

# Application of imitation learning in automating end-to-end exploratory data analysis

A Thesis

submitted to

Indian Institute of Science Education and Research Pune

in partial fulfillment of the requirements for the

BS-MS Dual Degree Programme

by

Devarsh Patel



Indian Institute of Science Education and Research Pune

Dr. Homi Bhabha Road,  
Pashan, Pune 411008, INDIA.

April, 2023

Supervisor: Hima Patel

Co-Supervisor: Dr. Naresh Manwani

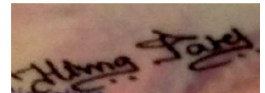
© Devarsh Patel 2023

All rights reserved



# Certificate

This is to certify that this dissertation entitled Application of imitation learning in automating end-to-end exploratory data analysis towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Devarsh Patel at IBM Research under the supervision of Hima Patel, STSM and Research Manager, IBM Research, and Dr. Naresh Manwani, Assistant Professor, IIIT Hyderabad during the academic year 2022-2023.



Hima Patel

Committee:

Hima Patel

Dr. Naresh Manwani

Dr. Amit Apte



This thesis is dedicated to Maa and Papa



# Declaration

I hereby declare that the matter embodied in the report entitled Application of imitation learning in automating end-to-end exploratory data analysis are the results of the work carried out by me at the IBM Research, Bangalore under the supervision of Hima Patel and Dr. Naresh Manwani, and the same has not been submitted elsewhere for any other degree.

A handwritten signature in black ink, appearing to read 'Devarsh Patel', with a horizontal line underneath.

Devarsh Patel





# Acknowledgments

I am deeply thankful to everyone who has given me their encouragement and assistance during this thesis. My supervisors, Hima Patel and Dr. Naresh Manwani, deserve my heartfelt appreciation for their constant support, guidance, motivation, and valuable feedback throughout this thesis. They have been excellent mentors and a role model for me, and I am grateful for their input to my work. Moreover, I acknowledge Dr. Amit Apte, my expert member, whose helpful comments and suggestions improved the quality of my work.

I would like to express my special thanks to my teammate Abhijit Manatkar, who was working with me on this thesis project. He was always supportive, helpful, and cooperative throughout the research process and provided valuable feedback and suggestions for improving the quality of the work.

I would like to acknowledge the support and the resources provided by PARAM Brahma Facility under the National Supercomputing Mission, Government of India at the Indian Institute of Science Education and Research, Pune.

My academic journey at IISER Pune would not have been possible without the constant support and encouragement of my friends. They have inspired me, motivated me, and brought joy to me. I am also grateful to my family for their unconditional love and support throughout my academic journey. My parents have always inspired me to follow my dreams and aspirations.



# Abstract

One of the open problems in data science is how to automate the end-to-end EDA process, which involves exploring the dataset, identifying patterns, outliers, and relationships among variables, and preparing the data for further analysis or modeling. Some of the existing approaches try to frame this problem as a Sequential Decision Making Problem and use Reinforcement Learning (RL) to solve it. However, a major challenge in this approach is how to define and assign rewards for each action (such as GROUP, FILTER, etc.) that is taken during the EDA process. These rewards are essential for RL to learn an optimal policy. The rewards are usually manually defined using various interestingness measures that capture how relevant or informative an action is given the current state of the analysis. However, these measures may not be able to capture all the important aspects of an action, such as its impact on subsequent actions or its alignment with the analysis goals.

We present a novel end-to-end EDA method that learns to perform data analysis tasks from human expert EDA notebooks without explicitly relying on any interestingness measures. Our method uses an imitation learning framework that learns the optimal policy for EDA by mimicking the actions of expert data analysts. Specifically, we employ generative adversarial imitation learning (GAIL) which allows our model to capture the essential aspects of data analysis in various domains. Our method can generate EDA notebooks that are comparable to human-generated ones in terms of quality and diversity.

The proposed approach is able to generate EDA sessions on different datasets that share the same schema. We evaluate our method on existing datasets for AutoEDA benchmarking and on synthetic datasets. We show that our method surpasses the current state-of-the-art end-to-end EDA method on various performance metrics and can generalize well on unseen datasets. Moreover, we show that the EDA sessions (generated using the learned model with our method) use a diverse set of interestingness measures for each step of the EDA process as a byproduct.



# Contents

<b>Abstract</b>	<b>xi</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Motivation and Analysis of Expert EDA Session . . . . .	5
1.2 Our Contributions . . . . .	8
1.3 Related Work . . . . .	9
1.3.1 Next Step Recommendation system . . . . .	10
1.3.2 Modeling user’s interestingness preferences . . . . .	11
1.3.3 Auto EDA Methods . . . . .	11
<b>2 Background information</b>	<b>15</b>
2.1 Reinforcement Learning . . . . .	15
2.2 Basic concepts and terminology . . . . .	16
2.3 Policy optimization . . . . .	18
2.3.1 Trust Region Policy Optimization . . . . .	20
2.3.2 Proximal Policy Optimization . . . . .	22
<b>3 Methodology</b>	<b>25</b>
3.1 Problem Definition . . . . .	25

3.2	Generative Adversarial Imitation Learning . . . . .	26
3.3	State and Action Space Representation . . . . .	27
3.3.1	Action Space . . . . .	27
3.3.2	State Space . . . . .	28
3.4	AdvEDA- System Description . . . . .	29
3.4.1	Policy and Discriminator Networks . . . . .	30
3.4.2	Penalties for more coherent sessions . . . . .	31
3.4.3	Training . . . . .	31
3.4.4	Policy Network initialization using Behavioural Cloning . . . . .	33
3.5	Score used for explaining model’s predictions . . . . .	34
3.5.1	Interestingness measures . . . . .	34
<b>4</b>	<b>Experimental Evaluation</b>	<b>37</b>
4.1	Dataset Details . . . . .	37
4.1.1	Cyber Security Datasets . . . . .	37
4.1.2	Synthetic datasets . . . . .	38
4.2	Experimental setup . . . . .	41
4.3	Evaluation Metrics . . . . .	42
4.4	Implementation details . . . . .	43
4.4.1	Data Preparation . . . . .	43
4.4.2	Architecture Details . . . . .	43
4.4.3	Model Initialization . . . . .	43
4.4.4	Implementation of AdvEDA . . . . .	44
<b>5</b>	<b>Results and analysis</b>	<b>45</b>

5.1	Baselines . . . . .	45
5.2	Results . . . . .	45
5.2.1	Ablation studies . . . . .	46
5.3	Analysis and Discussion . . . . .	47
5.3.1	Insights captured by generated trajectories . . . . .	47
5.3.2	Interestingness captured in trajectories . . . . .	49
5.3.3	Interestingness measure comparision analysis . . . . .	50
5.3.4	Effect of STOP action . . . . .	52
5.3.5	Effect of Penalties . . . . .	53
5.3.6	Coherency reward in ATENA . . . . .	53
<b>6</b>	<b>Conclusion</b>	<b>59</b>
6.1	Paper communicated . . . . .	60





# List of Figures

1.1	Pictorial representation of the proposed system. The input to our system is a dataset and the system outputs a sequence of operators that helps the user to explore the dataset . . . . .	6
3.1	The state representation $s$ formed by concatenating the features that are extracted from the previous three displays . . . . .	29
3.2	Overall flow of AdvEDA . . . . .	30
3.3	Training of the discriminator is done in mini-batches consisting of an equal number of expert and generated trajectories . . . . .	33
5.1	A comparison of Gold, ATENA and AdvEDA trajectories based on normalized interestingness metrics. The results show that AdvEDA can adapt to different metrics at different time steps, similar to the expert gold trajectory. ATENA, on the other hand, only focuses on the specific metrics that it uses as reward.	51
5.2	Distribution of lengths of trajectories generated by our model with STOP action	52



# List of Tables

1.1	The table shows an example of exploratory data analysis (EDA) done by an expert and the values of different metrics that measure how interesting each operation in the analysis sequence is. A notable observation from this expert-generated trajectory is that different metrics emphasize different parts of the trajectory as interesting. . . . .	7
4.1	Description of Cyber security datasets . . . . .	37
4.2	Database schema of Cyber Security datasets . . . . .	38
4.3	Trajectory distribution for training across datasets . . . . .	39
4.4	Trajectory distribution for synthetic dataset . . . . .	41
4.5	A-EDA benchmark results on Cyber and Synthetic datasets . . . . .	44
4.6	A-EDA benchmark results on Synthetic datasets . . . . .	44
5.1	EDA session without penalty score on Cyber Dataset . . . . .	46
5.2	Ablation study scores averaged out on Cyber dataset . . . . .	47
5.3	EDA session #1 generated by model on dataset 1 . . . . .	47
5.4	EDA session #2 generated by model on dataset 1 . . . . .	48
5.5	Interestingness scores on ATENA session . . . . .	49
5.6	Interestingness scores for EDA session #1 . . . . .	50
5.7	Interestingness scores for EDA session #2 . . . . .	50



# Chapter 1

## Introduction

Data scientists utilise exploratory data analysis (EDA) to examine and investigate data sets and characterise their attributes, frequently utilising various data visualisation approaches. It helps data scientists decide how to best manipulate dataset to obtain the answers they need, making it easier for them to identify patterns, detect anomalies, test hypotheses, or verify assumptions. EDA is mainly used to see what data can show beyond the formal modeling or hypothesis testing task and provides a better understanding of data set variables and the relationships between them. It can also help determine if the statistical techniques you are considering for data analysis are appropriate. One of the challenges involved is that the patterns and problems in the data vary from dataset to dataset. A data scientist often finds some patterns, develops a theory, and then has to write code to find more patterns. As a result, there is no guarantee that all of the problems and patterns in the dataset can be discovered. This stage is currently done by trial and error and varies from data scientist to data scientist.

### 1.1 Motivation and Analysis of Expert EDA Session

A possible way to design an AutoEDA system is to formulate this problem as a sequential decision-making problem and to use deep reinforcement learning to train a model that maximizes a reward function that encourages good analysis actions and discourages bad ones. Previous works define this reward function as a combination of some of the interestingness

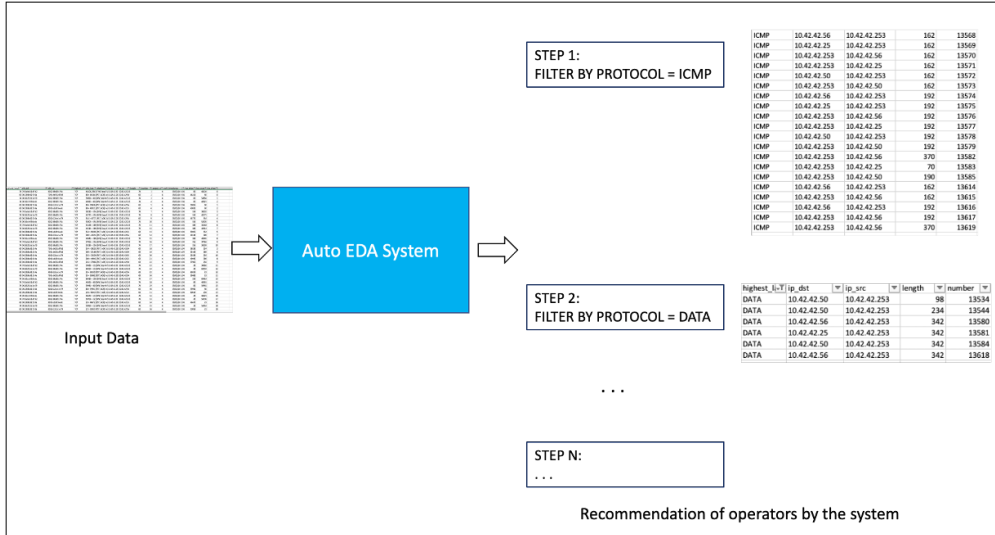


Figure 1.1: Pictorial representation of the proposed system. The input to our system is a dataset and the system outputs a sequence of operators that helps the user to explore the dataset

metrics mentioned above [1, 2]. An interestingness measure is a way of evaluating how relevant or surprising a pattern in data is for a user, depending on various criteria such as frequency, diversity, exceptionality, etc. ATENA [1] uses a mixture of metrics for diversity, compactness, and coherence to model the reward function in its RL-based method. The challenge with such a system is to be able to specify the best reward for guiding the system to generate actions that reveal all the aspects of the dataset that would be relevant to the user. Also, one has to consider the question - *Can the reward capture all the interesting aspects of the data that a user may want to explore?*

To better analyze this question and to motivate our choice of learning from expert demonstrations without using heuristic interestingness metrics performs for EDA. We use a cyber security dataset as an example and compare the actions taken by an expert with the scores obtained by different interestingness metrics. The interestingness metrics we consider are based on previous works and are explained briefly in 3.5.1:

We use an expert’s EDA session on a cyber security dataset [3] to justify our approach of learning from expert demonstrations without using heuristic interestingness metrics. We compute different interestingness metrics for each step of the EDA session and report the ones that scored above 0.8. Table 1.1 shows the sequence of expert actions and the corresponding interestingness metrics that best captured them. The interestingness metrics we considered

are from [1] and [4], and we briefly describe them in Section 3.5.1 :

Action	Interestingness captured
GROUP highest_layer AGGREGATE COUNT packet_number BACK	A-INT, Diversity -
GROUP eth_src AGGREGATE COUNT packet_number GROUP ip_src AGGREGATE COUNT packet_number	A-INT, Readability Peculiarity
FILTER info_line CONTAINS Echo (ping) reply BACK	Diversity, Coherency -
BACK	-
BACK	-
FILTER highest_layer NEQ ICMP	Coherency
GROUP tcp_srcport AGGREGATE COUNT packet_number GROUP ip_src AGGREGATE COUNT packet_number	Diversity Coherency, Peculiarity
FILTER ip_src NEQ 192.168.1.122	Coherency

Table 1.1: The table shows an example of exploratory data analysis (EDA) done by an expert and the values of different metrics that measure how interesting each operation in the analysis sequence is. A notable observation from this expert-generated trajectory is that different metrics emphasize different parts of the trajectory as interesting.

From Table 1.1, we observe the following:

- Different interestingness metrics are optimized for each step within a single expert session.
- Some actions optimize for peculiarity and readability, which are interestingness metrics that ATENA does not include in its reward definition and thus, cannot train to capture.
- Multiple interestingness metrics are maximized for some actions in the session.

The same findings emerge when we examine other expert sessions as well. Systems that rely on handcrafted rewards such as ATENA may fail to capture some important aspects of the data that are not covered by the rewards (for example, readability and peculiarity metrics in our analysis). **This is a major limitation of such systems.** We also note that we randomly selected a few metrics from the literature for our analysis. The effect we observed in our analysis could be more alarming if a wider range of metrics is chosen.

To obtain the goal of developing a meaningful AutoEDA system, systems like ATENA depend on rewards that are specific to each dataset (such as coherency reward from ATENA).

The coherency reward of ATENA is composed of several rules that are related to each dataset. Some examples of these rules are:

1. Rewarding Filtering/Grouping on certain columns that can be filtered/grouped, and penalizing applying these operations on columns that cannot be filtered or grouped.
2. Rewarding using specific operators to filter the column `info_line`.
3. Penalizing Grouping on column `highest_layer` when the display is already filtered.

(Note : More discussion about these heuristic rules are done in Section 5.3.6 where we investigate and analyse model results.)

The rules, including the ones mentioned above, indicate that the user already has a high degree of familiarity with the dataset, and using them as reward functions defeats the purpose of building an AutoEDA system in the first place. We thus conclude that it may be hard to build generic systems that handcraft some rewards, and instead, a better version would be to learn directly from the human-generated trajectories. In summary, we believe that our proposed method is well motivated by the following factors:

- Building a system that can learn directly from expert sessions how to measure interestingness for EDA, instead of trying to define a comprehensive set of metrics that capture all relevant factors.
- Systems like ATENA that use RL to learn policies need very specific reward functions, which depend on the dataset and require a lot of analysis beforehand. This defeats the purpose of an AutoEDA system. Our method avoids reward crafting and provides a complete end-to-end system for AutoEDA, without needing any intensive preliminary data analysis.

## 1.2 Our Contributions

We challenge the assumption that manually defined reward functions can capture a user’s intent. Instead, we suggest building a system that can automatically learn the policy from



human-generated training datasets. In Section 3, we present our system and we discuss our experimental results in Sections 5.2 and 5.3. The main contributions of this work are as follows:

- We introduce **AdvEDA**, a framework based on imitation learning that can automatically generate EDA sessions by learning from expert sessions. Unlike previous methods that require manual definition of reward functions to capture interestingness, our framework can learn a policy from expert examples alone. Our system also avoids the problem of having to do preliminary in-depth analysis on the dataset to hand-craft a domain specific reward as done in [1].
- Our framework can generalize across datasets with the same schema, which is an advantage over ATENA, which can only perform EDA on the dataset for which it has been trained. ATENA needs a new model for every new dataset (even with the same schema). Our framework can build one model that works across multiple datasets with the same schema, resulting in a more scalable approach. We demonstrate this on the Cyber datasets [3] (also used in ATENA) as well as on synthetically generated datasets.
- We conduct a detailed analysis of our results and show that our approach can learn a policy that performs well on various benchmarks with only a small number of expert trajectories. We also show that the EDA sessions generated by our model capture multiple aspects of that makes up a good EDA session, such as interestingness, coherence and readability. This is discussed in detail in Section 5.3.2 and 5.3.3.

### 1.3 Related Work

This section will discuss some other relevant and useful literature works that focuses on automating exploratory data analysis. We categorize them into three main groups:

### 1.3.1 Next Step Recommendation system

These systems aim to guide the user towards the next best action to take in an exploratory data analysis (EDA) process. Two main types of EDA recommendation systems have been studied: data-driven systems and log-based systems.

**Data-driven recommendations** rely on the data presented or the outcome of previous operations performed by the user. This system uses a predefined measure of interestingness to evaluate the output of a specific EDA operation, such as [6] for data-visualization, [7], [8] and [9] for finding interesting data tuple-subsets or data cube subsets, OLAP drill-down [10], and data summaries [11]. The system searches efficiently over the space of possible EDA operations and selects one (or top-k) with the highest interestingness score.

**Log-based recommendations** use historical logs of past EDA sessions by other users to suggest relevant operations. The system learns from the patterns and preferences of previous users and recommends operations that are likely to be useful for the current user. The input for systems that use log-based recommendations is the current state of the user’s EDA session, and they generate suggestions for the next exploratory operation. These systems, such as [12], [13] and [14], use a collection of previous EDA sessions from the same or different users. They assume that if two sessions have similar prefixes, then they are likely to have similar continuations. To generate recommendations for an ongoing user session, they retrieve the top-k most similar session prefixes from the collection, using a heuristic similarity measure for EDA sessions. Then, they use the continuations of the retrieved prefixes to suggest the next step in the ongoing user session.

Both data-driven and log-based approaches have limitations for providing personalized recommendations to the user. The data-driven approach relies only on the available dataset and does not account for the user’s preferences or needs. The log-based approach depends on the existence of similar past EDA sessions that may not always be present. Therefore, neither approach can guarantee optimal recommendations that suit the user’s context and goals.

The authors of [15] and [16] propose hybrid EDA recommender systems that use a session log to find similar prefixes, generate a set of possible “next-actions”, and convert them into a set of abstract actions. These abstract actions can be further transformed to concrete recommendations of EDA operations by selecting operations that have high interestingness

values according to the interestingness measures. All of these systems also use some predefined measure of score and thus may overlook some interesting views of the data if they are not captured by the user’s definition of interestingness score.

### 1.3.2 Modeling user’s interestingness preferences

A user’s preference for interestingness is not fixed, but depends on the data they are exploring and how their exploration evolves over time. The authors of [17] proposed a dynamic way of choosing an interestingness measure for each step of an EDA session, based on a kNN classifier that predicts the user’s preference among different measures. They formulated the problem as a multi-class classification task. This is a valuable contribution to the field, but it does not address the issue of recommending appropriate data-analysis operations for EDA.

To learn what data views are interesting to the user, some systems ask the user to label data views as relevant or non relevant. The system in [18] uses an active-learning approach that builds a user interest model based on the user’s feedback on the presented tuples. The model that captures user interest, improves its accuracy as more user preferences are collected. These systems require human involvement and need to be trained for each dataset. They may also need a lot of user labeling before they can be used effectively.

### 1.3.3 Auto EDA Methods

One of the most challenging tasks in EDA space is to automatically synthesize a complete sequence or trajectory of operations for a given dataset. The authors of [1] introduced ATENA, a framework that can automatically synthesize EDA session from a given dataset as input. The EDA session is shown to the user as an EDA notebook that helps them to explore the main features of the dataset. EDA is modeled as a control problem that consists of.

1. A machine-compatible EDA interface that produces numerical representations of the outcomes of exploratory operations and the parameterized, atomic composition of those operations.

2. An objective function that evaluates exploratory sessions based on three criteria: interestingness, diversity and human understandability (coherency). The interestingness criterion measures how informative and surprising the results of exploratory operations are, using a predefined notion of interestingness. The diversity criterion measures how different the results of exploratory operations are from each other, using a distance metric between sets of results. The human understandability criterion measures how coherent and interpretable the results of exploratory operations are, using a binary classifier that predicts if results are comprehensible or not.

ATENA is a deep reinforcement learning (DRL) framework for automatically generating complete EDA session which helps user to guide them through highlights of the dataset for a given input dataset. DRL is a method that combines deep learning and reinforcement learning, which enable agents to learn how to achieve their goals by using artificial neural networks. The DRL agent explores the dataset by performing different EDA operations and trains itself in a self-supervised manner itself using a reward signal from the objective function. The objective function signals the agent to generate exploratory session that is interesting, diverse, and coherent for human to understand using various interestingness measure defined literature [1], [4] as reward functions. EDA sessions generated using this approach optimize for a few heuristics only and may not be able to capture all interesting aspects of data.

However, we argue that a generic system cannot include all possible rewards that capture the interestingness of dataset, rather a better solution would be to learn from human-generated EDA sessions. Our method uses imitation learning to learn from expert demonstrations and hence, avoiding explicit definition of reward functions like ATENA. Our proposed work overcomes some challenges that ATENA faces as follows:

- ATENA defined specific reward functions to measure the interestingness of generated EDA sessions. However, this approach may not capture all the aspects of data that are interesting, since it only optimizes for the predefined heuristics. Instead of designing a universal system that incorporates all possible interestingness rewards, a better solution is to learn from human-generated sessions. Our method applies imitation learning, which is a type of supervised machine learning that learns from expert demonstrations, to mimic the human behavior in EDA sessions.

- ATENA outputs a fixed-length sequence of EDA operations for each dataset, which may not be optimal for capturing the insights of different datasets. The proposed method can generate variable-length sequences of EDA operations that are tailored to the characteristics of each dataset.
- To use ATENA for generating EDA notebooks for a new dataset, we need to train it on that specific dataset beforehand. This limits its applicability and scalability to different domains and schemas. AdvEDA is a more general system that can generate EDA notebooks for any dataset that has the same schema and domain as the one it was trained on, without requiring additional training.



# Chapter 2

## Background information

### 2.1 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning that focuses on training algorithms to interact with a specific environment and optimize their actions based on rewards. A RL algorithm, or agent, learns by exploring its environment and receiving feedback in the form of rewards or penalties for each action it takes. The goal of the agent is to maximize the total reward it can obtain in the environment.

**Reinforcement vs Supervised Learning:** Both supervised and reinforcement learning involve learning a function that maps inputs to outputs. However, the main difference is that supervised learning uses historical data with known labels to train the model, while reinforcement learning uses a reward function that provides feedback to the agent based on its actions. For instance, in a chess game, there are many possible moves that can be made at any point. Creating a labeled dataset of all the moves and their outcomes is very difficult. Therefore, it is more practical to use reinforcement learning, where the agent learns from its own experience and improves its performance over time.

**Reinforcement vs Unsupervised Learning:** Reinforcement learning involves learning a function that maps inputs to outputs based on feedback from the environment, while unsupervised learning involves finding patterns and structure in the data without any predefined outputs. For instance, if the problem is to recommend a news article to a user, a rein-

forcement learning algorithm would learn from the user’s reactions to different articles and adjust its recommendations accordingly, while an unsupervised learning algorithm would group articles based on their similarity and select one from each group.

## 2.2 Basic concepts and terminology

- Agent: RL algorithm that interacts with environment and takes actions to obtain rewards.
- Environment  $E$ : Interacting space for agent. Inputs the action taken by agent and current state and outputs rewards and next state for the agent.
- Action  $a$ : The action performed by agent based on environment state.
- Action space  $\mathcal{A}$ : Finite set of all possible actions that can be performed by agent.
- State  $s$ : Current status of environment and hold all useful information agent require to perform an action  $a$ .
- State space  $\mathcal{S}$ : Set of all possible states, agent can be present in the environment.
- Reward  $R(s_t, a_t, s_{t+1})$ : Feedback received by the agent when it perform certain action. Positive reward incentivise agent to perform similar actions and negative reward punishes agent for taking wrong action.
- Policy  $\pi(s)$ : Function that maps state space to action space  $\pi : \mathcal{A} \rightarrow \mathcal{S}$ . It can be a deterministic policy (Eq 2.1) or a stochastic policy (Eq 2.2).

$$a = \pi(s) \tag{2.1}$$

$$\pi(a|s) = P[\mathcal{A}_t = a | \mathcal{S}_t = s] \tag{2.2}$$

- Model: Representation of environment learnt by agent. It predicts next state and rewards.

$$P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | \mathcal{S}_t = s, \mathcal{A}_t = a] \tag{2.3}$$



$$R_s^a = \mathbb{E}[R_{t+1} | \mathcal{S}_t = s, \mathcal{A}_t = a] \quad (2.4)$$

- Episode (rollout): A sequence of states  $s_t$  and actions  $a_t$  for  $t$ . The agent starts in a given state of its environment and the agent observes the current state  $s_t \in \mathcal{S}$  at each timestep  $t$  and performs an action  $a_t \in \mathcal{A}$  to obtain new state  $s_{t+1}$ , that depends only on the state  $s_t$  and on the action  $a_t$ . The agent obtains a reward  $r_t$  and the agent observes the new state  $s_{t+1} \in \mathcal{S}$  and this process is looped till end of episode.
- Trajectory  $\tau$ : A sequence of state,  $s_t$ , action,  $a_t$  and reward,  $R_t$  over a set of contiguous timestamps from initial time,  $t_0$  till time associated with a certain event  $H$ ,  $t_H$

$$\tau = \{(s_t, a_t, R_t)\}_{t \in [t_0, t_H]} \quad s_t \in \mathcal{S}, a_t \in \mathcal{A} \quad (2.5)$$

- Markov Decision Process (MDP): A mathematical framework used in RL to model decision-making problems in a stochastic environment.

In an MDP, the goal of the agent is to learn a policy that maximizes the expected cumulative reward over time. A policy is a function that maps states to actions, and it tells the agent what action to take in each state. RL algorithms use the MDP framework to learn an optimal policy by repeatedly interacting with the environment, observing rewards and transitions, and updating their estimate of the optimal policy.

The mathematical formulation of an MDP can be represented as a tuple  $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ , where:

- $\mathcal{S}$  is set of states
- $\mathcal{A}$  is set of actions
- $P$  is the transition probability function  $P_{ss'}^a = \mathbb{P}[S_{t+1} = s' | \mathcal{S}_t = s, \mathcal{A}_t = a]$
- $R$  is the reward function
- $\gamma$  is the discount factor

The goal of RL algorithms is to learn the optimal policy  $\pi^*$  that maximizes the expected cumulative reward, which can be represented as:

$$V^\pi(s) = \mathbb{E}_\pi \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s \right] \quad (2.6)$$

where  $V^\pi(s)$  represents the expected overall value from state  $s$ , under a given policy from state, and  $\gamma$  is the discount factor. The optimal policy  $\pi^*$  is defined as the policy that maximizes  $V^\pi(s)$  for all states  $s$ .

RL algorithms use iterative methods, such as value iteration or policy iteration, to estimate the optimal value function  $V^{\pi^*}(s)$  and the optimal policy  $\pi^*$ . Once the optimal policy is learned, the agent can use it to take actions in the environment and maximize its cumulative reward over time.

- Q-value  $Q(s, a)$ : It is the "state action" value function, also known as the quality function and represents the value of performing a certain action in a given state. It is calculated as expected return starting from state  $s$ , taking action  $a$ , then following policy  $\pi$

$$Q^\pi(s, a) = E_\pi\{R_t | s_t = s, a_t = a\} = E_\pi\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right\} \quad (2.7)$$

- Advantage  $A(s, a)$ : For a given policy,  $\pi$  and state,  $s$ , the advantage indicates the difference between the expected cumulative rewards for a specific action and overall expectation value.

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (2.8)$$

## 2.3 Policy optimization

Policy optimization is a class of reinforcement learning algorithms that aim to find the optimal policy for an agent interacting with an environment. Rather than estimating the value function and deriving a policy from it, these algorithms directly optimize the policy function of an agent. Policy optimization methods can be divided into two categories: policy iteration methods, which alternate between evaluating and improving a policy, and policy gradient methods, which directly optimize a parameterized policy using gradient ascent. Policy gradient methods are popular because they can handle high-dimensional and continuous action spaces, and can be combined with function approximation techniques such as deep neural networks.

In mathematical terms, policy optimization algorithms aim to find the policy function  $\pi$  that maximizes the objective function  $J(\pi)$ , which is defined as:

$$J(\pi) = \mathbb{E}_\pi \left[ \sum_t^\infty \gamma^t r_t \right] \quad (2.9)$$

where  $\mathbb{E}_\pi$  represents the expected value under policy  $\pi$ ,  $\gamma$  is the discount factor,  $r_t$  is the reward received at time step  $t$ , and  $\sum_t^\infty \gamma^t r_t$  is the discounted cumulative reward.

The objective of policy optimization is to maximize  $J(\pi)$  by updating the parameters of the policy function. The policy function is typically represented as a parametric function  $\pi_\theta$  that takes a state  $s$  as input and outputs a probability distribution over actions  $a$ . The parameters  $\theta$  of the policy function are updated through gradient ascent on the objective function  $J(\pi_\theta)$ .

The update rule for policy optimization can be derived using the policy gradient theorem, which states that the gradient of the objective function  $J(\pi_\theta)$  with respect to the policy parameters  $\theta$  can be expressed as:

$$\nabla_\theta J(\pi_\theta) = E_\pi [\nabla_\theta \log \pi_\theta(a|s) Q_\pi(s, a)] \quad (2.10)$$

where  $Q_\pi(s, a)$  is the state-action value function under policy  $\pi$ , and  $\nabla_\theta \log \pi_\theta(a|s)$  is the derivative of the logarithm of the policy function with respect to the policy parameters.

The policy gradient theorem suggests that the policy parameters should be updated by taking a step in the direction of the gradient of the objective function with respect to the policy parameters:

$$\theta_{new} = \theta_{old} + \alpha \nabla_\theta J(\pi_\theta) \quad (2.11)$$

where  $\alpha$  is the learning rate, which determines the step size of the parameter update.

Policy optimization algorithms typically use variants of this update rule, such as the trust region policy optimization (TRPO) or the proximal policy optimization (PPO) algo-

rithm, which incorporate additional constraints or penalties to ensure stable and efficient convergence to the optimal policy.

### 2.3.1 Trust Region Policy Optimization

Trust Region Policy Optimization (TRPO) [19] is a popular reinforcement learning algorithm for optimizing policies in continuous control tasks. It is designed to optimize the expected cumulative reward of a policy in an environment while ensuring that the policy updates are stable and do not deviate too far from the current policy.

The algorithm is based on the idea of a trust region, which is a region around the current policy that is considered safe to explore. The size of the trust region determines how much the policy can change in each update, and it is typically chosen to be small enough to ensure stability, but large enough to allow for significant improvements.

TRPO uses a constrained optimization approach to ensure that the policy updates stay within the trust region. Specifically, it solves a constrained optimization problem to find the policy update that maximizes the expected reward subject to a constraint on the distance between the new policy and the old policy.

The main advantage of TRPO is that it guarantees monotonic improvement in the expected reward, meaning that the performance of the policy is guaranteed to improve with each update. This is because the algorithm maximizes the expected reward subject to a constraint on the distance between the old and new policies, which ensures that the new policy cannot be worse than the old policy.

The objective of TRPO is to maximize the expected reward under the new policy  $\pi_\theta$ , where  $\theta$  represents the policy parameters. This is expressed as:

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (2.12)$$

where  $s_t$  and  $a_t$  represent the state and action at time step  $t$ ,  $r(s_t, a_t)$  is the reward received for taking action  $a_t$  in state  $s_t$ , and  $\gamma$  is the discount factor.

The policy update is performed by solving the following constrained optimization problem:

$$\begin{aligned} & \max_{\theta'} L(\theta, \theta') \\ & \text{subject to } D_{\text{KL}}(\theta || \theta') \leq \delta \end{aligned}$$

where  $L(\theta, \theta')$  is the surrogate objective function defined as:

$$L(\theta, \theta') = \mathbb{E}_{s, a \sim \pi_{\theta}} \left[ \frac{\pi_{\theta'}(a|s)}{\pi_{\theta}(a|s)} A_{\pi_{\theta}}(s, a) \right] \quad (2.13)$$

where  $\pi_{\theta'}(a|s)$  is the probability of taking action  $a$  in state  $s$  under the new policy,  $\pi_{\theta}(a|s)$  is the probability of taking action  $a$  in state  $s$  under the old policy, and  $A_{\pi_{\theta}}(s, a)$  is the advantage function, which estimates the advantage of taking action  $a$  in state  $s$  under the old policy compared to taking the mean action.

The KL-divergence constraint  $D_{\text{KL}}(\theta || \theta')$  ensures that the new policy is close enough to the old policy to maintain stability. The trust region size  $\delta$  determines the maximum distance between the old policy and the new policy, and it is typically chosen to be small enough to ensure stability but large enough to allow for significant improvements.

The constrained optimization problem is solved using the conjugate gradient method, which is a fast and efficient method for solving large-scale optimization problems. The solution of the optimization problem yields the updated policy parameters  $\theta'$ , which are used to update the policy.

The main advantage of TRPO is that it guarantees monotonic improvement in the expected reward, meaning that the performance of the policy is guaranteed to improve with each update. This is because the algorithm maximizes the expected reward subject to a constraint on the distance between the old and new policies, which ensures that the new policy cannot be worse than the old policy. However, TRPO can be computationally expensive due to the need to solve a constrained optimization problem at each iteration. Additionally, the size of the trust region can be difficult to choose, and a too-small trust region may result in

slow progress, while a too-large trust region can lead to instability and poor performance.

Despite these challenges, TRPO remains a popular and effective algorithm for optimizing policies in continuous control tasks, and it has been used in a variety of applications, including robotics, autonomous vehicles, and game playing.

### 2.3.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [20] is a state-of-the-art policy gradient method that operates by collecting experience using the current policy, updating the policy based on the collected experience, and repeating the process. PPO improves upon previous methods such as Trust Region Policy Optimization (TRPO) by being simpler to implement, more general, and having better sample efficiency. PPO works by alternating between sampling data through interaction with the environment, and optimizing a surrogate objective function using stochastic gradient ascent. The surrogate objective function is designed to ensure that the new policy does not deviate too much from the old policy, while still maximizing the expected return.

The objective of PPO is to maximize the expected cumulative reward of the policy, which is given by:

$$J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (2.14)$$

where  $\theta$  represents the policy parameters,  $s_t$  and  $a_t$  represent the state and action at time step  $t$ ,  $r(s_t, a_t)$  is the reward received for taking action  $a_t$  in state  $s_t$ , and  $\gamma$  is the discount factor.

The PPO algorithm uses a surrogate objective function that approximates the true objective function, making it easier to optimize the policy. The surrogate objective function is defined as:

$$L(\theta) = \mathbb{E}_{s, a \sim \pi_{\theta}} \left[ \frac{\pi_{\theta}(a|s)}{\pi_{\theta, \text{old}}(a|s)} A_{\text{clip}}(s, a) \right] \quad (2.15)$$

$$\mathcal{L}_{\pi_{\theta}}^{\text{CLIP}}(\pi_{\theta_k}) = \mathbb{E}_{\tau_{\pi_{\theta}}} \left[ \sum_{t=0}^T \left[ \min \left( \rho_t(\pi_{\theta}, \pi_{\theta_k}) A_t^{\pi_{\theta_k}}, \text{clip}(\rho_t(\pi_{\theta}, \pi_{\theta_k}), 1 - \epsilon, 1 + \epsilon) A_t^{\pi_{\theta_k}} \right) \right] \right] \quad (2.16)$$

where  $\pi_{\theta}(a|s)$  is the probability of taking action  $a$  in state  $s$  under the current policy,  $\pi_{\theta_{\text{old}}}(a|s)$  is the probability of taking action  $a$  in state  $s$  under the old policy (which is the policy used to collect the experience), and  $A_{\text{clip}}(s, a)$  is a clipped version of the advantage function, defined as:

The policy update in PPO is performed by solving a constrained optimization problem that maximizes a clipped surrogate objective function:

$$\theta' = \arg \max_{\theta'} \mathbb{E} \pi_{\theta} \left[ \frac{\pi_{\theta'}(a_t|s_t)}{\pi_{\theta}(a_t|s_t)} A_t^{\text{clip}}(\theta, \theta') \right],$$

where  $a_t$  and  $s_t$  are the action and state at time step  $t$ , and  $A_t^{\text{clip}}(\theta, \theta')$  is the clipped advantage function, which is defined as:

where  $r_t(\theta)$  is the estimated advantage of taking action  $a_t$  in state  $s_t$  under the old policy,  $\epsilon$  is a hyperparameter that controls the degree of clipping, and  $\text{clip}(x, a, b)$  clips  $x$  to the range  $[a, b]$ .

The clipped surrogate objective function ensures that the policy updates are stable and do not deviate too far from the current policy. Specifically, if the ratio of probabilities  $\frac{\pi_{\theta'}(a_t|s_t)}{\pi_{\theta}(a_t|s_t)}$  is greater than  $1 + \epsilon$ , then the objective function is clipped to  $1 + \epsilon$ . If the ratio is less than  $1 - \epsilon$ , then the objective function is clipped to  $1 - \epsilon$ . Otherwise, the objective function is not clipped.

The clipped surrogate objective function is maximized subject to a constraint on the distance between the old and new policies:

$$\text{subject to } D_{\text{KL}}(\theta, \theta') \leq \delta \quad (2.17)$$

where  $D_{\text{KL}}(\theta, \theta')$  is the KL-divergence between the old and new policies, and  $\delta$  is a hyperparameter that controls the size of the trust region.

The optimization problem is solved using stochastic gradient ascent, which updates the policy parameters  $\theta$  in the direction of the gradient of the clipped surrogate objective function. The trust region size  $\delta$  is updated dynamically to ensure that the policy updates stay within a reasonable range.

PPO is a computationally efficient algorithm that has been shown to achieve state-of-the-art performance in a variety of continuous control tasks. The clipped surrogate objective function and the KL-divergence constraint ensure that the policy updates are stable and do not deviate too far from the current policy. This makes PPO more sample-efficient than other reinforcement learning algorithms that do not have such constraints, such as vanilla policy gradient. Overall, PPO is a powerful reinforcement learning algorithm that is widely used in practice due to its efficiency and stability. Its combination of a clipped surrogate objective function, a KL-divergence constraint, and a value function make it well-suited for optimizing policies in continuous control tasks, where stability and sample efficiency are crucial.



# Chapter 3

## Methodology

In this section, we describe our proposed system to overcome the limitations observed in the ATENA system. Our key objective is to train the system to learn and mimic humans by teaching it straight from human-generated trajectories. We also discuss the difficulties posed by the fact that the trajectory lengths can vary.

### 3.1 Problem Definition

The goal is to learn a policy that can generate EDA trajectories given a dataset as input. We model this problem as a reinforcement learning (RL) task. We next define the state space  $\mathcal{S}$  and action space  $\mathcal{A}$  of the underlying Markov Decision Process (MDP) as follows. Action space  $\mathcal{A}$  is the set of possible actions in the EDA process, which are `GROUP`, `FILTER`, `BACK`, and `STOP`. While we pick these actions to be consistent with ATENA to compare and contrast, our framework is not restricted to just these actions. State space  $\mathcal{S}$  is the space of all possible tabular datasets which can be derived from the original dataset by performing a sequence of actions. We will describe more details of the  $\mathcal{S}$  and  $\mathcal{A}$  later. In this MDP, we do not have access to the reward structure. The reason is that at any step in the exploratory data analysis, we generally do not explicitly receive feedback about the appropriateness of the action taken. Thus, we want to learn the optimal policy without knowing the rewards.

Multiple approaches to learning optimal policy without access to the reward structure

have been proposed in the RL literature. Some of the techniques used to learn optimal policy when we do not know the reward structure are behavioural cloning [21], imitation learning [22], inverse RL [23, 24, 25]. To learn optimal policy, all of these approaches necessitate expert trajectories.

To learn an optimal policy for an end-to-end auto EDA, the proposed study employs an imitation learning approach known as Generative Adversarial Imitation Learning (GAIL). Let `Exp_EDA` be a set of EDA sessions by human experts. defined as  $\text{Exp\_EDA} = \{\tau_1, \tau_2, \dots, \tau_N\}$  where each trajectory,  $\tau_i$  is a sequence of state-action pairs  $\{(s_1, a_1), (s_2, a_2) \dots (s_T, a_T)\}$  such that  $s_i \in \mathcal{S}$  and  $a_i \in \mathcal{A}$ .

The learning objective to find a policy  $\pi_\theta$  parameterized by  $\theta$  that closely resemble the underlying expert policy  $\pi_E$ . That is, for any state,  $s \in \mathcal{S}$ ,  $\pi_\theta(s)$  outputs a distribution over the action set  $\mathcal{A}$  such that action  $a$  sampled from this distribution, closely is quite similar to the action  $a$  that a expert data analyst would take in  $s$ .

The next section discuss about how generative adversarial imitation learning (GAIL) is used to model the end-to-end EDA process.

## 3.2 Generative Adversarial Imitation Learning

Generative Adversarial Imitation Learning (GAIL) [26] is a state-of-the-art imitation learning method that learns a policy directly from expert demonstrations without any reward signal. It treats the problem as two-player min-max game. The method involves two neural networks: a policy network  $\pi_\theta$  with parameters  $\theta$  and a discriminator  $D_w$  with parameters  $w$ . The goal is to make  $\pi_\theta$  imitate the expert policy  $\pi_E$  by finding a saddle point  $(\pi, D)$  of the following objective function that resembles the one used in generative adversarial networks:

$$\mathbb{E}_\pi[\log(D(s, a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s, a))] - \lambda H(\pi) \tag{3.1}$$

where  $H(\pi)$  is the entropy of the policy  $\pi$ . The discriminator and the policy network is trained alternately. The discriminator is trained to maximize Eq 3.1, which means it tries to distinguish between the expert and the policy trajectories. The policy is trained to minimize Eq 3.1, which means it tries to fool the discriminator by imitating the expert

behavior.

GAIL is known to be very sample efficient regarding the number of expert demonstrations required, and this is ideal for our case as the number of expert demonstrations we seek to learn from is not very large. Significant challenges are associated with collecting expert demonstrations for the EDA problem setting.

One of the advantages of GAIL is that it does not require a large amount of expert demonstrations to learn an optimal policy, which is suitable for our scenario where we have a limited number of expert demonstrations to imitate and collecting expert demonstrations for complex tasks such as EDA is still challenging and costly.

GAIL is a suitable method for our problem setting because it can capture long-term dependencies in the expert behavior better than methods like simple Behavioural Cloning (BC). BC learns a supervised mapping between states and actions, but it ignores the long-term effects of actions on the state distribution. This can lead to co-variate shift and error accumulation if there is not enough data. GAIL solves this problem by matching the state distribution of the agent and the expert using a generative adversarial framework.

In following section, we present how we model the state and action spaces for the EDA problem and how we use a GAIL-based algorithm to learn from expert demonstrations

### 3.3 State and Action Space Representation

We use a similar approach to ATENA to define the state,  $\mathcal{S}$  and action spaces,  $\mathcal{A}$  for the EDA problem.

#### 3.3.1 Action Space

The action space includes the following actions:

1. `GROUP(grp_col, agg_col, agg_func)`: This action performs a grouping operation on the `grp_col` and computes the aggregation function `agg_func` on the `agg_col`. The possible values for `agg_func` are `SUM`, `COUNT`, `MEAN`, `MIN`, `MAX`.

2. `FILTER(filter_col, filter_func, filter_term)`: This action applies a filtering operation on the dataset by checking if the condition given by the operator `filter_func` (`=`, `≠`, `CONTAINS`, `STARTS_WITH`, `ENDS_WITH`) and the term `filter_term`, which is either a numeric or textual value from the `filter_col`, is satisfied.
3. `BACK()`: This action allows the agent to undo the last action and return to the previous dataset display in the current analysis session.
4. `STOP()`: This action allows the agent to indicate that it has finished the current analysis session.

We introduce a new action called `STOP` in addition to the existing actions of `FILTER`, `GROUP` and `BACK` that are supported by ATENA. The `STOP` action enables the agent to end the current EDA session when it does not find any more actions that can lead to interesting explorations. This is useful because otherwise the agent would keep generating actions that either repeat previous explorations or show nothing insightful to the user. Without an action to indicate the end, an agent would continue to generate actions, which would either go down exploration paths that have already been explored or have nothing insightful to show to the user.

### 3.3.2 State Space

For any display obtained by employing a sequence of actions from  $\mathcal{A}$ , we create a display vector  $\vec{d}$  with these features extracted from the display:

1. For each attribute (column) in the display, we include features that measure number of nulls values, number of unique values and entropy of that attribute
2. For each attribute, we include a feature to determine whether that column is grouped by some value, aggregated by some function or neither of those.
3. Three global features: number of groups, mean size of groups, and variation of size of groups.

To represent a state  $s$ , we concatenate the display vectors of the three most recent displays in the EDA session. If the EDA session has not included three displays yet, then display vectors are padded with 0.

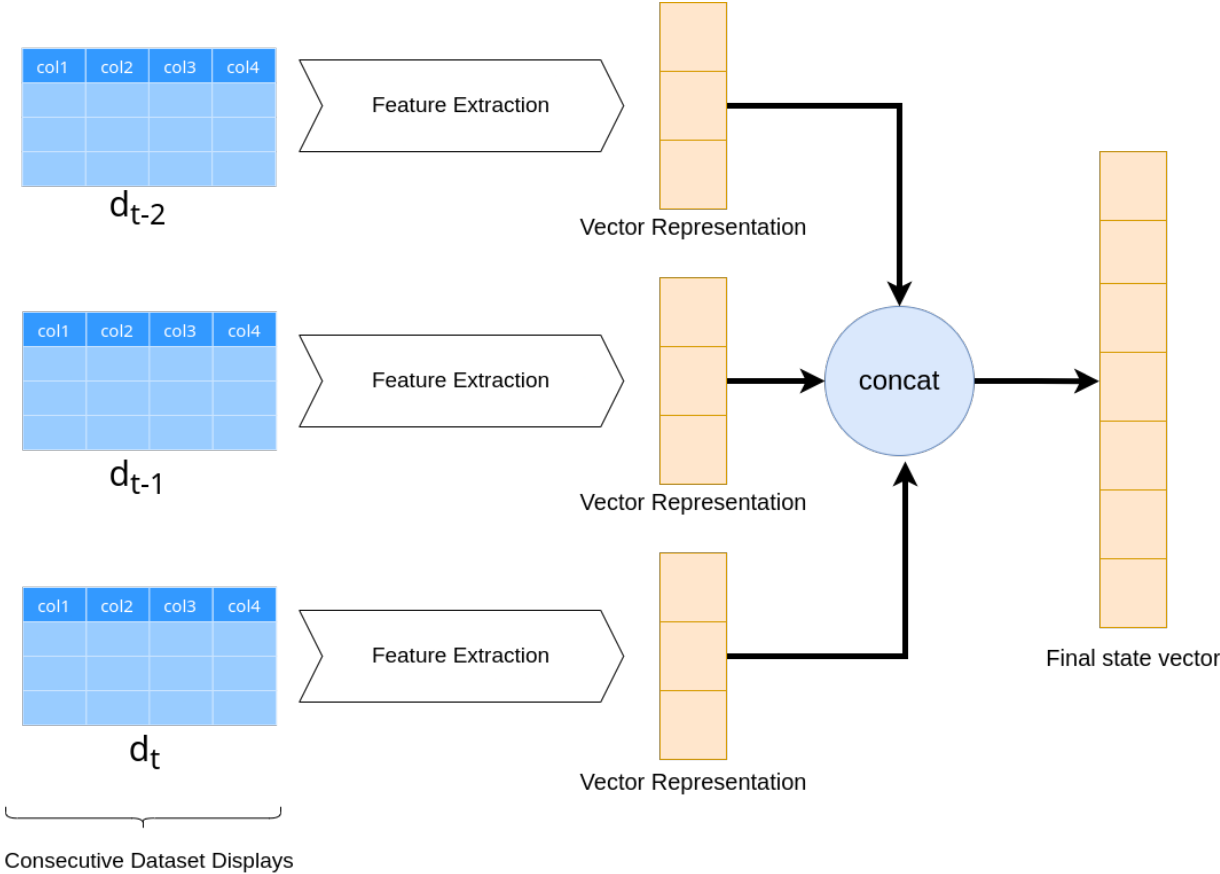


Figure 3.1: The state representation  $s$  formed by concatenating the features that are extracted from the previous three displays

### 3.4 AdvEDA- System Description

We build on the previous work of GAIL with some modifications to suit the EDA problem setting better. Figure 3.2 shows the overall architecture and the flow diagram of our proposed approach AdvEDA.

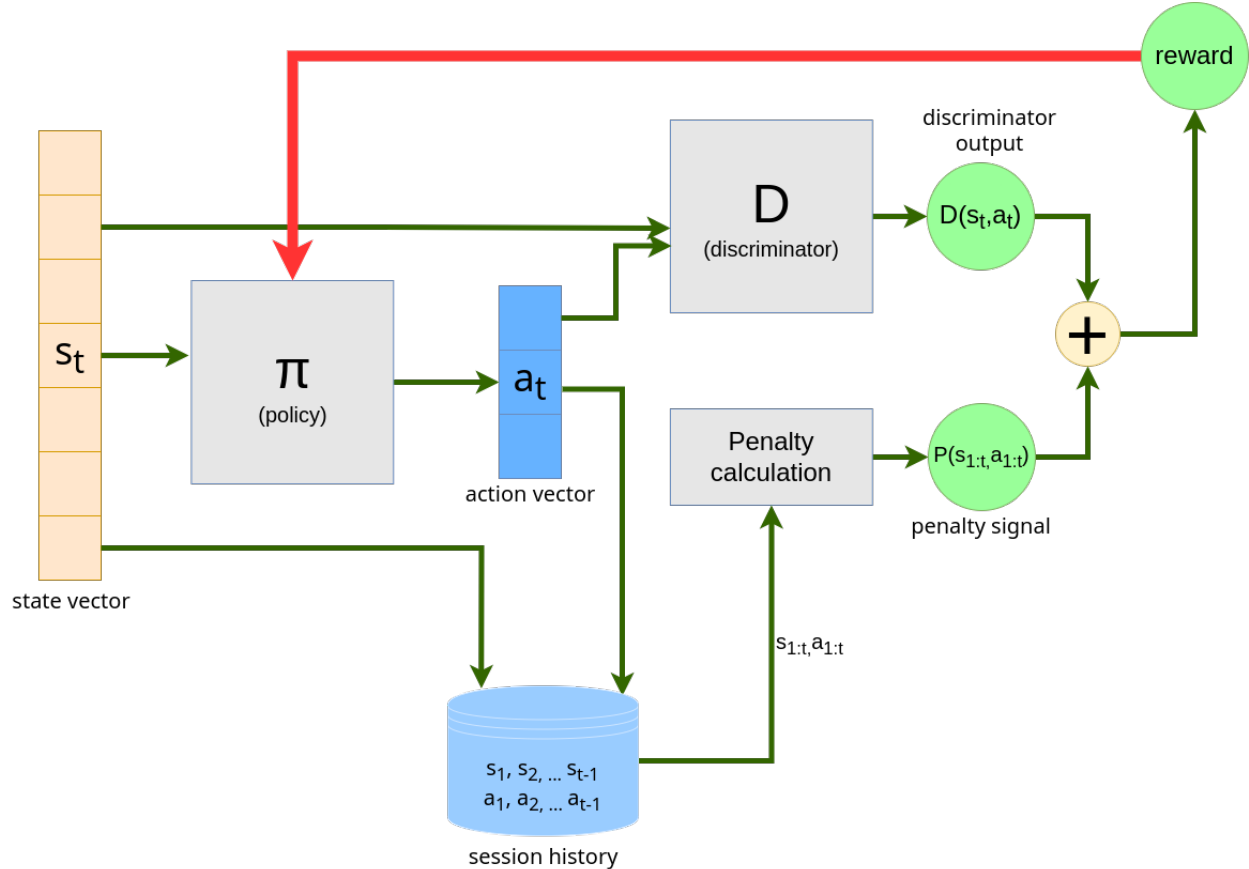


Figure 3.2: Overall flow of AdvEDA

### 3.4.1 Policy and Discriminator Networks

Our method is similar to GAIL in that it consists of a policy network  $\pi_\theta$  and a discriminator network  $D_w$ . The policy network outputs a distribution over the actions given a state representation  $s$  as input. The discriminator assigns a value between 0 and 1 to a state-action pair, indicating how likely it is that the pair comes from existing expert trajectories rather than from the policy network.

The output of the discriminator can be represented as:

$$D_w(s, a) = \sigma(L_w(s, a)) \quad (3.2)$$

where  $L$  is the unnormalized output of the final layer of  $D$  which is then passed through the sigmoid function  $\sigma$  to map it to the interval  $[0, 1]$ .

### 3.4.2 Penalties for more coherent sessions

We propose introduction of penalties to improve the coherence of our model’s output trajectories. These penalties are added to the reward signal that the policy network receives. Their aim is to prevent the model from producing actions that are not suitable for an EDA setup, such as

1. Performing a **BACK** action at the beginning of the analysis
2. Repeating the same **FILTER/GROUP** action consecutively

The model is penalized when it outputs actions that indicate uncertainty. We define such actions as those that involve applying **FILTER/GROUP** operations to analyze the data, followed by taking the **BACK** operation to undo them. We hypothesize that these actions show a lack of confidence in the analysis steps.

Mathematically, the penalties are defined as follows:

$$P(s_{1:t}, a_{1:t}) = \begin{cases} -1.0 & \text{if } a_t = \text{BACK and } s_t = s_1 \\ -1.0 & \text{if } a_t \neq \text{BACK and } a_t = a_{t-1} \\ -1.0 \times l & \text{if } a_t = a_{t-2} \cdots = a_{t-2l} = \\ & \text{BACK} \neq a_{t-2(l+1)} \text{ and} \\ & a_{t-1}, a_{t-3} \dots, a_{t-2l-1} \in \\ & \{\text{FILTER, GROUP}\} \text{ and } l > 1 \\ 0.0 & \text{otherwise} \end{cases} \quad (3.3)$$

### 3.4.3 Training

We adopt an alternating training strategy for the policy and discriminator networks, which is akin to GAIL. However, unlike GAIL, where the policy network was optimized using Trust Region Policy Optimization (TRPO) with a cost function, we use Proximal Policy Optimization (PPO) with reward signals to train our policy network. PPO and TRPO are both methods for optimizing a reinforcement learning agent with an actor-critic architecture,

but PPO has a better sample efficiency than TRPO. The overall training approach is outlined in Algorithm 1.

---

**Algorithm 1** AdvEDA- Learning a policy for performing EDA from expert demonstrations

---

- 1: **Input:** Expert trajectory data (Exp\_Traj), initial policy and discriminator parameters  $\theta_0$  and  $w_0$ .
- 2: Pre-train the policy network parameterized by  $\theta_0$  using Behavioural Cloning using state-action pairs from Exp\_Traj.
- 3: **for**  $i=1,2,\dots$  **do**
- 4:   Sample trajectories  $\tau_i \sim \pi_{\theta_i}$
- 5:   Update discriminator parameters  $w_i$  to  $w_{i+1}$  by taking a gradient ascent step with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(1 - D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(D_w(s, a))]$$

- 6:   Update policy parameters  $\theta_i$  to  $\theta_{i+1}$  using a PPO clipped optimizer considering the reward signal

$$r_t = -\log(1 - D_w(s_t, a_t)) + P(s_{1:t}, a_{1:t})$$

at every time-step  $t$  for all  $\tau_i$

- 7: **end for**
- 

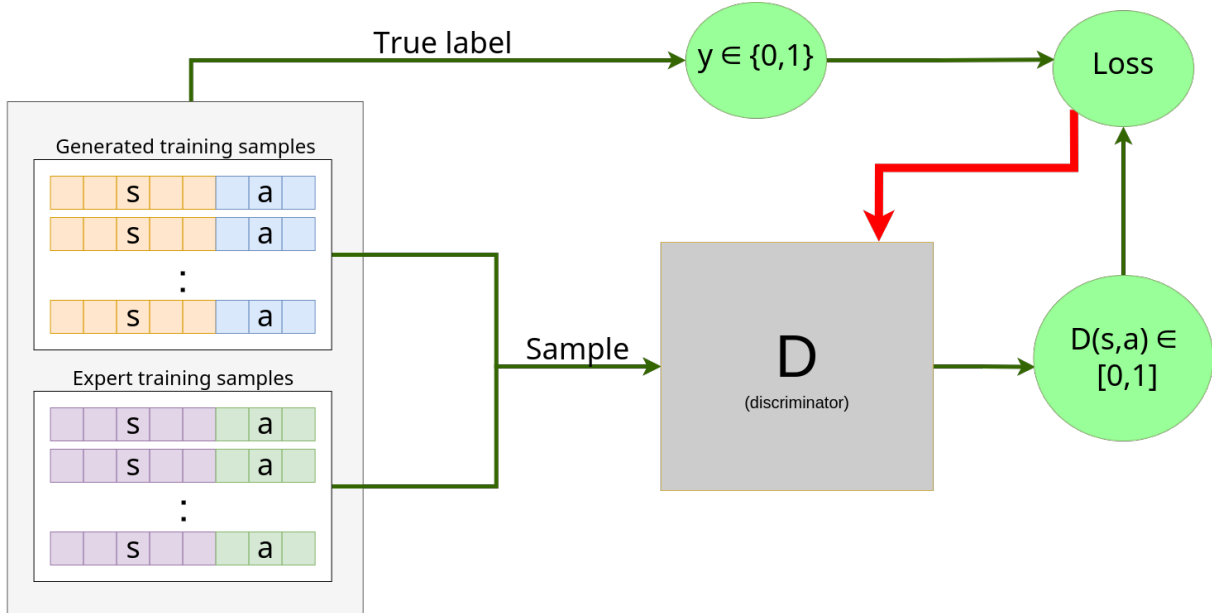
**Training of Policy Network:** To train the policy network, we sample multiple trajectories using the current policy  $\pi_{\theta}$  in each iteration. Since there are no rewards defined in the underlying MDP, we have to provide a reward signal that can help us update the policy network to make  $\pi_{\theta}$  closer to  $\pi_E$  while keeping some coherency constraints. For any sampled trajectory, our reward signal at time step  $t$  is as follows:

$$r_t = -\log(1 - D_w(s_t, a_t)) + P(s_{1:t}, a_{1:t}) \tag{3.4}$$

The first term  $-\log(1 - D_w(s_t, a_t))$  is a function that increases as the discriminator outputs increase. This means that it captures how confident the discriminator is that the policy’s action is similar to what an expert would do. The second term  $P(s_{1:t}, a_{1:t})$  is a penalty term that discourages the policy from taking actions that are non-sensical or uncertain. A PPO update is made with this modified reward signal at each time step of every sampled trajectory.

**Training of Discriminator Network:** The discriminator network differentiate between state-action pairs generated by our policy and state-action pairs from expert demonstrations





Mini-batch of training examples

Figure 3.3: Training of the discriminator is done in mini-batches consisting of an equal number of expert and generated trajectories

at each iteration. We sample a mini-batch of equal number of state-action pairs from both sources ( $\tau_i$  and  $\tau_E$ , respectively) and train the discriminator to classify them as generated or expert samples. We use Binary Cross-Entropy Loss as our loss function and the objective function for the discriminator is given by:

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(1 - D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(D_w(s, a))] \quad (3.5)$$

### 3.4.4 Policy Network initialization using Behavioural Cloning

We adopt a two-stage approach to train our policy network in imitation learning. We initialize the policy network with behavioral cloning before applying GAIL as an adversarial training method. Behavioral cloning is a supervised learning technique that aims to learn a policy from expert demonstrations by maximizing the likelihood of matching the expert actions given the same states. This initialization strategy can provide a "better" initial policy than random initialization, which may help with faster convergence. We conduct an

ablation study to investigate the impact of this initialization choice.

## 3.5 Score used for explaining model’s predictions

Taking inspiration from data-driven EDA systems discussed in Section 1.3, we use various measures, i.e., diversity, coherency, readability, and peculiarity, along with interestingness measures, to evaluate the EDA operations produced by the model. Interestingness, diversity, and coherency are adopted from ATENA.

### 3.5.1 Interestingness measures

1. **A-INT**: This is the *Interestingness score* defined in ATENA. It is defined separately for the case of a **GROUP** operation and a **FILTER** operation:

- (a) Interestingness Score for a **GROUP** operation: This is a score based on *conciseness measures* [27, 4] that consider compact group-by views covering many rows as informative and easy to understand and assigns them a higher reward. This measure considers the number of groups  $g$ , the number of grouped attributes,  $a$ , and the number of tuples,  $r$ . Score is calculated as

$$h_1(g.a)/h_2(r) \tag{3.6}$$

where  $h_1$  and  $h_2$  are normalized sigmoid function with fixed width and center.

- (b) Interestingness Score for a **FILTER** operation: To quantify the interestingness of a **FILTER** operation, *exceptionality* [8, 28, 6] of filtered rows (generated after applying the operation) is compared with those of the unfiltered table. The Kullback-Leibler (KL) [29] divergence on each column is used to measure how much the filtered data view differs from the unfiltered view.

KL divergence is a measure of how one probability distribution is different from another. Value probability distribution  $P_t^A$  of an attribute  $A \in Attr$ , is the relative frequency of its values in  $d_t$ . If  $d_t$  is grouped,  $Attr$  is a set of aggregated

attributes else it is a set of all attributes. The score is given by

$$h(\max_{A \in Attr} D_{KL}(P_{t-1}^A, P_t^A)) \quad (3.7)$$

where  $h$  is the sigmoid function.

This score highly rewards those FILTER operations whose resultant views deviate from the unfiltered views.

2. **Diversity score:** This metric, also from [1], favors a display (data view) that highlights parts of the dataset that are different from those seen in any of the previous displays in the session so far. It is calculated as the minimum Euclidean distance between the vectorized representation (observation vector,  $d_t$ ) of the resultant view and the vectorized representations of all previous views.

$$\min_{0 \leq t' < t} \delta(d_t, d_{t'}) \quad (3.8)$$

3. **Coherency score :** This score, also from [1], is a highly detailed metric determined by a set of hand-crafted rules which assign each view a penalty or reward based on whether the action performed at the current step is coherent with previous actions. The rules set consists of two types: (1) Rules that apply to any generic EDA session. For e.g., an operation that results in an empty or unchanged view is considered incoherent. (2) Rules specific to the dataset’s domain to be explored. In [1], many such rules are defined for the Cyber datasets [3] being analyzed. These rules are highly specific and are carefully hand-tuned to predict coherency for actions taken on the Cyber dataset.
4. **Readability score:** This metric builds on top of compaction gain from [27]. Let compact display score,  $C_{d_t}$  for current display, dt with number of groups, g defined as

$$h_1(g \cdot |d_t|) \quad (3.9)$$

where  $h_1$  is normalized sigmoid function with fixed width and center. Readability gain,  $RG_{d_t}$  is defined

$$1 - C_{d_{t-1}}/C_{d_t} \quad (3.10)$$

Readability gain measures changes the compact display score from before and after taking current EDA operation. We favour compact displays at each of EDA step and

thus readability gain captures this idea. Readability score is obtained by

$$h_2(1 - RG_{d_t} C_{d_t}) \tag{3.11}$$

where  $h_2$  is normalized sigmoid function with fixed width and center.

5. **Peculiarity score:** This score helps quantify patterns that are anomalous. We have adopted a deviation-based measure from [6] which favors display with a high difference from reference display (given dataset  $d_0$  in our implementation). It is calculated by taking KL divergence of  $P_t^A$  defined above, which denotes the value probability distribution of display  $d_t$ , and  $P_0^A$  which is the value probability distribution of reference display  $d_0$ .

# Chapter 4

## Experimental Evaluation

### 4.1 Dataset Details

We conduct experiments on two types of datasets to evaluate the performance of proposed method, which generates EDA sessions using training data.

#### 4.1.1 Cyber Security Datasets

The "cyber security datasets" are four distinct and mutually exclusive datasets as detailed in Table 4.1 that were obtained from the HoneyNet Project [3]. The database schema for all cyber datasets is identical as shown in Table

<b>Dataset</b>	<b>No of rows</b>	<b>Underlying insight</b>
Cyber 1	8648	ICMP scan on IP range
Cyber 2	348	Remote code execution attack
Cyber 3	745	Web based phishing attack
Cyber 4	13625	TCP port scan

Table 4.1: Description of Cyber security datasets

Column	Data type
captured_length	int64
eth_dst	string
eth_src	string
highest_layer	string
info_line	string
interface_captured	float64
ip_dst	string
ip_src	string
length	int64
number	int64
project_id	int64
sniff_timestamp	string
tcp_dstport	float64
tcp_srcport	float64
tcp_stream	float64

Table 4.2: Database schema of Cyber Security datasets

For each dataset, we generated human trajectories in two different ways:

1. **REACT Dataset:** The authors of [16] collected and curated the REACT dataset, which consists of exploratory data analysis (EDA) session traces from 56 cyber security analysts. The analysts were given four cyber security datasets to investigate, each containing a different security event that they had to discover and describe using various actions.
2. **Gold Dataset:** The authors from [1] use walkthrough documents created by cyber-security experts and developed the gold-standard EDA sessions. These documents guide viewers through the EDA process and point out key insights in the dataset. Table 4.3 shows how the distribution of trajectories from each dataset. The following sections explain how these trajectories are used for training and testing purposes.

### 4.1.2 Synthetic datasets

We also examine our strategy on synthetically created datasets because there aren't many expert EDA sessions for the cyber security datasets. We create datasets with a specific

	REACT	Gold
<b>Dataset 1</b>	104	7
<b>Dataset 2</b>	79	7
<b>Dataset 3</b>	76	7
<b>Dataset 4</b>	59	7

Table 4.3: Trajectory distribution for training across datasets

schema using an algorithm mentioned in following paragraphs, then induced interesting patterns in them. We then algorithmically produce EDA sessions on these synthetic datasets that discover these injected patterns. We use these sessions to train an AutoEDA model on the dataset and to compare the sessions generated by the learned model.

The proposed synthetic data generation algorithm consists of two steps:

**(1) Pattern Injection and Correlation:**

We start with a schema  $\mathcal{S} = \{C_1, C_2 \dots C_k, N_1, N_2 \dots N_l, T_1, T_2 \dots T_m\}$  where  $C_i, N_i, T_i$  denote categorical, numeric and text columns respectively. For each column  $c \in \mathcal{S}$ , we create a set of random *patterns*  $P(c) = \{p_1, p_2 \dots p_n\}$  and assign a random weight  $W(c) = \{w(p_1), w(p_2) \dots w(p_n)\}$  to each pattern ( $\sum w(p_i) = 1$ ). These patterns represent typical features of the data in column  $c$  and the weights indicate the probability that an element in a column follows a pattern.

- For categorical column, each  $p_i$  represents a category that occurs in the column.
- For a numerical column, each  $p_i$  is a pair  $(\mu_i, \sigma_i)$  where  $(\mu_i, \sigma_i)$  denotes mean and variance of Gaussian distribution respectively. Each element in the numeric column  $N_j$  will be sampled from one of the  $p_i \in P(N_j)$ .
- For a column with text, each  $p_i$  is a pair  $(s_i, POS_i)$  where  $s_i$  is a string and  $POS_i \in \{START, MIDDLE, END\}$ . If an element in a text column follows pattern  $(s_i, POS_i)$ , it means that the element contains the string  $s_i$  at its position specified by  $POS_i$ . The rest of it is padded with random strings.

We randomly create a set of *correlations*  $\mathcal{C} = \{C_1, C_2 \dots C_n\}$  where each  $C_i$  consists of two columns  $(c_{i1}, c_{i2})$  from  $\mathcal{S}$  and a set of pairs  $\{r_{i1}, r_{i2} \dots r_{in}\}$  where each pair  $r_{ik} = (p_{ik}^1, p_{ik}^2)$  represents a pattern from  $P(c_{i1})$  and a pattern from  $P(c_{i2})$ . The term correlation here does

not imply the mathematical term correlation. Rather, it indicates that the probability of observing a pattern in one column depends on the probability of observing patterns in the other column.

We can interpret each  $\mathcal{C}_i = (c_{i1}, c_{i2}, \{r_{i1}, r_{i2} \dots r_{in}\})$  as follows: column  $c_{i1}$  has a correlation with column  $c_{i2}$  such that for each  $r_{ik} = (p_{ik}^1, p_{ik}^2)$ , the probability of  $p_{ik}^2$  occurring in column  $c_{i2}$  increases whenever  $p_{ik}^1$  occurs in column  $c_{i1}$ . To ensure that the list of correlations does not form any cyclic chains (a set  $\{\mathcal{C}_1, \mathcal{C}_2 \dots \mathcal{C}_k\}$  where  $c_{12} = c_{21}, c_{22} = c_{31} \dots c_{n2} = c_{11}$ ), we limit the number of correlations that involve each column and make sure that the set of correlations forms a Directed Acyclic Graph (DAG) over the set of columns.

**(2) Row population:** In this step, we take the schema  $\mathcal{S}$  and generated list of correlations  $\mathcal{C}$  and generate the rows of the dataset. We generate each row one by one. To generate the element at column  $c$  in a row, we do the following:

- Set base distribution  $W = W(c)$  and get the relevant set of correlations  $\mathcal{C}_R = \{\mathcal{C}_i \in \mathcal{C} \text{ such that } c_{i2} = c\}$ .
- For each correlation  $\mathcal{C}_i \in \mathcal{C}_R$ , for each  $r_{ik} = (p_{ik}^1, p_{ik}^2)$ , multiply weight of  $p_{ik}^2$  in  $W$  with pre-defined multiplier  $m$ .
- Normalize  $W$  and sample pattern  $p$  according to updated weights. Generate element according to  $p$  as described above.

Our synthetic data generation algorithm is designed to generate a specified number of rows, each with a defined set of columns, including numerical, categorical, and string data. The algorithm takes in parameters such as a minimum and a maximum number of categories for categorical columns, which define the possible range of categories, and the range of the mean and standard deviation for numerical columns, which determine the normal distribution from which values are sampled. In addition, each string column is defined by a specific pattern that occurs at the start, middle, or end of the string.

The values in each column are correlated with values from other columns, with a specified maximum number of correlations. This correlation is represented as a set of tuples of correlated values and is mapped out as a directed acyclic graph (DAG) with nodes corresponding to correlated columns and edges representing correlated values from each column.



The correlation mapping determines the probability of the occurrence of one value based on the presence of another.

The algorithm starts at the root node of the DAG and traverses the subtree using a depth-first search approach. During this process, the type of column and its values determine the operations performed. For instance, if there is an edge between two categorical columns ( $c_i$  and  $c_j$ ) for corresponding values ( $v_i$  and  $v_j$ ), the possible order of operation would be either `[FILTER  $c_i$  EQ  $v_i$ , FILTER  $c_j$  EQ  $v_j$ ]` or `[FILTER  $c_i$  EQ  $v_i$ , GROUP  $c_i$  AGGREGATE COUNT  $c_j$ ]`. Numerical columns are filtered based on the mean of the distribution from which the values in each column are sampled, and string columns are filtered based on the pattern present in the column and its position. The algorithm returns to the root node of the subtree using `BACK` actions, and the trajectory ends when each node has been visited in topological order.

We create five artificial datasets for our experiments, each containing three categorical columns, two text columns and three numeric columns, and 1000 rows. We then produce expert trajectories as explained above. The resulting trajectories are randomly divided into train and evaluation sets, whose distribution is shown in Table 4.4.

	Train	Evaluation
<b>Dataset 1</b>	614	154
<b>Dataset 2</b>	3686	922
<b>Dataset 3</b>	614	154
<b>Dataset 4</b>	921	231
<b>Dataset 5</b>	1638	410
<b>Dataset 6</b>	819	205
<b>Dataset 7</b>	1228	308

Table 4.4: Trajectory distribution for synthetic dataset

## 4.2 Experimental setup

One of our objectives is to train a model that can generate insightful EDA sessions that can also handle datasets that it has not encountered before. Therefore, we design our training and evaluation procedure such that we only evaluate the model on datasets that are **unseen during training**. This way, we can assess the generalization capabilities of AdvEDA.

A “leave one out” strategy is used to train our model on the Cyber datasets, where we use

both REACT and gold trajectories from three of the four datasets for training and reserve the remaining one for testing. We evaluate the model by generating trajectories on the test dataset and comparing them with the gold-standard trajectories for that dataset. We repeat this process four times for each of the four cyber security datasets, and we compute scores to measure performance on each left-out dataset.

We use five out of seven synthetic datasets, which are artificially generated data that mimic real-world data, to train our model on all the trajectories in each dataset. For evaluation, we use the remaining two synthetic datasets and compare the trajectories generated by our model with the evaluation trajectories of these datasets.

### 4.3 Evaluation Metrics

To evaluate the quality of AdvEDA and the baseline, we compute benchmark scores that quantify how closely the generated sessions resemble expert EDA sessions. We adopt the following similarity metrics proposed in [1] for evaluation:

1. **Precision:** This metric counts the number of times a view occurs in the gold standard notebook. This is calculated by considering the generated EDA notebooks as distinct EDA actions. Thus, regardless of the sequence, counting the number of times a view occurs in the gold-standard notebooks. Every time a view is encountered, it is called a “hit”; else, consider it a “miss.”
2. **T-BLEU score:** This metric is adopted from BLEU [30] score, used for calculating the similarity of the machine translations to a set of reference translations. In this case, we consider a sequence of EDA views in the notebooks as a “sentence.” T-BLEU is stricter than Precision since it compares subsequences of size  $n$  (rather than single views) and considers both the order and prevalence of each view in the gold-standard set. Benchmark uses measures of T-BLEU-1, T-BLEU-2, and T-BLEU-3 ( $n$  between 1 to 3).
3. **EDA-Sim:** This metric was introduced in [16]. EDA-Sim considers the order of views and enables a fine-grained comparison of EDA views. Nearly identical views are regarded as “hit” as EDA-Sim will evaluate them as highly similar, whereas they

would have been considered “miss” in the abovementioned measures. The generated notebook is compared to each of the gold-standard notebooks to get the final EDA-Sim score, and the maximal score is used.

## 4.4 Implementation details

In the following section, we present the technical details of our implementation.

### 4.4.1 Data Preparation

We train our model using state-action representations derived from EDA sessions in the REACT data and gold-standard notebooks from various datasets (see Section 3.3 for details). The REACT data consists of EDA trajectories with different lengths, so we add a **STOP** action at the end of each trajectory. This allows our model to generate EDA notebooks of variable lengths by terminating the session when it outputs the **STOP** action. EDA session is a sequence of actions performed by our model until it outputs the **STOP** action.

### 4.4.2 Architecture Details

The policy network is a fully-connected neural network that has 3 hidden layers. Each hidden layer has 50 neurons and uses `tanh` as the activation function. The discriminator network is also a fully-connected neural network, but it has only 2 hidden layers. Each hidden layer has 32 neurons and uses `ReLU` as the activation function.

### 4.4.3 Model Initialization

We begin by learning a policy from expert demonstrations using Behaviour Cloning (BC). The BC model was trained for 100 epochs with a learning rate of  $10^{-4}$ . We then use the BC model to initialize the policy network’s weights, instead of using random weights. We further train the policy network for 10000 total environment interaction steps with a learning rate of

$10^{-6}$ . We also conduct ablation studies to examine the impact of this initialization method on the policy learning which is discussed in detail in Section 5.2.1

#### 4.4.4 Implementation of AdvEDA

Our system is built using PyTorch [31], an open source machine learning framework.. We extend the existing implementations of BC and GAIL in [32] to incorporate our proposed methods. We also leverage [33] for additional functionalities and utilities in our system.

	Cyber dataset 1		Cyber dataset 2		Cyber dataset 3		Cyber dataset 4	
<b>Metric</b>	<b>AdvEDA</b>	<b>ATENA</b>	<b>AdvEDA</b>	<b>ATENA</b>	<b>AdvEDA</b>	<b>ATENA</b>	<b>AdvEDA</b>	<b>ATENA</b>
Precision	<b>0.3750</b>	0.1855	<b>0.4000</b>	0.2340	<b>0.1429</b>	0.1153	<b>0.7500</b>	0.1929
T-BLEU-1	<b>0.3333</b>	0.1855	<b>0.2857</b>	0.2325	<b>0.1314</b>	0.1122	<b>0.3333</b>	0.1929
T-BLEU-2	<b>0.2041</b>	0.1377	<b>0.2182</b>	0.1873	0.0449	<b>0.0550</b>	<b>0.2041</b>	0.1451
T-BLEU-3	<b>0.0841</b>	0.0625	0.1060	<b>0.1182</b>	<b>0.0359</b>	0.0320	<b>0.0841</b>	0.0708
EDA-Sim	<b>0.2950</b>	0.2704	<b>0.3900</b>	0.2682	<b>0.3497</b>	0.2462	0.2051	<b>0.3017</b>

Table 4.5: A-EDA benchmark results on Cyber and Synthetic datasets

	Synthetic dataset 6		Synthetic dataset 7	
<b>Metric</b>	<b>AdvEDA</b>	<b>ATENA</b>	<b>AdvEDA</b>	<b>ATENA</b>
Precision	<b>0.4286</b>	0.1111	<b>0.8333</b>	0.1111
T-BLEU-1	<b>0.4286</b>	0.0515	<b>0.5333</b>	0.0429
T-BLEU-2	<b>0.1816</b>	0.0173	<b>0.4781</b>	0.0143
T-BLEU-3	<b>0.0669</b>	0.0132	<b>0.3852</b>	0.0114
EDA-Sim	<b>0.5647</b>	0.1539	<b>0.5536</b>	0.1265

Table 4.6: A-EDA benchmark results on Synthetic datasets

# Chapter 5

## Results and analysis

### 5.1 Baselines

We compare our approach with ATENA as a baseline to compare against. We use their publicly available code [34] to train and evaluate their model. However, note that this code does not implement the coherency reward that was described in their paper. Therefore, our comparison is based on a version of ATENA without the coherency score, which might affect its performance. We argue that this is a fair comparison because we also present a general model that does not rely on any dataset-specific rewards. The coherency reward might be very sensitive to the dataset and improve the results significantly. Section 5.3.6 discuss about more about coherency reward and it’s limitations. Moreover, to understand the impact of different components of our model, we also conduct an ablation study (see Section 5.2.1).

### 5.2 Results

Our model’s performance on various evaluation metrics and its comparison with ATENA are presented in Table 4.5. Unlike ATENA models, which are trained and evaluated on the same dataset, our models employ a ”leave one out” strategy as explained in Section 4.2. This means that they do not have access to any expert trajectories from the test dataset. The results show that our model outperforms ATENA in most scenarios. We conduct further

experiments to analyze these results in Section 5.3 and provide more insights.

### 5.2.1 Ablation studies

To examine the impact of some of our design choices, we run two ablation experiments:

1. **Model training without penalty scores:** The aim of this experiment is to examine how the quality of generated EDA sessions changes when we do not apply penalty reward to the model based on the design described in 3.4.2. We use the same configuration as AdvEDA except for penalty reward to train the model. Table 5.2 shows the results. We observe that the model without penalty performs worse than ATENA and AdvEDA. Table 5.1 lists some generated EDA sessions that can help us explain the lower scores. The EDA views 2 and 3 are the same because consecutive GROUP operators have been used. Our original model discourages such actions with penalty reward. Furthermore, we can see alternating GROUP and BACK actions. These sequences indicate uncertainty and are incoherent for users to follow.

---

```

FILTER eth_src EQ 00:26:b9:2b:0b:59
GROUP ip_src AGGREGATE AggregationFunction.COUNT length
GROUP ip_src AGGREGATE AggregationFunction.COUNT length
BACK
GROUP tcp_srcport AGGREGATE AggregationFunction.SUM ip_dst
BACK
GROUP info_line AGGREGATE AggregationFunction.SUM eth_src
GROUP tcp_srcport AGGREGATE AggregationFunction.SUM length
GROUP info_line AGGREGATE AggregationFunction.SUM length
BACK

```

---

Table 5.1: EDA session without penalty score on Cyber Dataset

2. **Training without BC initialization:** AdvEDA outperforms the version of model that is not pre-trained with Behavioral Cloning. This result indicates that pre-training the policy before applying the GAIL-based objective improves the model quality. We hypothesize that this phenomenon occurs because the pre-trained policy has a *warm* start that puts it in a better position to benefit from the adversarial training.

Table 5.2 shows the benchmarking scores for these experiments across four datasets. The average scores for each experiment are also presented. Compared to our complete model, these models perform significantly worse and often fail to surpass the baseline ATENA on several benchmarks. This indicates that the penalties and Behavioral cloning pre-training we introduced are crucial for our model’s performance.

	<b>Precision</b>	<b>T-BLEU-1</b>	<b>T-BLEU-2</b>	<b>T-BLEU-3</b>	<b>EDA-sim</b>
Without BC	0.259	0.103	0.038	0.029	0.195
Without penalty	0.413	0.239	0.144	0.058	0.258
ATENA	0.182	0.181	0.131	0.071	0.272
<b>AdvEDA (Ours)</b>	<b>0.417</b>	<b>0.271</b>	<b>0.168</b>	<b>0.078</b>	<b>0.310</b>

Table 5.2: Ablation study scores averaged our on Cyber dataset

## 5.3 Analysis and Discussion

In following sections, the experimental results are analysed and discussed in detail.

### 5.3.1 Insights captured by generated trajectories

We present two examples of EDA sessions produced by our model on dataset 1, as illustrated in Table 5.3 and 5.4. These examples demonstrate how our model can explore different aspects of the data.

---

```

FILTER eth_src EQ 00:26:b9:2b:0b:59
FILTER ip_src NEQ 82.108.69.238
FILTER highest_layer NEQ ARP
FILTER highest_layer NEQ ICMP
GROUP tcp_srcport AGGREGATE COUNT packet_number

```

---

Table 5.3: EDA session #1 generated by model on dataset 1

1. **Insights from Session 1:** As shown in Table 5.3, the model filters packets based on different values of highest\_layer column, which indicates the network layer protocol (such as Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP)) and Transmission Control Protocol (TCP) used by each packet . The model

first eliminates packets with ARP protocol, which is used to associate IP addresses with MAC addresses. Then it removes packets with ICMP protocol, which is used to send notifications of IP datagram problems. The resulting dataframe has a significant reduction in rows, implying that many packets use ICMP protocol. By examining the info\_line column, these packets are found to be ping requests, which are one of the key insights in this dataset. The dataset 1 aims to detect ICMP flood attack, which is successfully identified by the model through the generated EDA session. This demonstrates that our model can learn data insights from expert EDA demonstrations on similar domain data and can mimic expert behavior on unseen data. In the final action, the model groups the remaining packets by tcp\_srcport. These packets all use TCP protocol, which is used for reliable and ordered delivery of data. The grouping operator produces a group view that shows 304 and 135 packets in ports 139 and 445 respectively. These ports are SMB ports that are used for authentication and file sharing. The presence of these ports in TCP layer suggests that they are exposed to the internet and vulnerable to attacks.

---

```

FILTER eth_src EQ 00:26:b9:2b:0b:59
FILTER ip_src NEQ 82.108.69.238
FILTER highest_layer NEQ ICMP
FILTER highest_layer NEQ ARP
FILTER captured_length NEQ 62
GROUP captured_length AGGREGATE COUNT packet_number
BACK
GROUP highest_layer AGGREGATE COUNT length
GROUP ip_src AGGREGATE COUNT length
GROUP tcp_srcport AGGREGATE COUNT length

```

---

Table 5.4: EDA session #2 generated by model on dataset 1

2. **Insights from Session 2:** As in EDA session 1, the model filters out packets from different network layers as shown in Table 5.4. The model reveals that most of the packets belong to the ICMP layer after excluding them from the analysis. This shows that the model can generate trajectories that capture important insights from a given input dataset. Next, the model investigates the IP source of TCP packets. This behavior is similar to one of the gold-standard EDA notebooks where expert tried to find out the source and destination IP for TCP packets. This indicates that our model can discover the main highlight from the dataset as well as explore other potential



<b>A-INT</b>	<b>diversity</b>	<b>coherency</b>	<b>readability</b>	<b>peculiarity</b>
0.999	0.897	1.000	1.000	0.706
0.893	0.722	0.533	0.096	1.000
0.000	0.000	0.667	0.104	0.706
0.000	0.000	0.800	0.091	0.000
1.000	0.720	0.533	1.000	0.611
0.793	0.962	0.000	0.102	0.077
0.941	0.656	0.533	0.102	0.108
0.930	0.595	0.511	0.102	0.108
0.914	0.587	0.544	0.102	0.108
0.892	0.581	0.000	0.102	0.108
0.865	0.577	0.444	0.102	0.108
0.460	1.000	0.000	0.000	0.000

Table 5.5: Interestingness scores on ATENA session

insights. Moreover, this happens within a single EDA session unlike ATENA-generated EDA sessions where this phenomenon is not observed.

### 5.3.2 Interestingness captured in trajectories

We use interestingness measures described in section 1.1 to obtain scores on model generate EDA sessions. The scores are given in Table 5.6 and 5.7.

1. **Interestingness Measures on Session 1:** As shown in Table 5.6, A-INT, diversity, and peculiarity scores increased significantly when ICMP packets were excluded. This indicates that the model is able to detect important patterns in the data related to this view. Furthermore, the last view obtained the highest scores in most measures, which reflects that it is meaningful and interesting to viewers as it reveals another key aspect of the data.
2. **Interestingness Measures on Session 2:** Consistent with what we observed with interestingness scores in session 1, excluding ICMP packets resulted in high scores in most interestingness measures (see Table 5.7). In this EDA session, the views generated by the model had high readability scores as well as high diversity and peculiarity scores. This suggests that our model can effectively filter data and present relevant tuples to

users that are easy to read. The views presented are diverse, each showing different insights from the data.

Our model differs from ATENA in that it does not optimize for a specific set of interestingness scores when generating views. As shown in Table 5.5, ATENA’s views have high A-INT, diversity, and coherency scores, because these were the reward signals used to train it. However, our model’s views vary in their scores depending on the different interestingness measures they maximize. Our model aims to produce EDA views that highlight different aspects of the dataset rather than focusing on a particular set of interestingness measure.

<b>A-INT</b>	<b>diversity</b>	<b>coherency</b>	<b>readability</b>	<b>peculiarity</b>
0.890	0.721	0.924	0.006	0.659
0.000	0.000	1.000	0.000	0.659
0.000	0.037	0.000	0.000	0.659
0.416	0.312	0.000	0.005	1.000
1.000	1.000	0.978	1.000	0.000

Table 5.6: Interestingness scores for EDA session #1

<b>A-INT</b>	<b>diversity</b>	<b>coherency</b>	<b>readability</b>	<b>peculiarity</b>
1.000	0.893	0.832	1.000	0.000
0.048	0.616	0.900	0.948	0.000
0.492	0.729	0.000	0.993	0.967
0.049	0.675	0.000	0.948	0.970
0.050	0.680	0.000	0.949	1.000
0.000	1.000	0.000	0.000	1.000
0.000	0.000	0.800	0.845	1.000
0.000	0.808	0.000	0.000	1.000
0.141	0.916	1.000	0.845	1.000
0.068	0.774	0.880	0.916	1.000

Table 5.7: Interestingness scores for EDA session #2

### 5.3.3 Interestingness measure comparison analysis

We use normalization comparison from [17] to evaluate the interestingness of example sessions from experts, ATENA and AdvEDA. We compute interestingness scores for each action in the sessions and select the action  $a$  with the highest normalized score  $s_i$  as the most interesting

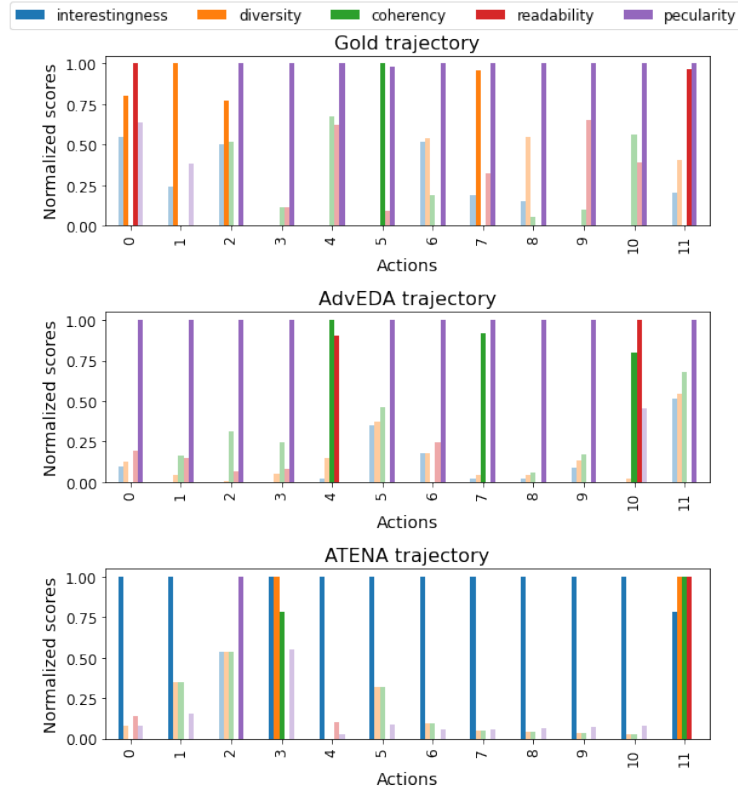


Figure 5.1: A comparison of Gold, ATENA and AdvEDA trajectories based on normalized interestingness metrics. The results show that AdvEDA can adapt to different metrics at different time steps, similar to the expert gold trajectory. ATENA, on the other hand, only focuses on the specific metrics that it uses as reward.

one. Figure 5.1 highlights the actions with score  $\geq 0.8$  and the interestingness measures they capture in the sessions.

The trajectories generated by AdvEDA show that the model prioritizes peculiarity, coherency, and readability more among all measures of interestingness. The trajectories obtained from experts also exhibit high scores for peculiarity, readability, and diversity, which are similar to those of AdvEDA. However, the trajectories produced by ATENA mainly focus on A-INT and coherency, which are the scores used as rewards for training ATENA.

This analysis show that human experts perform actions that optimize for various measures of interestingness, which are hard to define and implement explicitly. Therefore, proposed imitation learning approach learns to mimic the expert’s decisions by using their demonstrations. The imitation learning based model produces trajectories that match the expert’s interestingness measures better than handcrafted ones, as shown by our analysis.

### 5.3.4 Effect of STOP action

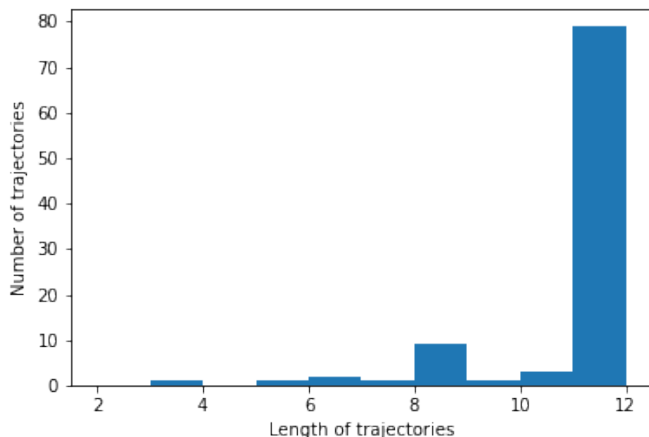


Figure 5.2: Distribution of lengths of trajectories generated by our model with STOP action

The distribution of exploration lengths for the trajectories generated by our model that has a STOP action is presented in Figure 5.2. Most of the trajectories have a length of around 12, which is not a clear reflection of the training data where most of the trajectories are very short. The longer lengths seem to be influenced by the penalties that we applied to the trajectories for making them coherent.

We also observe that some trajectories have lengths shorter than the maximum length, which suggests that some exploration paths require shorter lengths. A good example of this is the trajectory in Table 5.3 which shows good performance with only a length of 5.

### 5.3.5 Effect of Penalties

One way to evaluate the impact of penalties on our model is to compare the generated trajectories with and without penalties. We trained a version of our model without including any penalty terms in the reward function. A sample trajectory from this model is shown in Table 5.1. We observe that the model tends to repeat the same action multiple times in a row. This indicates that the model does not learn to explore different actions or avoid redundant actions. On the other hand, when we train our model with penalties, we do not see such repetitive patterns in the trajectories. This suggests that penalties help our model to learn more diverse and efficient sequences of actions. Therefore, we can infer that penalties are useful for preventing spurious sequences such as consecutive repetitions of the same action.

### 5.3.6 Coherency reward in ATENA

ATENA defines a coherency reward as the confidence score (in  $[0,1]$ ) of a weak-supervised classifier that judges if an EDA operation is coherent or not. The weak-supervised classifier is trained on hand-crafted heuristic rules. The authors used two kinds of rules:

- **General rules** : These rules consider the properties of the operation sequence.

For e.g., applying a filter on a dataframe with a few rows is regarded as incoherent.

```
"""Filter on a dataset containing
small amount of rows in the data
layer is NON-HUMANE"""
prev_fdf = get_previous_fdf(
state_history, past_steps=2)
if len(prev_fdf) < 40:
    too_low_rows_to_filter_punishment = -1.0
    return trigger_rule(
```

```

NetHumanRule.filter_small_number_of_rows,
too_low_rows_to_filter_punishment,
)

```

Or Performing a group operation on a column that is already grouped which adds no information for user

```

"""Group on a column that is
already grouped is NON-HUMANE"""
prev_state = state_history[-2]
if grouped_column in prev_state.grouping:
    column_already_grouped_punishment = -1.0
    return trigger_rule(
        NetHumanRule.column_already_grouped,
        column_already_grouped_punishment,
        remove_all=True,
    )

```

- **Data-dependent rules:** These are the optional rules that can be customized based on the schema and key attributes of the input dataset from user.

```

"""Filter on 'highest_layer', 'ip_dst',
'ip_src', 'info_line', 'tcp_dstport',
'tcp_srcport', 'tcp_stream' is HUMANE"""

most_humane_filter_columns = {"highest_layer",
"info_line"}
humane_filter_columns = {"ip_dst", "ip_src",
"tcp_dstport", "tcp_srcport"}

"""Filter on 'eth_dst', 'eth_src', 'length'
is MOSTLY-NON-HUMANE
(note that no human actually did that)"""

neutral_filter_columns = {"captured_length",
"length", "eth_dst", "eth_src", "tcp_stream"}

"""Filter on 'packet_number',
'sniff_timestamp' is NON-HUMANE"""
non_humane_filter_columns = {"number",
"packet_number", "sniff_timestamp"}

```

```

"""Group on 'eth_src', 'highest_layer',
'ip_dst', 'ip_src', 'tcp_dstport',
'tcp_srcport', 'tcp_stream' is HUMANE"""
most_humane_grouped_columns = {
    "highest_layer", "ip_dst", "ip_src"}

humane_grouped_columns2 = {
    "eth_src",
}
humane_grouped_columns = {"tcp_dstport",
    "tcp_srcport", "tcp_stream"}

"""Group on 'eth_dst', 'length'
is MOSTLY-NON-HUMANE"""

neutral_grouped_columns =
{"captured_length", "length", "eth_dst"}

"""Group on 'packet_number', 'info_line',
'sniff_timestamp' is NON-HUMANE"""
non_humane_grouped_columns =
{"number", "packet_number",
    "info_line", "sniff_timestamp"}

```

However, the authors of ATENA used a coherency reward to fine-tune their model on cyber datasets, which they reported in their paper . The coherency reward assigns scores to different columns based on how suitable they are for applying FILTER or GROUPBY actions. It also encodes some rules that specify which kinds of operations are more appropriate for each column.

In addition, each column has some rules that specifies the best operation or action for that column.

```

"""Filter using '<built-in function ne>' or '<built-in function eq>' on
the 'info_line' column is NON-HUMANE"""
filter_operator = last_action["operator"].lower()
if filtered_column == "info_line" and filter_operator in {"eq", "ne"}:
    info_line_bad_filter_operators = -1.0
    return trigger_rule(
        NetHumanRule.info_line_bad_filter_operators,

```

```

        info_line_bad_filter_operators ,
        remove_all=True ,
    )
elif filtered_column == "info_line" and filter_operator in {"contains"}:
    info_line_good_filter_operators = 0.5
    trigger_rule(
        NetHumanRule.info_line_good_filter_operators ,
        info_line_good_filter_operators ,
    )
elif filter_operator == "ne":
    """
    Using the filter operator NOT EQUAL should be discouraged
    """
    not_equal_filter_operator_punishment = -0.2
    trigger_rule(
        NetHumanRule.using_not_equal_filter_operator ,
        not_equal_filter_operator_punishment ,
    )

    """
    GROUP on eth_src that applied on a filtered display is INHUMANE
    """
if grouped_column == "eth_src":
    if state.filtering:
        eth_src_group_on_filtered_display_punishment = -0.5
        trigger_rule(
            NetHumanRule.group_eth_src_on_filtered_display ,
            eth_src_group_on_filtered_display_punishment ,
        )
    else:
        eth_src_group_on_non_filtered_display_incentive = 0.5
        trigger_rule(
            NetHumanRule.group_eth_src_on_filtered_display ,
            eth_src_group_on_non_filtered_display_incentive ,
        )

```

Using such kind of hard-coded rule defies the purpose of automatic EDA. ATENA used these manually designed coherency rules to guide its training and reward its outputs. When we train ATENA without these coherency rules, we find that its performance drops significantly as shown in Table 4.5. This suggests that the coherency rules are the main factor



behind ATENA's performance and not its ability to learn from data.



# Chapter 6

## Conclusion

We study the problem of automatically generating EDA sessions using a Reinforcement Learning framework. We first analyse the human generated trajectories with different interestingness scores to understand indirectly how a human would generate these trajectories and what should an Auto EDA system try to achieve. Based on our analysis, we suggest a novel algorithm for automatically generating EDA sessions and compare it against the baseline results. We see that our algorithm outperforms the baselines. We also show ablation studies to discuss our design choices. We finally analyse and discuss on our results to get insights on our model’s performance.

We introduced AdvEDA, a new end to end AutoEDA system that uses imitation learning to learn a policy for AutoEDA without relying on hand-crafted heuristics for interestingness. The proposed work is evaluated using automated benchmarking and manual analysis of generated EDA sessions shows that AdvEDA can perform meaningful EDA and outperform the baseline across benchmarks. Moreover, we demonstrate the improved generalizability of our system over the baseline. We also conduct ablation studies to justify our design choices. We finally discuss and analyze our results to gain insights into our model’s performance. Some possible directions for future work are: The following are some of the technical challenges and directions for future work:

1. Training a system that generalizes to perform AutoEDA across datasets with different schemas.

2. Developing a generalizable system that can perform AutoEDA across datasets with different schemas. This requires designing a robust and flexible framework that can automatically detect and adapt to the schema changes in the data sources over time.
3. Investigating the effect of other penalties on the performance of AutEDA. The current approach penalizes model for producing actions which are incoherent for users. However, other penalties, may also influence the quality and efficiency of generated trajectories. A systematic study on how different penalties affect AdvEDA is needed.
4. Incorporating actions other than FILTER and GROUP into AdvEDA. The current version of AdvEDA only supports 3 types of actions: BACK, FILTER and GROUP. However, there are many other possible actions that can be applied to data analysis, such as JOIN, AGGREGATE, SORT, PIVOT, etc. Extending AdvEDA to support more actions would increase its expressiveness and applicability to a wider range of data analysis tasks.

## 6.1 Paper communicated

This work is submitted to KDD 2023 conference [35]

# Bibliography

- [1] Ori Bar El, Tova Milo, and Amit Somech. Automatically generating data exploration sessions using deep reinforcement learning. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1527–1537, 2020.
- [2] Aurélien Personnaz, Sihem Amer-Yahia, Laure Berti-Equille, Maximilian Fabricius, and Srividya Subramanian. Balancing familiarity and curiosity in data exploration with deep reinforcement learning. In *Fourth Workshop in Exploiting AI Techniques for Data Management*, pages 16–23, 2021.
- [3] Lance Spitzner. The honeynet project: Trapping the hackers. *IEEE Security & Privacy*, 1(2):15–23, 2003.
- [4] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9–es, 2006.
- [5] Tova Milo and Amit Somech. Automating exploratory data analysis via machine learning: An overview. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2617–2622, 2020.
- [6] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. Seedb: Efficient data-driven visualization recommendations to support visual analytics. In *Proceedings of the VLDB Endowment International Conference on Very Large Data Bases*, volume 8, page 2182. NIH Public Access, 2015.
- [7] Marina Drosou and Evaggelia Pitoura. Ymaldb: exploring relational databases via result-driven recommendations. *The VLDB Journal*, 22(6):849–874, 2013.
- [8] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of olap data cubes. In *International Conference on Extending Database Technology*, pages 168–182. Springer, 1998.
- [9] Sunita Sarawagi. User-adaptive exploration of multidimensional data. In *VLDB*, volume 2000, pages 307–316. Citeseer, 2000.

- [10] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. Interactive data exploration with smart drill-down. *IEEE Transactions on Knowledge and Data Engineering*, 31(1):46–60, 2017.
- [11] Manish Singh, Michael J Cafarella, and HV Jagadish. Dbexplorer: Exploratory search in databases. In *EDBT*, pages 89–100, 2016.
- [12] Julien Aligon, Enrico Gallinucci, Matteo Golfarelli, Patrick Marcel, and Stefano Rizzi. A collaborative filtering approach for recommending olap sessions. *Decision Support Systems*, 69:20–30, 2015.
- [13] Magdalini Eirinaki, Suju Abraham, Neoklis Polyzotis, and Naushin Shaikh. Querie: Collaborative database exploration. *IEEE Transactions on knowledge and data engineering*, 26(7):1778–1790, 2013.
- [14] Xiaoyan Yang, Cecilia M Procopiuc, and Divesh Srivastava. Recommending join queries via query log analysis. In *2009 IEEE 25th International Conference on Data Engineering*, pages 964–975. IEEE, 2009.
- [15] Tova Milo and Amit Somech. React: Context-sensitive recommendations for data analysis. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2137–2140, 2016.
- [16] Tova Milo and Amit Somech. Next-step suggestions for modern interactive data analysis platforms. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 576–585, 2018.
- [17] Amit Somech, Tova Milo, and Chai Ozeri. Predicting” what is interesting” by mining interactive-data-analysis session logs. In *EDBT*, 2019.
- [18] Kyriaki Dimitriadou, Olga Papaemmanouil, and Yanlei Diao. Aide: an active learning-based approach for interactive data exploration. *IEEE Transactions on Knowledge and Data Engineering*, 28(11):2842–2856, 2016.
- [19] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.
- [20] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [21] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI’18*, page 4950–4957. AAAI Press, 2018.

- [22] Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [23] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00*, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.
- [24] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04*, page 1, New York, NY, USA, 2004. Association for Computing Machinery.
- [25] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3, AAAI'08*, page 1433–1438. AAAI Press, 2008.
- [26] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.
- [27] Varun Chandola and Vipin Kumar. Summarization–compressing data into an informative representation. *Knowledge and Information Systems*, 12(3):355–378, 2007.
- [28] Matthijs van Leeuwen. Maximal exceptions with minimal descriptions. *Data Mining and Knowledge Discovery*, 21(2):259–276, 2010.
- [29] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.
- [30] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.
- [31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [32] Steven Wang, Sam Toyer, Adam Gleave, and Scott Emmons. The imitation library for imitation learning and inverse reinforcement learning. <https://github.com/HumanCompatibleAI/imitation>, 2020.
- [33] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

- [34] Ori Bar El, Tova Milo, and Amit Somech. Atena basic implementation. <https://github.com/TAU-DB/ATENA-A-EDA/tree/master/atena-basic>, 2020.
- [35] Abhijit Manatkar, Devarsh Patel, Hima Patel, Naresh Manwani, and Shanmukha Gut-tula. Adveda: An advanced automatic end to end eda approach. Submitted to KDD 2023, 2023.