and $Vel_x = (-0.168, -0.0044]$, then Fire down engine'.

In the case limited by the third rule, the lander angle presented a negative angle with a leftward motion on the x-axis and a downward motion on the y-axis. In order to land as smoothly as possible, the velocity of the lander was expected to be as close to 0 in the x-axis and y-axis as possible. Because the negative angle of the lander, firing down engine would cause the lander to acquire a rightward component in the x-axis and an upward component in the y-axis. This was consistent with the mission of the lander and the logic of the physical system. According to this strategy, the action to be performed by the lander at this point was firing down engine. The action predicted by the rule was Fire down engine, which reflected the interpretability of the black box strategy. This was also consistent with physics.

The lander was actually in a falling body-like motion throughout its movement. Because of the effect of gravity, the velocity of the lander downwards on the y-axis increased continuously in the absence of external forces. To ensure a smooth landing, it was common for the lander to execute the Fire down engine from time to time to slow down the lander. It was therefore logical that the Fire down engine was commonly executed in the Default case. In summary, the rules generated by the CN2 algorithm had some explanatory effect on the black box policy in the Lunar Lander environment. From the rules, humans could clearly understand what actions the lander should perform in the specific feature range.

Following the interpretability analysis of the rules generated by CN2, the performances of the CN2 algorithm model also needed to be demonstrated to prove its effectiveness. The methods used here were as described in the Greedy Rule List section, the testset and the controller's methods were used to evaluate the performance of the model. The results of the evaluation were shown in Table 4.11.

	Performances of CN2
Prediction accuracy	65%
Controlling performance	2%

Table 4.11: The performances of CN2 algorithm in Lunar Lander

From the table above, the performances of the CN2 algorithm on the test set was generally acceptable. Considering that the performance of the lander under reinforcement learning was not 100% successful either, CN2 still worked well for the local approximation of the black-box policy. The reason why the results in the test set could not keep the same performance as the one in the training set may be followed by the overfitting of the model because of its high complexity. Some specific cases of particular data were generalised by the model. Also likely to affect the performance of the model was the uneven distribution of data in the dataset. As the dataset was derived from well performing motion trajectories in Lunar Lander. All trajectories had a particularly high proportion of data near the start point as well as the target point, which led to an uneven distribution of features in the overall dataset. In future work, improving the performance of the CN2 algorithm model by tuning and optimising the dataset would also be a major task to be considered. However, CN2 also showed poor prediction accuracy in the performance of the controller. The seed would influence the results significantly. The best result by the suitable seed could get 2% prediction accuracy. Through this result was better than others by other algorithms, it was still unacceptable. Since continuous prediction of correct actions was very important in complex dynamic systems, it was clear that a training approach based on recorded discrete data alone could not achieve 100% prediction of the lander's actions. It was possible to speculate that although the CN2 algorithm had acceptable prediction accuracy for the test set, the inevitable loss of accuracy in performance followed by its interpretability would lead to some prediction errors. A very small number of prediction errors could have irreversible negative consequences in complex dynamic systems. These undesirable consequences could ultimately lead to ineffective control of the controller.

4.2.3 The results from Rulefit algorithm

As in the Frozen Lake setting, Rulefit, which combined a classifier with a regressor, was used to account for the black-box strategy in the Luanr Lander setting. With the Rulefit algorithm, rules containing multiple eigenvalues were presented. Because the number of rules generated was large, it was impractical to show them all in the text, so only the rules with top 5 importance were shown in the results analysis session as examples of the Rulefit algorithm explaining reinforcement learning in Lunar Lander. These rules and the overall decisions under the corresponding rules were shown in Table 4.12. A more detailed form could be found in the Github repository. The importance of specific rules was also presented in the table to help humans understand the focus of the black box strategy and the direction of the planned strategy.

Rules	Predicted action	Importance
$0 < Velocity_y \leqslant 0.0$ and $Angle \leqslant 0$ and $Angle\ Velocity \leqslant 0$	-	0.136
$Velocity_x \leqslant 0$ and $Angle\ Velocity \leqslant 0$	Fire the right engine	0.112
$Velocity_x$ (Linear)	-	0.111
$Position_x$ (Linear)	-	0.104
$Position_x > 0$ and $Position_y \leqslant 0$ $Velocity_y \leqslant 0$	Fire the main engine	0.098

Table 4.12: The explainable results with top 5 importance by Rulefit for Lunar Lander

The interpretability of the rules derived from Rulefit became poor when faced with complex datasets with multiple features. It could be concluded from the rules given that the rules given by Rulefit were very inaccurate in qualifying the specific values of the features when faced with a dataset with multiple features and a large number of different values covered by each feature. A glance at the rules listed above showed that most of the rules generated could only be separated by a zero. This led to a low precision of the rules and consequently to a poor interpretation of their coverage.

An assumption could be made as to the reasons. Firstly, the rules created by Rulefit were based on binary decisions. This was the normal application for Rulefit algorithm. In the Frozen Lake environment there were only two types of features (x-axis coordinates and y-axis coordinates) and the values of each feature were only integers between 1 and 12. The number of rules from this simple environment was small. And the number of conditions of each rules was small (Only one or two conditions) The generated rules were therefore clear and unambiguous. The numbering of the predicted actions could also be inferred from the average number of predicted values between the areas defined by the rules. However, this was not feasible in the complex dataset of Lunar Lander, which had six sets of features with specific values and two sets of binary decision features with the range $[0, 1]$. As the input values to rulefit could only be arrays, grouping of feature values for the dataset was not applicable here. This posed a great challenge for the generated rules. Each feature would have a different node value for splitting in the decision tree.

Also, the number of conditions in each rule became very large. Although the conditions for each rule were kept to 1-4 on the basis of tuning, many of the rules were not well understood in terms of comprehensibility. Some rules from Rulefit could not give a specific prediction when there were many conditions in the rule. For example, the first rule was '$0 < Velocity_y \leqslant 0.0$ and $Angle \leqslant 0$ and $Angle\ Velocity \leqslant 0$'. In this rule, the range of $Velocity_y$ had a contradiction in the numerical limits. In the region limited by this rule, there were not any sample. So the rule could not give the prediction. According to such a rule, a hypothesis was formulated. When faced the complex task such as the Lunar Lander in this project, some conditions would be combined into a rule with some errors. From the table above, only two rules could give the predictions. These predictions were logical to a certain extent. For example, the rule '$Velocity_x \leqslant 0$ and $Angle\ Velocity \leqslant 0$' could predict that next action would be firing the right engine. This action could made the $Velocity_y$ and Angle Velocity close to 0, which were the correct state values for these two features when landing the final point. But there were too few rules that were logical and give predictions. Overall, it seemed that Rulefit generated rules that were not very explanatory. At the same time, because Rulefit itself took the binary decisions created by decision trees and brought them into a sparse linear model for feature reduction, it had a certain chance of producing linear results when working with the complex dataset, as in the table above for the linear conclusion on x-axis velocity and x-axis position. With these linear conclusions could not determine the rule limits and could not make further predictions about future actions.

Another reason not to be overlooked was that the more features there were, the more rules there would be. Despite the filtering and cleaning of the features by the Lasso model, the final rules still overwrited each other. At the same time, Rulefit had also been applied to multi-feature classification problems in static settings, such as bicycle rental factors or bank lending problems, where it was often used to compare the importance of features and to make binary decisions. This project applied the Rulefit algorithm to multi-action decision making in dynamic systems, which in itself was a new attempt at Rulefit algorithm. The poor results obtained may be a side indication that Rulefit was not applicable to the interpretation of decision making in dynamic systems. The above was a conjecture as to why Rulefit was much less interpretable in complex dynamic systems. Specific further validation and related improvements could be the focus of future work.

After the analysis of the specific rules generated, the performances of the Rulefit model were evaluated in the same way. The results of this evaluation were presented in Table 4.13.

	Performances of Greedy Rule List
Prediction accuracy	30%
Controlling performance	Fail

Table 4.13: The performances of Rulefit in Lunar Lander

Rulefit was poorly interpretable in explaining the black box strategy in Lunar Lander. Many of the rules generated did not predict actions correctly. As a result, the model performance effectiveness maintained by Rulefit in Lunar Lander was also significantly degraded. By comparing the final linear fit results after rounding with the corresponding action labels in the actual testset, the best accuracy was only around 30%. This indicated that the Rulefit-generated rules were unable to correctly predict actions for unknown samples. At the same time, it could be concluded from the controller's performance that the Rulefit algorithm was also very ineffective when applied to actual real-time dynamic control rather than process data analysis. The probability of the lander successfully completing the task under Rulefit control was also very small. This phenomenon was from the fact that Rulefit maintained a certain level of effectiveness when dealing with discrete data. However, when dealing with continuous dynamics, the linear non-integer predictions of its final output could not be accurately located to the predicted action. In dynamic systems, one or two wrong predictions could sometimes lead to a significant drop in the final result.

4.2.4 Comparative analysis of results

In summary, the CN2 algorithm, which processed the grouped features to generate the corresponding classification patterns, gave the relatively best results in terms of interpretability and model performance when dealing with the large dataset generated by the relatively complex Lunar Lander environment. Neither the Greedy Rule List, which followed a greedy strategy alone, nor the Rulefit algorithm, which combined binary decision rules with linear fitting, provided an accurate and understandable explanation of the black-box decisions.

4.3 Comparative analysis for different environments and discussion

In this section, the results in the two environments were compared and analysed. Also through the differences in these results, conclusions about specific XAI techniques for black-box strategies in dynamic systems were presented.

Through the analysis of the results as above, the specific performances of different algorithms in different environments were shown. By grading these algorithms specifically for interpretability as well as performance maintenance, the specific performances of the final three algorithms in the two environments were recorded in Table 4.14.

	Interpretability	Model performance
Greedy Rule List in Frozen Lake	Low	Low
CN2 in Frozen Lake	Medium	Medium
Rulefit in Frozen Lake	High	High
Greedy Rule List in Lunar Lander	Low	Low
CN2 in Lunar Lander	High	Low
Rulefit in Lunar Lander	Low	Low

Table 4.14: The performance levels of three algorithms in two environment

Combined with Table 4.14, the analysis of the results of the three algorithms in the two environments led to the following conclusions:

- The datasets generated by the post-training trajectories from reinforcement learning in dynamic systems were more complex than the usual discrete datasets. The results obtained by the rule-based XAI algorithm after training these datasets did not reach a very efficient level. This phenomenon may be followed by the fact that the entire trajectory in dynamic systems was a continuous process. If this was discretised and then applied to the XAI algorithm model, a large amount of information was lost in it. This missing information affects the final performance of the XAI algorithm.

- In either environment, the Greedy Rule List algorithm, which followed a greedy strategy, generated rules containing only one feature at a time. This results in a very wide range of rules, which led to poorly interpreted rules with low accuracy. The disadvantages of Greedy Rule List were even more evident in complex environments.

- When faced with a simple environment (Frozen Lake) generating a dataset with relatively few features, the Rulefit algorithm needed to produce a small number of rules with few conditions per rule. In such cases, the combination of the range rules generated by Rulefit for the x-axis coordinates as well as the y-axis coordinates and the overall actions predicted within the corresponding rule range could provide a clear explanation of the overall black box strategy. It allowed humans to intuitively understand the fixed range of action predictions and the region of focus considered by the black box strategy. However, when more features were added and the amount of data increases, Rulefit's rules based on binary decisions also increased significantly, while each rule contained more feature conditions. Even though the Lasso model made a fitted weight reduction to the generated rules, the generated feature conditions on specific values were still very complex. This resulted in the final generated rules being difficult to understand. The accuracy of the generated rules was unacceptable when used to predict the testset data. However, its accuracy would be higher than the one of Greedy Rule List. At the same time, when used as a controller to manipulate the continuum of motion of the lander, the results were still qualified. An analysis of the Rulefit algorithm itself led to the conjecture that Rulefit was simply a linear fit to the rules for binary decisions. When it was used to classify continuous movements in complex dynamic systems, it was unable to provide a valid interpretation of the specific values of the individual features.

- Finally, the CN2 algorithm combining the ID3 decision tree as well as the AQ algorithm could be effectively classified for multiple features. In the simple environment, the CN2 algorithm could only generate rules for each specific value of a feature, as there were too few features to classify them effectively. This allowed some specific features of the data to be treated as general rules. Too many general rules caused the model to be overfitted. This resulted in the CN2 could not show efficient performances in the testset and as a controller. However, when dealing with complex datasets, by classifying features into ranges and treating these ranges as classes of features, the CN2 algorithm could generate rules that were easily understood by humans in accordance with real physics. The CN2 algorithm also performed the best of the three algorithms in the testset. However, the CN2 algorithm did not provide very efficient results for complex dynamic systems that required high accuracy in predicting each step in the controller.

Followed by the problems that occurred throughout the project, the conclusions was drawn and the conjectures was proposed. The future work could focus on these aspects. Specific future plans would be shown in the next chapter. A summary of the whole project would also be presented in the next chapter.

Chapter 5

Future Work and Conclusion

5.1 Future work

Through the analysis of the results, the shortcomings of this project were shown. In response to the existing shortcomings, future work would focus on improving various aspects of the project. The specific improvements were summarised in the following points.

5.1.1 Further improvements in model performance

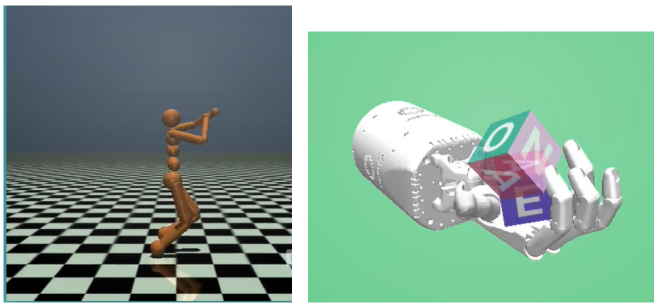
In future work, how to further improve the efficiency of existing models will be the first thing to be considered. Based on the existing research, the dataset would be subdivided into validation sets, through which the hyperparameters of the model will be tuned. The optimal hyperparameters will be selected to further improve the model's specific results. Additional components, such as neural networks, will be also considered for inclusion in the original model to improve its efficiency.

5.1.2 Further verification of the conjectures

Future work will focus on validating the conjectures and hypotheses from the analysis of the results of this project. For the errors that occur throughout the experiment such as rules exceeding the Frozen Lake map bounds, future work will look at how to address these errors to help further improve the effectiveness of the XAI algorithms.

5.1.3 The more complex and factual construction of the environment

Throughout the environments involved in this project, both the relatively simple mesh world of Frozen Lake and the more complex 2D rigid-body movement of Lunar Lander were somewhat different from real-world dynamic systems. Both the mesh world and 2D rigid body motion limit dynamic motion planning to the 2D world. The real world is a three-dimensional world, where there are many more states of motion and many more types of action. Therefore, it is the future goal of this project to try to explain black-box reinforcement learning in a more complex 3D environment. The future plans can be formulated as still using the Open AI Gym platform and explaining the rules embedded in 3D motion and other interpretable AI-related phenomena in the MuJoCo and Robotics environments, the visualisations of the classic project Humanoid in the MuJoCo environment and of the classic project in the Robotics environment HandManipulateBlock are shown in Figure 5.1. These are both further enhancements that can be tried in the future in a 3-dimensional environment.



(a) Humanoid (b) HandManipulateBlock

Figure 5.1: The 3D environments for future work

Researchers have used these environments to launch a series of black box studies. The relevant literature may also provide technical support for future analysis of more complex XAI algorithms in these environments. [33] [10] [44]

5.1.4 The more comprehensive range of dataset types

After the data output from the dynamic system became a dataset in this project, the final rules were mostly used to generalise the data in these datasets, as discrete data for the rules. Although most of the rule generation at this stage also relied on discrete data. However, as real-world dynamic systems are often continuous processes, there is great potential value in rule generation and XAI techniques for continuous data. Future work will also focus on the analysis of continuous data and aim to derive XAI algorithmic techniques that are generally applicable to continuous data.

5.1.5 Issue handling for more features

In this project, the only features to be considered for the Frozen Lake part of the rule were the x-axis and y-axis, and the number of features to be determined was too small. Even in the more complex Lunar Lander environment, there were only 8 state attributes for the agent. In a real environment, there would be many more features to be interpreted by the rules, and many more factors that would affect the agent. To address this phenomenon, future work should focus on rule learning for data with more features and on testing other XAI algorithms. Perhaps a project with 10 or even 20 relatively independent features in the motion of an agent would be a valuable future project.

5.1.6 More rule-based XAI algorithms

In this project the Greedy Rule List, CN2 algorithm and Rulefit were used to explain the black-box reinforcement learning process in two environments. All three algorithms were rule-based learning algorithms. The final explanation of the black-box strategy was also demonstrated by specific rules. The analysis of the results showed that there are advantages and disadvantages to the three algorithms, but at the same time, none of the algorithms could give a very specific and detailed and accurate rule explanation of the black box strategy while maintaining efficient performance of the model. Therefore, future research will focus on more rule-based interpretable AI algorithms. The Skope Rule algorithm could be selected as a new rule-based XAI algorithm for future work. [13]

5.1.7 More expressions of XAI technology

All of the XAI techniques used in this project were rule-based learning. This made their interpretability somewhat limited. For future work, more diverse forms of XAI algorithmic techniques were worthy of further investigation. Not only could rule-based learning represent black-box strategies, but other XAI techniques can also explain black-box strategies to some extent at other levels. Model-independent XAI techniques, For example, are another major area of research for future work. Unlike model-related XAI algorithms such as rule-based learning, model-independent XAI techniques often work by adjusting the black-box model input and observing the output of the black-box model that changes as the input changes. The differences in the outputs are then used to produce statistical graphs that provide some explanation of the black box model. A detailed workflow can be found in Figure 5.2.

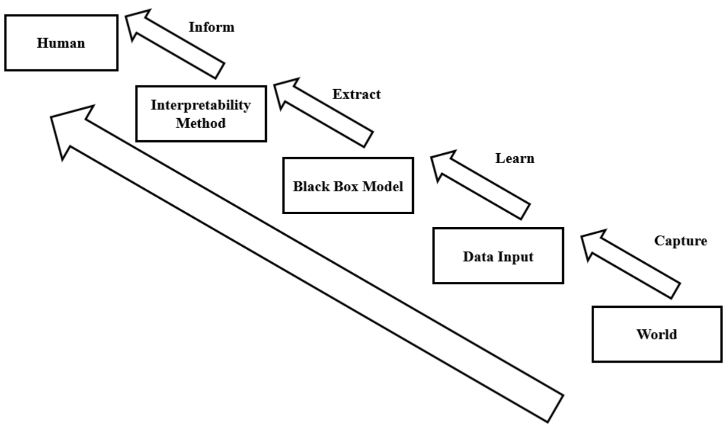


Figure 5.2: The basic flowchart of the model-independent XAI algorithms

The most common of these methods are Permutation Importance, which focuses on importance; [2] LIME, which focuses on local interpretation; [61] Shapley, which is derived from game theory [48] and Global Surrogate, which incorporates the idea of dimensionality reduction. [18] These methods are planned as an important part of future work.

5.1.8 Summary of the future work

Summing up the various plans for future work above, the future work of this project is summarised by considering dynamic systems with a wider variety of features and analysing their continuity data in a more complex and simulated 3-dimensional environment. In the XAI analysis of these data, a greater variety of XAI techniques will be applied. Not only rule-based XAI techniques, but also other techniques such as model-independent XAI will be used to obtain more comprehensive and insightful results.

5.2 Conclusion

The project focused on the analysis of XAI algorithms oriented towards black-box reinforcement learning processes. It was a meaningful experiment in XAI technology and a basis for future research on XAI algorithms. In this project, the Frozen Lake and Lunar Lander environments of the Open AI Gym platform were used as dynamic systems. The reinforcement learning process performed on the agent in these two environments could be considered as a black-box learning process. Three representative XAI algorithms, the Greedy Rule List, the CN2 algorithm and the Rulefit algorithm, were used for the black-box reinforcement learning process in these two environments. After the analysis of the results and in-depth discussions, the conclusions of this project could be summarised as follows:

- The Greedy Rule List algorithm did not perform well in a 2D environment because it could only generate rules with a single eigenvalue. In contrast, the CN2 and Rulefit algorithms performed much better. Simple classifier algorithms were no longer sufficient to meet the needs of XAI technology. The combination of classifier models and regression models was the only way to make the XAI algorithm more efficient.

- Also, as the algorithms used in this project were rule-based, they ended up as inductive rules that attempt to explain the black-box strategy. To some extent, this class of 'if, then' form of rules could be observed intuitively by humans. This point did bring some explanation to the original black-box learning process. However, it was also undeniable that a single rule did not provide a very comprehensive explanation of the global picture of a black box model. A combination of more diverse XAI algorithms may provide better results.

- Throughout the project, the initial perception of XAI technology was reaffirmed, not as a 'substitute' for black-box strategies, but as an 'explainer'. The XAI algorithm was shown to be a local approximation of the black-box model, providing interpretability while causing the performance of the black-box model to be somewhat degraded. This was in line with the prediction made in the introduction section: the rule-based XAI algorithm was highly effective in terms of interpretability. However, its accuracy for predicting outcomes was significantly lower compared to the black box model. This is the main pain point of rule-based XAI algorithms at this stage. How to maintain a balance between interpretability and model performance would be the biggest task and choice faced by XAI technology.

Throughout the project, the significance it brought is extraordinary. For the technical side of XAI research, this project was a fundamental exploration of XAI technology. All subsequent related research would build on this foundation. For the researchers, XAI technology was a completely new field. Through this project, the researcher understood how to proceed step by step when dealing with a completely new field. The process and content of the project, the supervision of the supervisors and the support of the staff involved in the project made this project a very significant one for the whole Masters programme.

Bibliography

[1] B Abdollahi and O Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129*, 2016.

[2] A Altmann, L Toloşi, O Sander, and T Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

[3] A B Arrieta, N Díaz-Rodríguez, J Del S, A Bennetot, S Tabik, A Barbado, Salvador García, S Gil-López, D Molina, R Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[4] A Banerjee, D Ghosh, and S Das. Evolving network topology in policy gradient reinforcement learning algorithms. In *2019 Second International Conference on Advanced Computational and Communication Paradigms (ICACCP)*, pages 1–5. IEEE, 2019.

[5] D C Brock. Learning from artificial intelligence's previous awakenings: The history of expert systems. *AI magazine*, 39(3):3–15, 2018.

[6] G Brockman, V Cheung, L Pettersson, J Schneider, J Schulman, J Tang, and W Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[7] M Bunge. A general black box theory. *Philosophy of Science*, 30(4):346–358, 1963.

[8] R Caruana, S Lundberg, M T Ribeiro, H Nori, and S Jenkins. Intelligible and explainable machine learning: Best practices and practical challenges. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[9] G Cervone, P Franzese, and A P Keesee. Algorithm quasi-optimal (aq) learning. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(2):218–236, 2010.

[10] P Christiano, Z Shah, I Mordatch, J Schneider, T Blackwell, J Tobin, P Abbeel, and W Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[11] W J Clancey. Notes on "epistemology of a rule-based expert system". *Artificial Intelligence in Perspective*, page 197, 1994.

[12] P Clark and T Niblett. The cn2 induction algorithm. *Machine learning*, 3(4):261–283, 1989.

[13] Skope Collaboration et al. Skope-rules. *GitHub repository*, 2021.

[14] B Jack Copeland. *The essential turing.* Clarendon Press, 2004.

[15] O Csiszár, Gábor Csiszár, and József Dombi. Interpretable neural networks based on continuous-valued logic and multicriteria decision operators. *Knowledge-Based Systems*, 199:105972, 2020.

[16] M Cullen, B Davey, K J Friston, and R J Moran. Active inference in openai gym: A paradigm for computational investigations into psychiatric illness. *Biological psychiatry: cognitive neuroscience and neuroimaging*, 3(9):809–818, 2018.

[17] A Das and P Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[18] A Dhurandhar, C Pin-Yu, R Luss, T Chun-Chen, P Ting, K Shanmugam, and P Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *arXiv preprint arXiv:1802.07623*, 2018.

[19] N Douglas, D Yim, B Kartal, P Hernandez-Leal, F Maurer, and M E Taylor. Towers of saliency: A reinforcement learning visualization using immersive environments. In *Proceedings of the 2019 acm international conference on interactive surfaces and spaces*, pages 339–342, 2019.

[20] J Druce, M Harradon, and J Tittle. Explainable artificial intelligence (xai) for increasing user trust in deep reinforcement learning driven autonomous systems. *arXiv preprint arXiv:2106.03775*, 2021.

[21] L M Fagan, E H Shortliffe, and B G Buchanan. Computer-based medical decision making: from mycin to vm. *Automedica*, 3(2):97–108, 1980.

[22] J H Friedman and B E Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[23] J Fürnkranz, D Gamberger, and N Lavrač. *Foundations of rule learning*. Springer Science & Business Media, 2012.

[24] S Gadgil, Y Xin, and C Xu. Solving the lunar lander problem under uncertainty using reinforcement learning. In *2020 SoutheastCon*, volume 2, pages 1–8. IEEE, 2020.

[25] Y Gao, X Huazhe, J Lin, F Yu, S Levine, and T Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[26] L H Gilpin, D Bau, B Z Yuan, A Bajwa, M Specter, and L Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[27] R Glanville. Black boxes. *Cybernetics & Human Knowing*, 16(1).

[28] D Gunning, M Stefik, J Choi, T Miller, S Stumpf, and Guang-Z Yang. Xai—explainable artificial intelligence. *Science Robotics*, 4(37), 2019.

[29] C Han-Yun and Ching-Hung. Vibration signals analysis by explainable artificial intelligence (xai) approach: Application on bearing faults diagnosis. *IEEE Access*, 8:134246–134256, 2020.

[30] N Heath. What is ai? everything you need to know about artificial intelligence. *ZDNet, available at: https://www. zdnet. com/article/what-is-ai-everything-you-need-to-know-about-artificial-intelligence/(accessed 23.09. 2018)*, 2018.

[31] A Holzinger, C Biemann, C S Pattichis, and journal="arXiv preprint arXiv:1712.09923" year=2017 Kell, D B". What do we need to build explainable ai systems for the medical domain?

[32] P Jain, M Kumar Singh, and P Kumar Gangwar. A look towards the human like intelligence: Artificial intelligence. *RTCSE-2016*, page 115.

[33] H Junning and H Zhifeng. Better sampling strategy for locomotion control tasks. 2018.

[34] Nikolai K. Hse reinforcement learning. *GitHub repository*, 2021.

[35] D Kahneman, S Paul Slovic, P Slovic, and A Tversky. *Judgment under uncertainty: Heuristics and biases*. Cambridge university press, 1982.

[36] A Kaplan and M Haenlein. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019.

[37] A Kirilenko, A S Kyle, M Samadi, and T Tuzun. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, 72(3):967–998, 2017.

[38] R Kurzweil. The singularity is near. In *Ethics and emerging technologies*, pages 393–406. Springer, 2014.

[39] M Kuzlu, U Cali, V Sharma, and Z Güler. Gaining insight into solar photovoltaic power generation forecasting utilizing explainable artificial intelligence tools. *IEEE Access*, 8:187814–187823, 2020.

[40] H Lakkaraju and C Rudin. Learning cost-effective and interpretable treatment regimes. In *Artificial intelligence and statistics*, pages 166–175. PMLR, 2017.

[41] M Langer, D Oster, T Speith, H Hermanns, L Kästner, E Schmidt, A Sesing, and K Baum. What do we want from explainable artificial intelligence (xai)?–a stakeholder perspective on xai and a conceptual model guiding interdisciplinary xai research. *Artificial Intelligence*, 296:103473, 2021.

[42] Xingyu Li and B I Epureanu. Ai-based competition of autonomous vehicle fleets with application to fleet modularity. *European Journal of Operational Research*, 287(3):856–874, 2020.

[43] T P Lillicrap, J J Hunt, A Pritzel, N Heess, T Erez, Y Tassa, D Silver, and D Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[44] J H Lim, P O O Pinheiro, N Rostamzadeh, C Pal, and S Ahn. Neural multisensory scene inference. *Advances in Neural Information Processing Systems*, 32:8996–9006, 2019.

[45] Z C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[46] Y Lu, M S Squillante, and C W Wu. A general markov decision process framework for directly learning optimal control policies. *arXiv preprint arXiv:1905.12009*, 2019.

[47] George F Luger. *Artificial intelligence: structures and strategies for complex problem solving.* Pearson education, 2005.

[48] S M Lundberg and S Lee. A unified approach to interpreting model predictions. In *Proceedings of the 31st international conference on neural information processing systems*, pages 4768–4777, 2017.

[49] Alessia M. Cn2 rule-based classifier. 2019.

[50] T Matiisen. Demystifying deep reinforcement learning. *Computational Neuroscience LAB*, 19, 2015.

[51] J McCarthy. Review of the question of artificial intelligence. *Annals of the History of Computing*, 10(3):224–229, 1988.

[52] V Mnih, K Kavukcuoglu, D Silver, A Graves, I Antonoglou, D Wierstra, and M Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[53] B Michael Moores. Artificial intelligence and deep learning in diagnostic radiology—is this the next phase of scientific and technological development? *Radiation Protection Dosimetry*, 2021.

[54] W James Murdoch, C Singh, K Kumbier, R Abbasi-Asl, and B Yu. Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*, 2019.

[55] N J Nilsson and N Johan Nilsson. *Artificial intelligence: a new synthesis.* Morgan Kaufmann, 1998.

[56] S Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)*, 54(6):1–25, 2021.

[57] C Peters, T C Stewart, R L West, and B Esfandiari. Dynamic action selection in openai using spiking neural networks. In *The Thirty-Second International Flairs Conference*, 2019.

[58] D Poole, A Mackworth, and R Goebel. Computational intelligence. 1998.

[59] E Puiutta and E M Veith. Explainable reinforcement learning: A survey. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 77–95. Springer, 2020.

[60] J R Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[61] M T Ribeiro, S Singh, and C Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[62] I Sample. Computer says no: why making ais fair, accountable and transparent is crucial. *The Guardian*, 5:1–15, 2017.

[63] A Pinar Saygin, I Cicekli, and Vl Akman. Turing test: 50 years later. *Minds and machines*, 10(4):463–518, 2000.

[64] M Shelley. *Frankenstein, or the Modern Prometheus: Annotated for Scientists, Engineers, and Creators of All Kinds, edited by David H.* Finn, and Jason Scott Robert, Cambridge and London: The MIT Press, 2017.

[65] C Singh, K Nasseri, Y S Tan, T Tang, and B Yu. imodels: a python package for fitting interpretable models, 2021.

[66] R Stuart and N Peter. *Artificial intelligence-a modern approach 3rd ed.* Berkeley, 2016.

[67] R S Sutton and A G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[68] W R Swartout. Xplain: A system for creating and explaining expert consulting programs. *Artificial intelligence*, 21(3):285–325, 1983.

[69] A B Tosun, F Pullara, M J Becich, D Taylor, J L Fine, and S Chakra Chennubhotla. Explainable ai (xai) for anatomic pathology. *Advances in Anatomic Pathology*, 27(4):241–250, 2020.

[70] M Turchetta, A Kolobov, S Shah, A Krause, and A Agarwal. Safe reinforcement learning via curriculum induction. *arXiv preprint arXiv:2006.12136*, 2020.

[71] J van der Waa, E Nieuwburg, A Cremers, and M Neerincx. Evaluating xai: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, 291:103404, 2021.

[72] José R Vázquez-Canteli, Jérôme Kämpf, G Henze, and Z Nagy. Citylearn v1. 0: An openai gym environment for demand response with deep reinforcement learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 356–357, 2019.

[73] Z Xingyu, D Shifei, A Yuexuan, and J Weikuan. Asynchronous reinforcement learning algorithms for solving discrete space path planning problems. *Applied Intelligence*, 48(12):4889–4904, 2018.

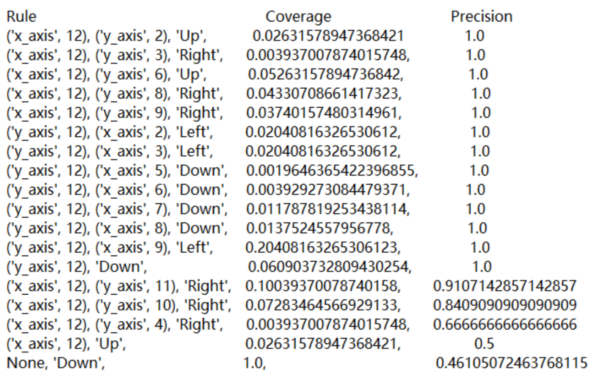
[74] L Yuxi. Reinforcement learning applications. *arXiv preprint arXiv:1908.06973*, 2019.

[75] I Zamora, N Gonzalez Lopez, V Mayoral Vilches, and A Hernandez Cordero. Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*, 2016.

Appendix A

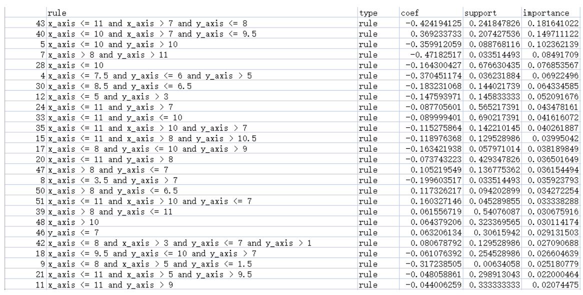
Appendix

A.1 The rule list of CN2 algorithm in Frozen Lake



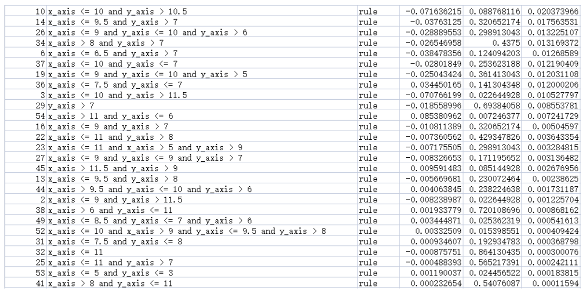
Rule	Coverage	Precision
('x_axis', 12), ('y_axis', 2), 'Up',	0.02631578947368421	1.0
('x_axis', 12), ('y_axis', 3), 'Right',	0.003937007874015748,	1.0
('x_axis', 12), ('y_axis', 6), 'Up',	0.05263157894736842,	1.0
('x_axis', 12), ('y_axis', 8), 'Right',	0.04330708661417323,	1.0
('x_axis', 12), ('y_axis', 9), 'Right',	0.03740157480314961,	1.0
('y_axis', 12), ('x_axis', 2), 'Left',	0.02040816326530612,	1.0
('y_axis', 12), ('x_axis', 3), 'Left',	0.02040816326530612,	1.0
('y_axis', 12), ('x_axis', 5), 'Down',	0.0019646365422396855,	1.0
('y_axis', 12), ('x_axis', 6), 'Down',	0.003929273084479371,	1.0
('y_axis', 12), ('x_axis', 7), 'Down',	0.011787819253438114,	1.0
('y_axis', 12), ('x_axis', 8), 'Down',	0.0137524557956778,	1.0
('y_axis', 12), ('x_axis', 9), 'Left',	0.20408163265306123,	1.0
('y_axis', 12), 'Down',	0.060903732809430254,	1.0
('x_axis', 12), ('y_axis', 11), 'Right',	0.10039370078740158,	0.9107142857142857
('x_axis', 12), ('y_axis', 10), 'Right',	0.07283464566929133,	0.8409090909090909
('x_axis', 12), ('y_axis', 4), 'Right',	0.003937007874015748,	0.6666666666666666
('x_axis', 12), 'Up',	0.02631578947368421,	0.5
None, 'Down',	1.0,	0.46105072463768115

Figure A.1: The rule list of CN2 algorithm in Frozen Lake



	rule	type	coef	support	importance
43	x_axis <= 11 and x_axis > 7 and y_axis <= 8	rule	-0.424194125	0.241847826	0.181641022
40	x_axis <= 10 and x_axis > 7 and y_axis <= 9.5	rule	0.369233733	0.207427536	0.149711122
5	x_axis <= 10 and y_axis > 10	rule	-0.359912059	0.088768116	0.102362139
7	x_axis > 8 and y_axis > 11	rule	-0.47182517	0.033514493	0.08491709
28	x_axis <= 10	rule	-0.164300427	0.676630435	0.076853567
4	x_axis <= 7.5 and y_axis <= 6 and y_axis > 5	rule	-0.370451174	0.036231884	0.06922496
30	x_axis <= 8.5 and y_axis <= 6.5	rule	-0.183231068	0.144021739	0.064334585
12	x_axis <= 5 and y_axis > 3	rule	-0.147593971	0.145833333	0.052091676
24	x_axis <= 11 and y_axis > 7	rule	-0.087705601	0.565217391	0.043478161
33	x_axis <= 11 and y_axis <= 10	rule	-0.089999401	0.690217391	0.041616072
35	x_axis <= 11 and x_axis > 10 and y_axis > 7	rule	-0.115275864	0.142210145	0.040261887
15	x_axis <= 11 and x_axis > 8 and y_axis > 10.5	rule	-0.118976368	0.129528986	0.03995042
17	x_axis <= 8 and y_axis <= 10 and y_axis > 9	rule	-0.163421938	0.057971014	0.038189849
20	x_axis <= 11 and y_axis > 8	rule	-0.073743223	0.429347826	0.036501649
47	x_axis > 8 and y_axis <= 7	rule	0.105219549	0.136775362	0.036154494
8	x_axis <= 3.5 and y_axis > 7	rule	-0.199603517	0.033514493	0.035923793
50	x_axis > 8 and y_axis <= 6.5	rule	0.117326217	0.094202899	0.034272254
51	x_axis <= 11 and x_axis > 10 and y_axis <= 7	rule	0.160327146	0.045289855	0.033338288
39	x_axis > 8 and y_axis <= 11	rule	0.061556719	0.54076087	0.030675916
48	x_axis > 10	rule	0.064379206	0.323369565	0.030114174
46	y_axis <= 7	rule	0.063206134	0.30615942	0.029131503
42	x_axis <= 8 and x_axis > 3 and y_axis <= 7 and y_axis > 1	rule	0.080678792	0.129528986	0.027090688
18	x_axis <= 9.5 and y_axis <= 10 and y_axis > 7	rule	-0.061076392	0.254528986	0.026604639
9	x_axis <= 8 and x_axis > 5 and y_axis <= 1.5	rule	-0.317238505	0.00634058	0.025180779
21	x_axis <= 11 and x_axis > 5 and y_axis > 9.5	rule	-0.048058861	0.298913043	0.022000464
11	x_axis <= 11 and y_axis > 9	rule	-0.044006259	0.333333333	0.02074475

(a) Part 1



10	x_axis <= 10 and y_axis > 10.5	rule	-0.071636215	0.088768116	0.020373966
14	x_axis <= 9.5 and y_axis > 7	rule	-0.03763125	0.320652174	0.017563531
26	x_axis <= 9 and y_axis <= 10 and y_axis > 6	rule	-0.028889553	0.298913043	0.013225107
34	x_axis > 8 and y_axis > 7	rule	-0.026546958	0.4375	0.013169372
6	x_axis <= 6.5 and y_axis > 7	rule	-0.038478356	0.124094203	0.01268589
37	x_axis <= 10 and y_axis <= 7	rule	-0.02801849	0.253623188	0.012190409
19	x_axis <= 9 and y_axis <= 10 and y_axis > 5	rule	-0.025043424	0.361413043	0.012031108
36	x_axis <= 7.5 and y_axis <= 7	rule	0.034450165	0.141304348	0.012000206
3	x_axis <= 10 and y_axis > 11.5	rule	-0.070766199	0.022644928	0.010527797
29	y_axis > 7	rule	-0.018558996	0.69384058	0.008553781
54	x_axis > 11 and y_axis <= 6	rule	0.085380962	0.007246377	0.007241729
16	x_axis <= 9 and y_axis > 7	rule	-0.010811389	0.320652174	0.00504597
22	x_axis <= 11 and y_axis > 8	rule	-0.007360562	0.429347826	0.003643354
23	x_axis <= 11 and x_axis > 5 and y_axis > 9	rule	-0.007175505	0.298913043	0.003284815
27	x_axis <= 9 and y_axis <= 9 and y_axis > 7	rule	-0.008326653	0.171195652	0.003136482
45	x_axis > 11.5 and y_axis > 9	rule	0.009591483	0.085144928	0.002676956
13	x_axis <= 9.5 and y_axis > 8	rule	-0.005669681	0.230072464	0.00238625
44	x_axis > 9.5 and y_axis <= 10 and y_axis > 6	rule	0.004063845	0.238224638	0.001731187
2	x_axis <= 9 and y_axis > 11.5	rule	-0.008238987	0.022644928	0.001225704
38	x_axis > 6 and y_axis <= 11	rule	0.001933779	0.720108696	0.000868162
49	x_axis <= 8.5 and y_axis <= 7 and y_axis > 6	rule	0.003444871	0.025362319	0.000541613
52	x_axis <= 10 and x_axis > 9 and y_axis <= 9.5 and y_axis > 8	rule	0.00332509	0.015398551	0.000409424
31	x_axis <= 7.5 and y_axis <= 8	rule	0.000934607	0.192934783	0.000368798
32	x_axis <= 11	rule	-0.000875751	0.864130435	0.000300076
25	x_axis <= 11 and y_axis > 7	rule	-0.000488393	0.565217391	0.000242111
53	x_axis <= 5 and y_axis <= 3	rule	0.001190037	0.024456522	0.000183815
41	x_axis > 8 and y_axis <= 11	rule	0.000232654	0.54076087	0.00011594

(b) Part 2

Figure A.2: The rule list of Rulefit algorithm in Frozen Lake

A.2 The detailed discrete grouping of data by MDLP

(a) Feature division (I)

Pos_x	Pos_y	Vel_x	Vel_y
(-0.5260, -0.3794]	(-0.0077, -0.0011]	(-0.8075, -0.7203]	(-0.6844, -0.3679]
(-0.3794, -0.1448]	(-0.0011, -0.000506]	(-0.7203, -0.6523]	(-0.3679, -0.1831]
(-0.1448, -0.1371]	(-0.000506, -0.000504]	(-0.6523, -0.6260]	(-0.1831, -0.0059]
(-0.1371, -0.1367]	(-0.000504, -0.000079]	(-0.6260, -0.4819]	(-0.0059, -0.0004
(-0.1367, -0.0884]	(-0.000079, -0.000025]	(-0.4819, -0.1684]	(-0.0004, -0.000001]
(-0.0884, -0.0496]	(-0.000025, 0.0649]	(-0.1684, -0.0044]	(-0.000001, 0]
(-0.0496, -0.0495]	(0.0649, 0.1728]	(-0.0044, -0.0001]	(0, 0.0004]
(-0.0495, -0.0312]	(0.1728, 0.4660]	(-0.0001, 0.000097]	(0.0004, 0.0771]
(-0.0312, -0.0310]	(0.4660, 1.0987]	(0.000097, 0.0057]	(0.0771, 0.2109]
(-0.0310, -0.0069]	(1.0987, 1.3385]	(0.0057, 0.1777]	(0.2109, 0.4549]
(-0.0069, 0.0095]	(1.3385, 1.3782]	(0.1777, 0.3404]	-
(0.0095, 0.0096]	(1.3782, 1.4326]	(0.3404, 0.5080]	-
(0.0096, 0.0108]	(1.4326, 1.5042]	-	-
(0.0108, 0.0109]	-	-	-
(0.0109, 0.0324]	-	-	-
(0.0324, 0.0332]	-	-	-
(0.0332, 0.0333]	-	-	-
(0.0333, 0.0566]	-	-	-

(b) Feature division (I)

Angle	Angvel	$Contact_1$	$Contact_2$
(-0.6011-0.4160]	(-0.3034, -0.1671]	(0, 0.5]	(0, 0.5]
(-0.4160, -0.1136]	(-0.1671, -0.0796]	(0.5, 1.0]	(0.5, 1.0]
(-0.1136, -0.0056]	(-0.0796, -0.0003]	-	-
(-0.0056, 0.0003]	(-0.0003, -0.000092]	-	-
(0.0003, 0.0038]	(-0.000092, 0.000045]	-	-
(0.0038, 0.0078]	(0.000045, 0.0010]	-	-
(0.0078, 0.0273]	(0.0010, 0.0448]	-	-
(0.0273, 0.2488]	(0.0448, 0.1489]	-	-
-	(0.1489, 0.3607]	-	-

Table A.1: The divisions of features with values based on MDLP