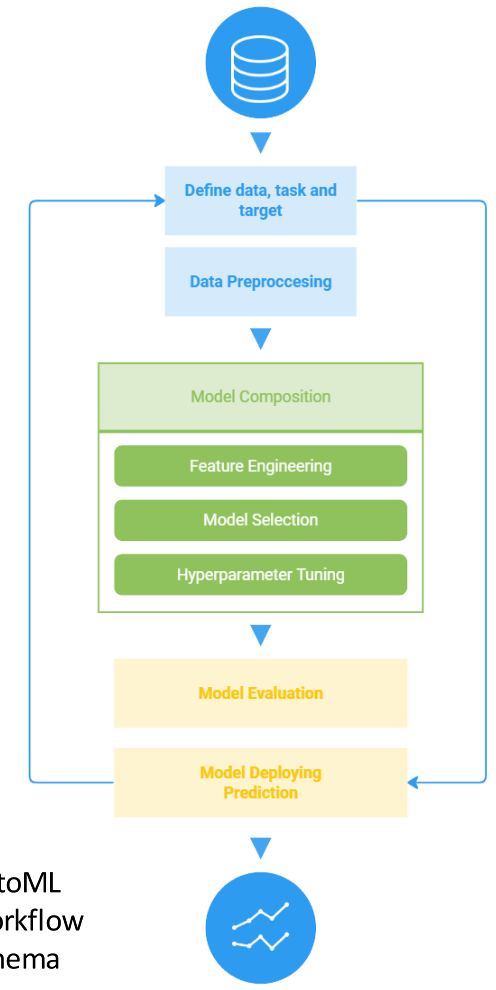
12th International Young Scientists Conference on Computational Science

Automated Generation of Ensemble Pipelines using Policy-Based Reinforcement Learning method

Andrey S. Stebenkov, Nikolay O. Nikitin
ITMO University

Introduction

Automated Machine Learning



AutoML Workflow schema

The central concept of **AutoML** is to build an automated workflow that could take raw data as input and produce a prediction automatically.

This automated workflow should automatically do **preprocessing**, **model selection**, **hyperparameter tuning**, and all other stages of the ML process.

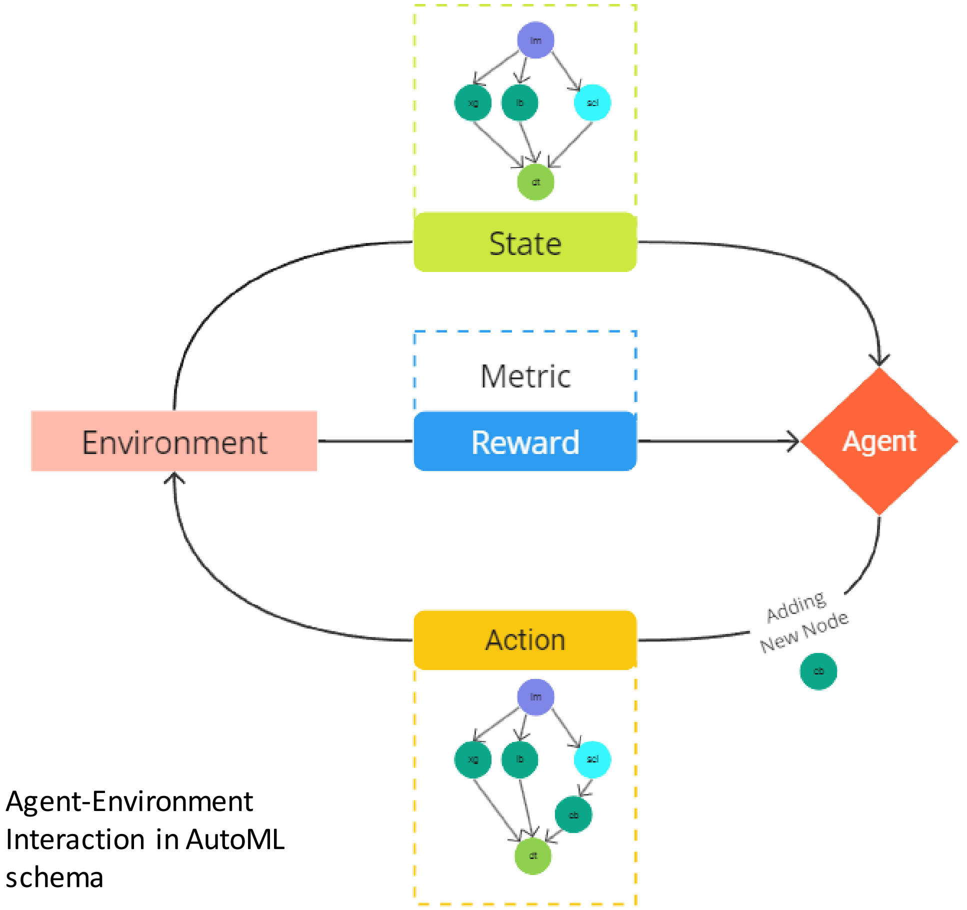
The final workflow is called a **Pipeline**.

There are a considerable number of different automated machine learning frameworks. The current widely used approaches are:

- Searching best from predefined models (e.g., AutoSklearn, AutoGluon);
- Constructing "step-by-step" with evolutionary algorithm (e.g., TPOT, FEDOT).

The **paper** introduces an approach to generate ensemble pipelines using **policy-based reinforcement learning**.

Problem statement



Agent-Environment Interaction in AutoML schema

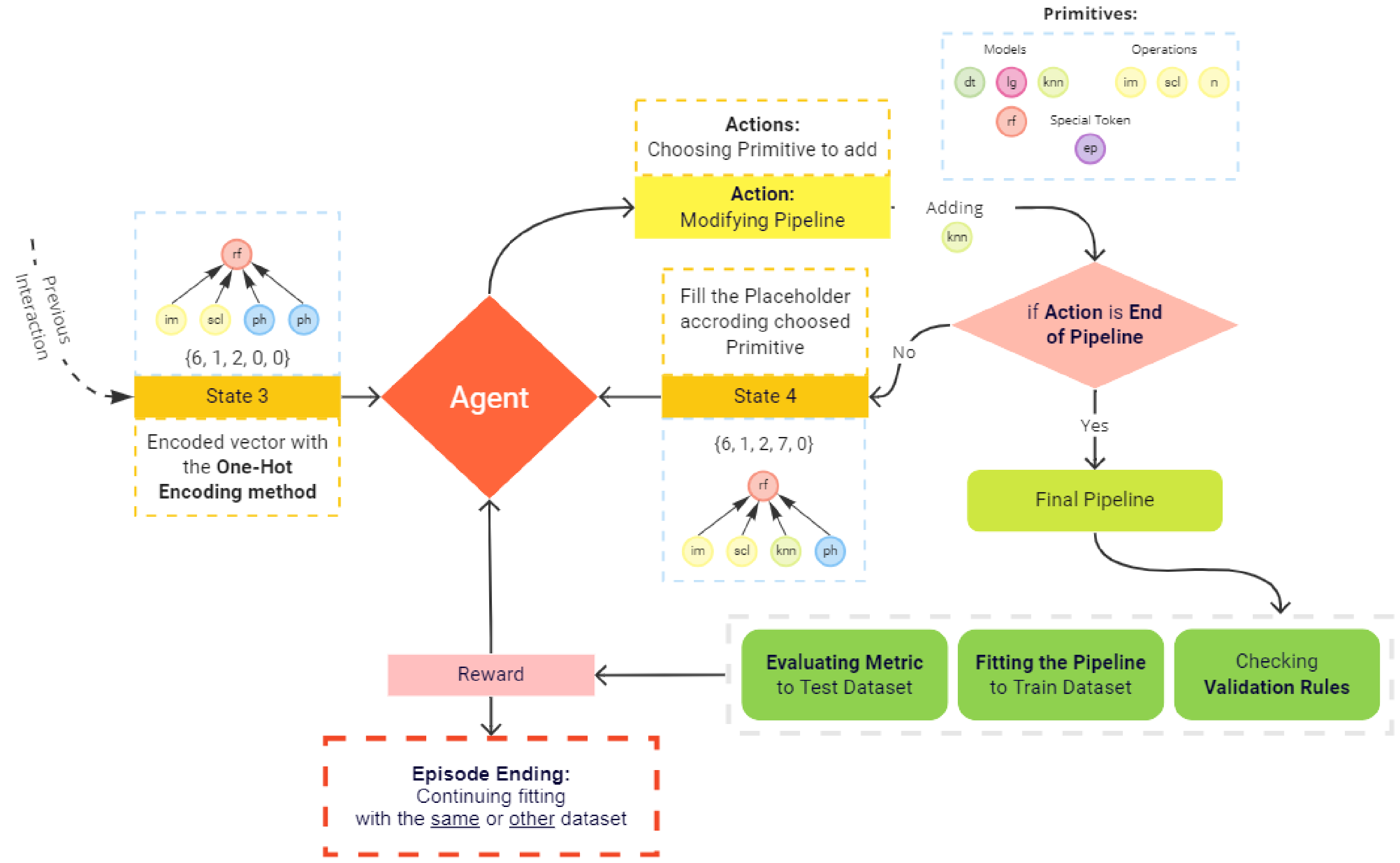
Implementating reinforcement learning to AutoML is not a novel idea. Similar to its predecessors, we suppose that the **pipeline generation problem** can be represented as **a computer game**.

The Agent learns how to **construct step-by-step** the optimal Pipeline by achieving maximum reward using **provided primitives** (models or operations).

State — Initial, Intermediate or Final pipeline structure;
Action — Modifying the pipeline (e.g., adding a new model);
Reward — Metric value of the final pipeline;
Episode — Generation one pipeline for the ML task;

Proposed approach

Full schema of fitting agent

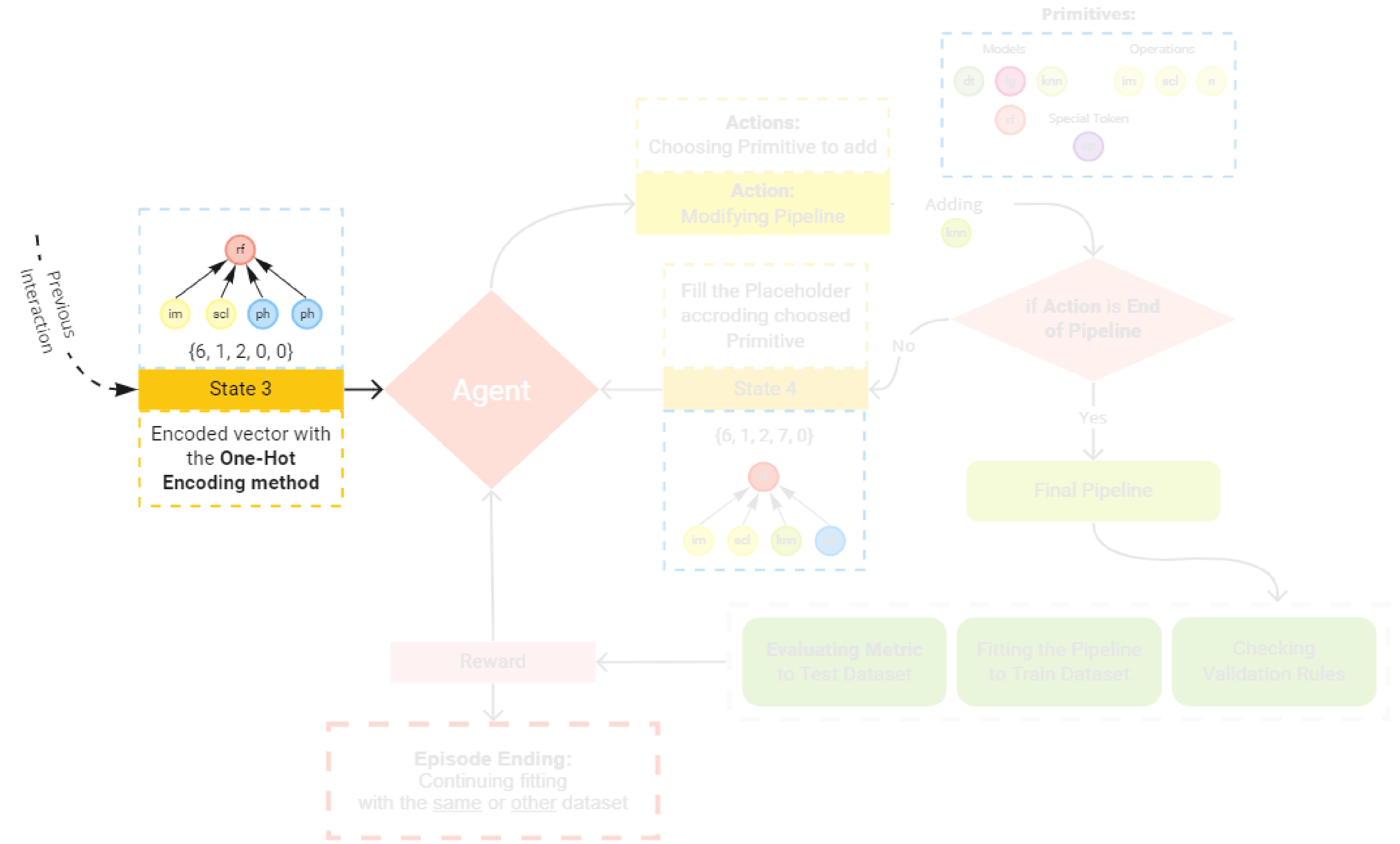


Primitives:
Models
dt
lg
knn
rf
Operations
im
scl
n
Special Token
ep
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
knn
if Action is End of Pipeline
No
Yes
Final Pipeline
Previous Interaction
rf
im
scl
ph
ph
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Agent
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
rf
im
scl
knn
ph
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

This schema shows the one stage from the entire cycle of Agent's fitting using a proposed approach.

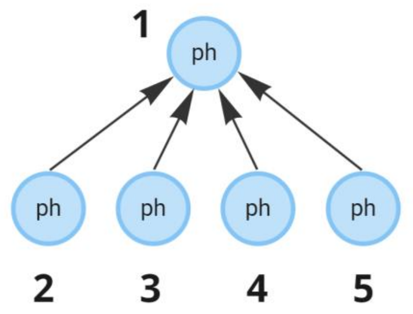
Proposed approach

Pipeline Encoding



Previous Interaction
rf
im
scl
ph
ph
{6, 1, 2, 0, 0}
State 3
Encoded vector with the **One-Hot Encoding method**
Agent
Actions: Choosing Primitive to add
Action: Modifying Pipeline
Primitives:
Models
Operations
dt
lo
knn
im
scl
n
rf
Special Token
Adding
knn
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
if Action is End of Pipeline
No
Yes
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending: Continuing fitting with the same or other dataset

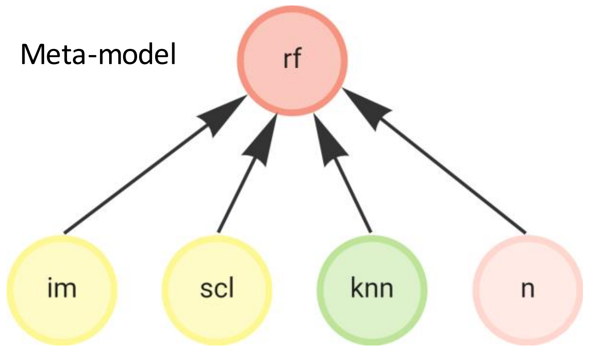
The stage will start with a pipeline graph into a one-hot encoding vector.



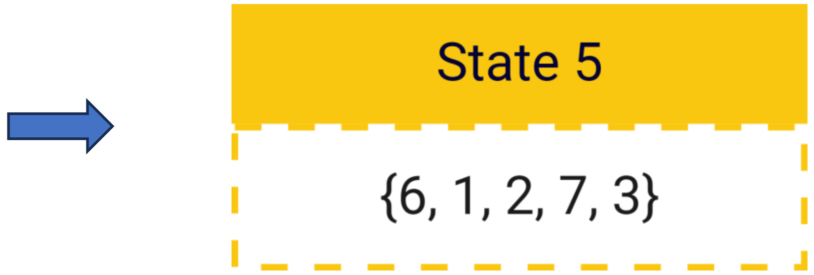
The initial structure of the Pipeline and the filling order.

Proposed approach

State Representation



Pipeline Graph



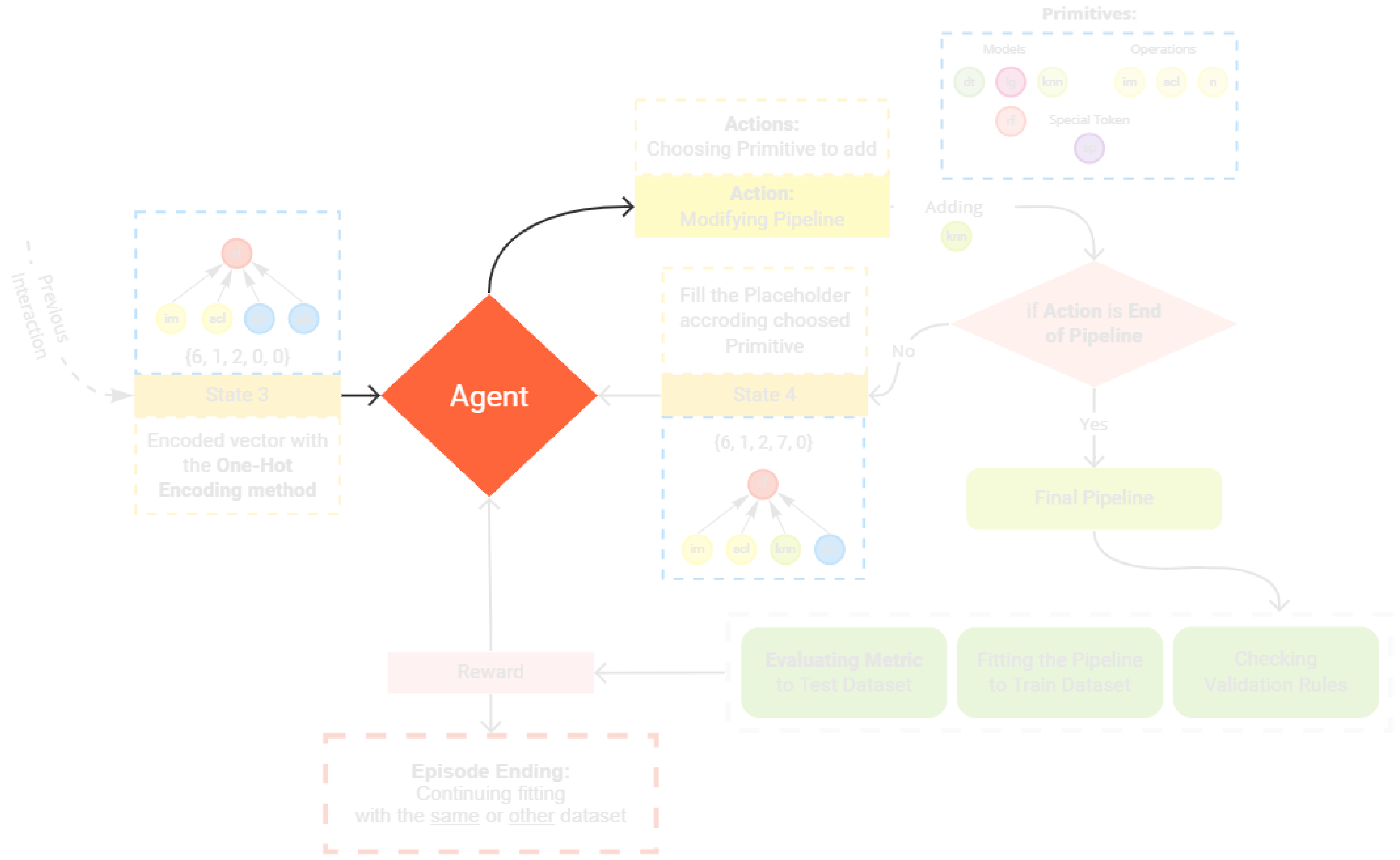
Pipeline vector representation

Ensemble structure use multiple primitives interconnected models in the main meta-model. Such pipelines also have constraints, **including primitives** and the **number of input nodes**.
The **state** can be formalized as **a one-dimensional vector** where primitives are **encoded labels**.

Initially, the state consists of ***Placeholders*** and after agent's actions it fills with primitives. The filling of such ensemble pipeline starts with choosing the meta-model. Finally, the state is converted into a **pipeline**.

Proposed approach

Agent


Primitives:
Models
Operations
Special Token
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
if Action is End of Pipeline
No
Yes
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Agent
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting with the same or other dataset

The main job is done by the Agent.

The Actor-Critic algorithm (A2C) was used as the agent.

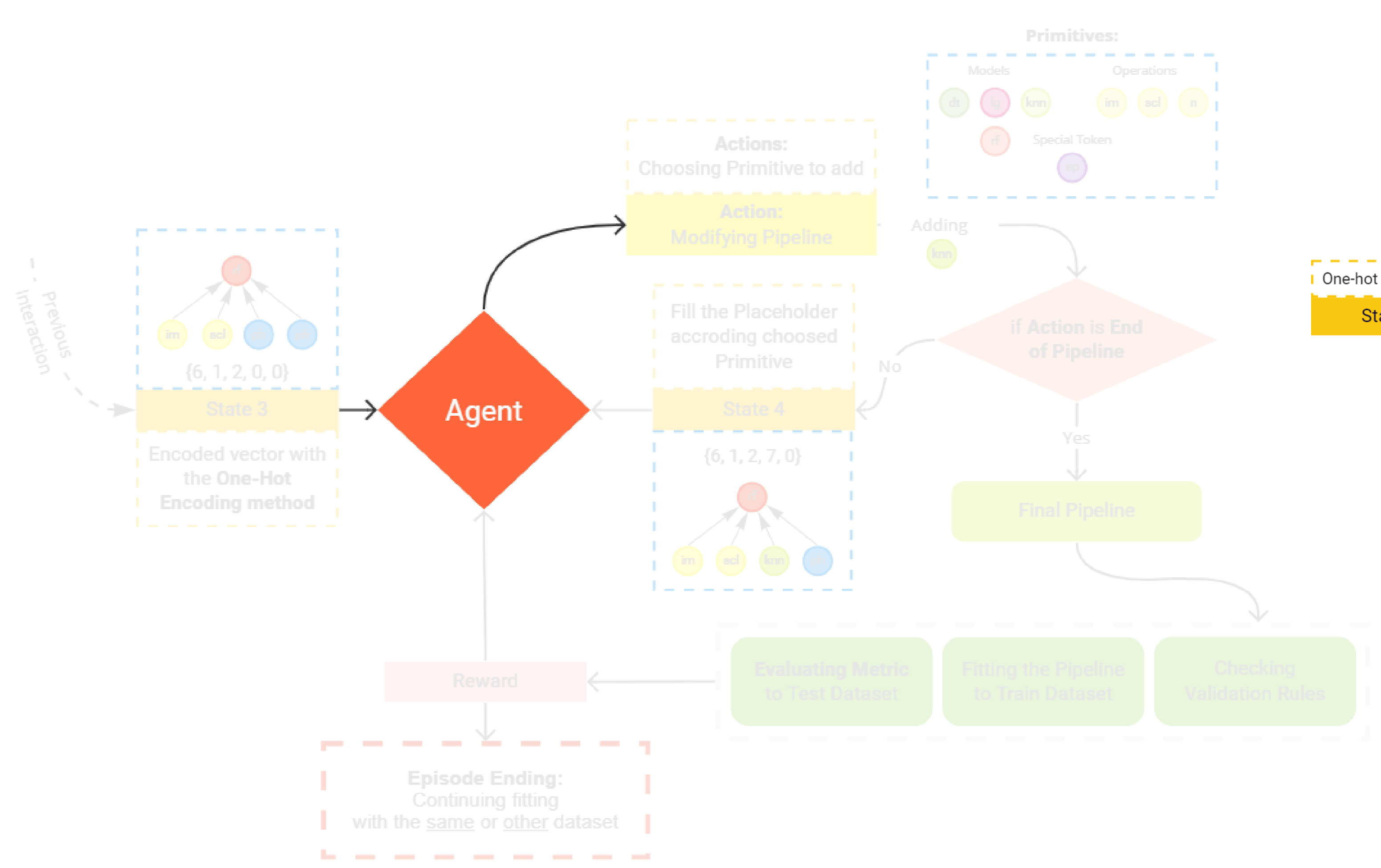
This method refers to the method based on policies.

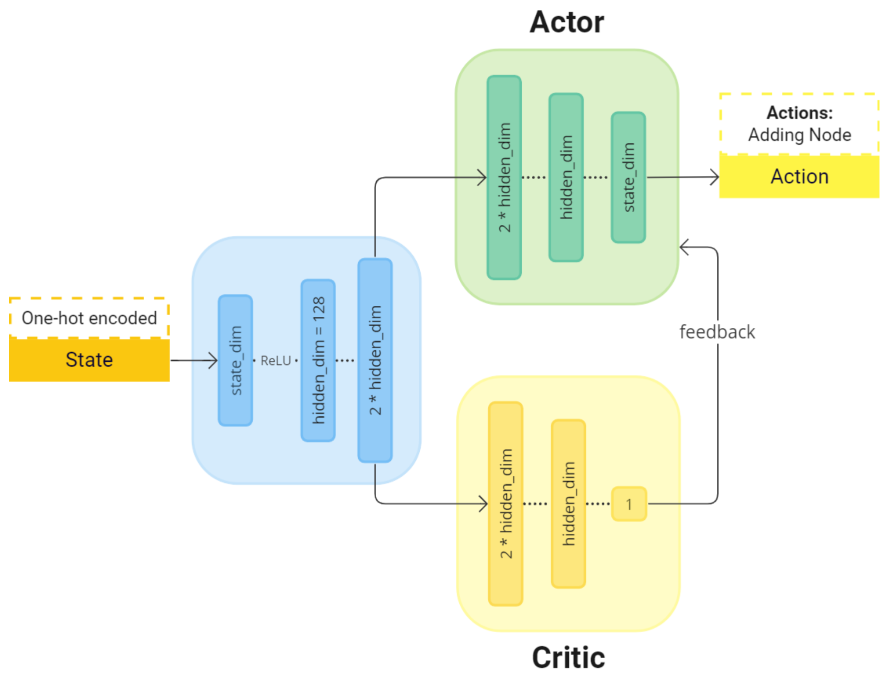
It purpose is to optimize the agent's behavior in various situations.

In the process, the Agent transforms the pipeline iteratively until it decides to stop.

Proposed approach

Agent Schema


Primitives:
Models
Operations
Special Token
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Agent
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
No
if Action is End of Pipeline
Yes
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

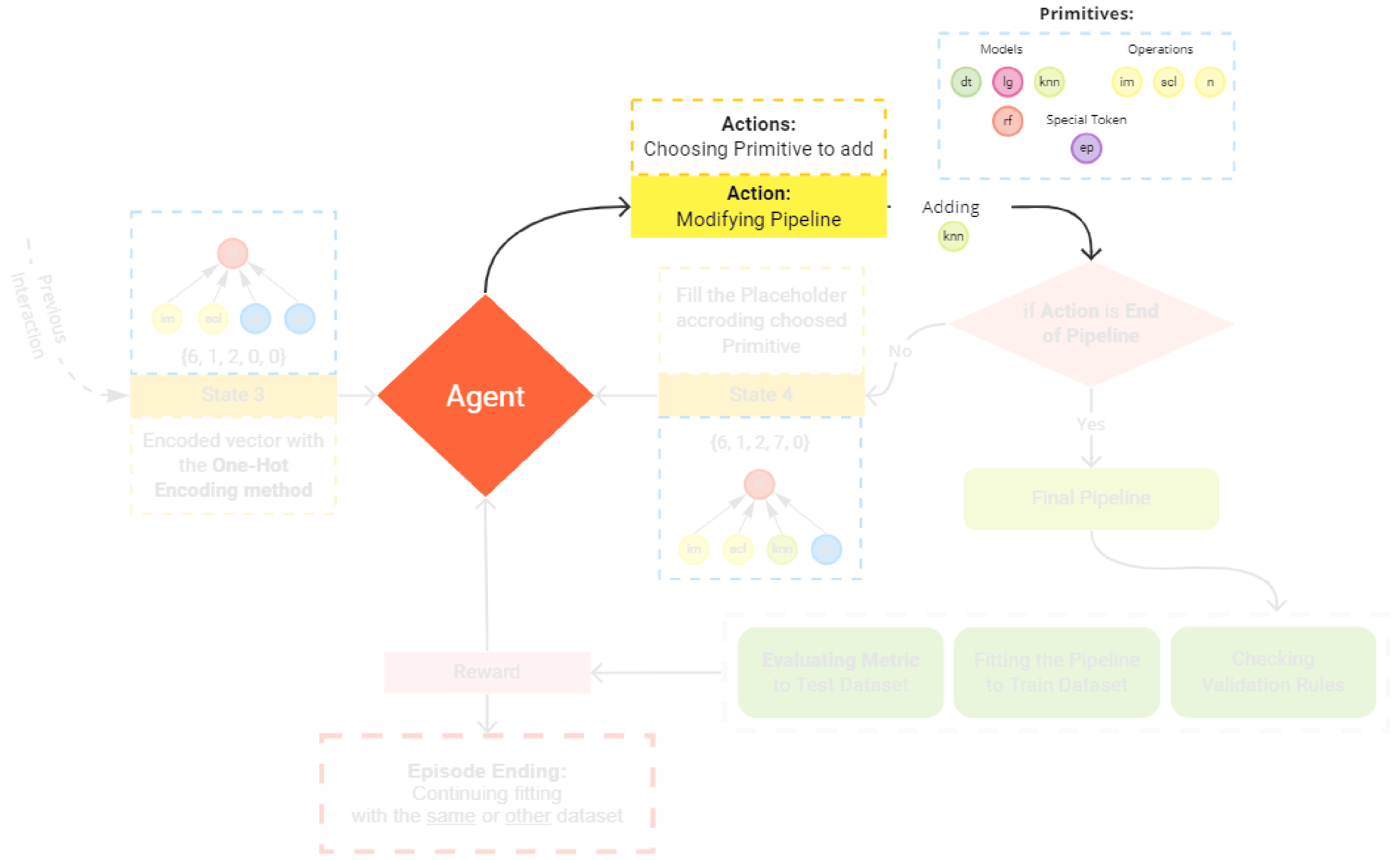


The architecture of network is consisting of 3 heads:

- Body;
- Actor;
- Critic.

Proposed approach

Pipeline Modifying



Primitives:
Models
dt
lg
knn
rf
Operations
im
scl
n
Special Token
ep
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
knn
Agent
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
if Action is End of Pipeline
No
Yes
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

The **Action** is changing ***Placeholder*** to primitives;

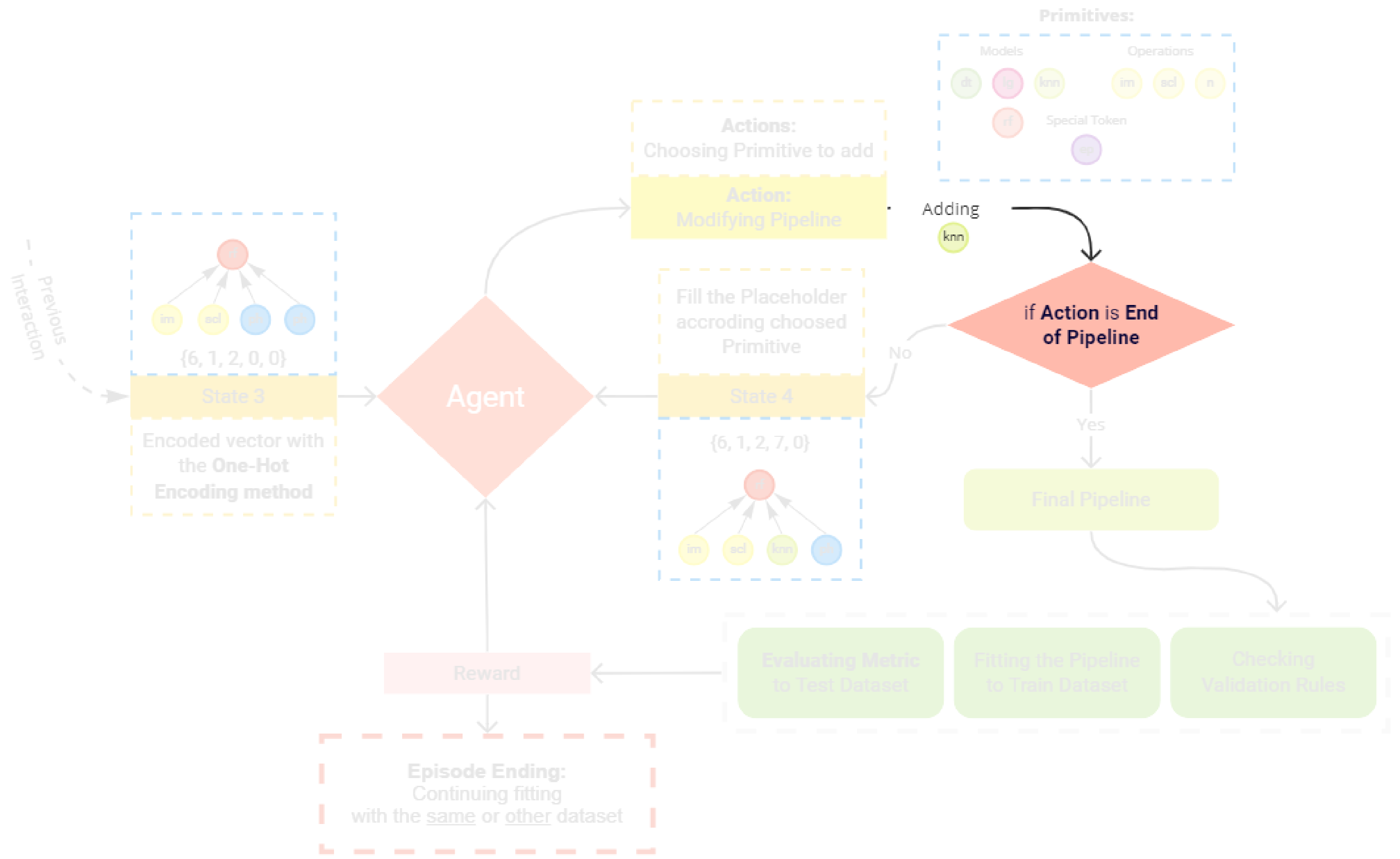
Each primitive is an encoded specific **operation** or **model**.

The primitives implementation was taken from AutoML FEDOT.

The agent can also stop the generative process early with ***End of Pipeline***.

Proposed approach

Checking is Action is the "End of Pipeline"?

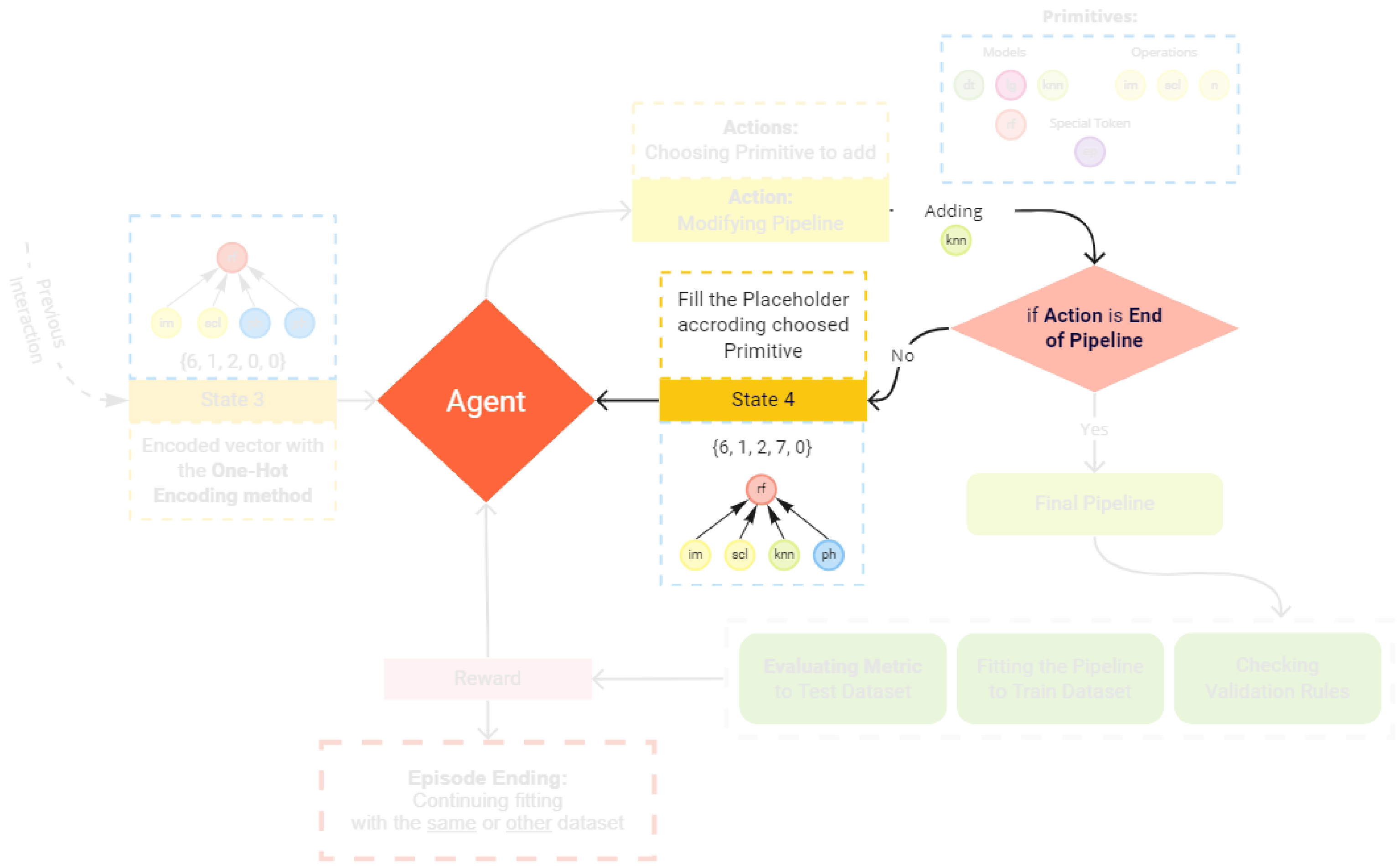


Primitives:
Models
Operations
dt
lg
knn
im
scl
n
rf
Special Token
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
knn
if Action is End of Pipeline
No
Yes
Final Pipeline
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
Agent
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

After the **Agent** chooses the action, the **Environment** checks for the ***End of Pipeline*** token.

Proposed approach

If Not -> Continue construction

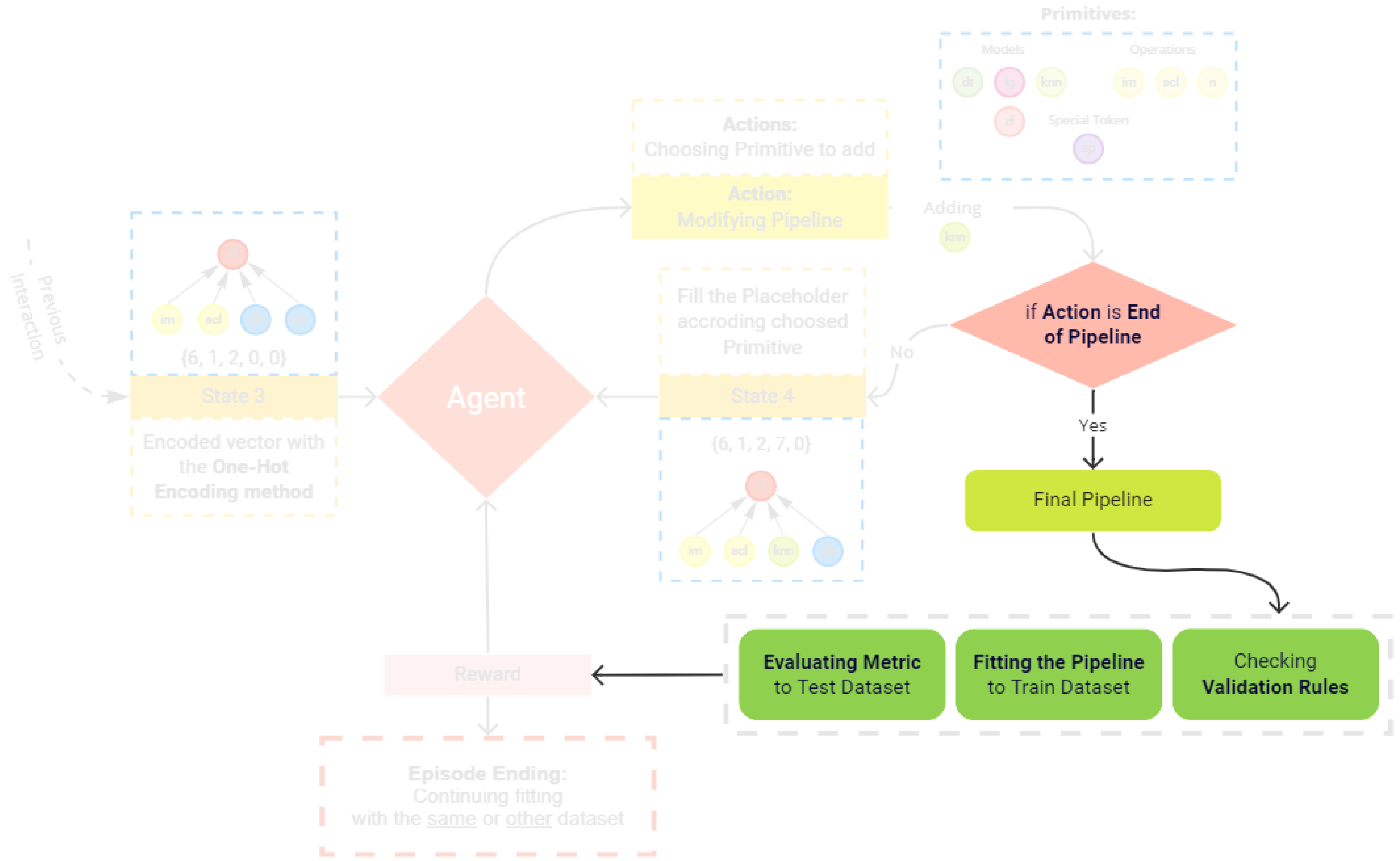

Primitives:
Models
Operations
dt
lg
knn
im
scl
n
rf
Special Token
ph
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
knn
if Action is End of Pipeline
No
Yes
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
rf
im
scl
knn
ph
Agent
Previous Interaction
State 3
{6, 1, 2, 0, 0}
Encoded vector with the One-Hot Encoding method
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

If the **Action** is not the ***End of Pipeline***:

- The Environment **modifies the previous state** by chosen Operations or Models.
- The next ***Placeholder*** in the Pipeline is filled by chosen **primitive**.
- The **modified Pipeline**, converted into **new state**, goes to the Agent, and the cycle will continue.

Proposed approach

If Yes -> Finish generation process



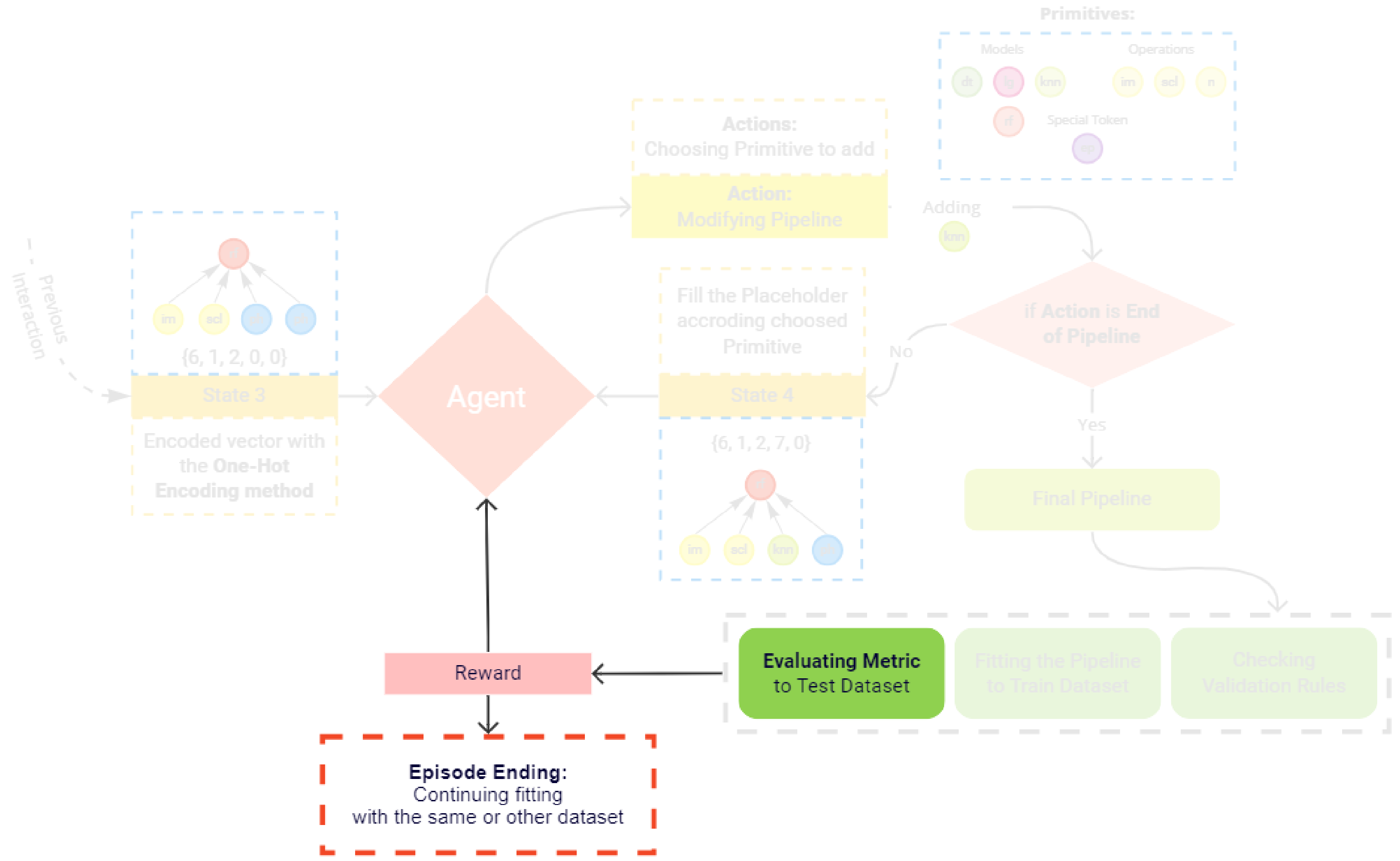
Primitives:
Models
Operations
Special Token
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Agent
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
No
if Action is End of Pipeline
Yes
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

If the **Action** is the ***End of Pipeline***:

- The Environment **build the final Pipeline**;
- Then, checks it at **Validation Rules**;
- **Fit** it to Train Data, if it is possible;
- Finally, **Evaluate Metric** to Test Data.

Proposed approach

Reward & Episode Finishing


Primitives:
Models
Operations
dt
lr
knn
im
scl
n
rf
Special Token
Actions:
Choosing Primitive to add
Action:
Modifying Pipeline
Adding
knn
Previous Interaction
{6, 1, 2, 0, 0}
State 3
Encoded vector with the One-Hot Encoding method
Agent
Fill the Placeholder accroding choosed Primitive
State 4
{6, 1, 2, 7, 0}
if Action is End of Pipeline
No
Yes
Final Pipeline
Evaluating Metric to Test Dataset
Fitting the Pipeline to Train Dataset
Checking Validation Rules
Reward
Episode Ending:
Continuing fitting
with the same or other dataset

The Agent **receives a reward for each interaction** with the Environment.

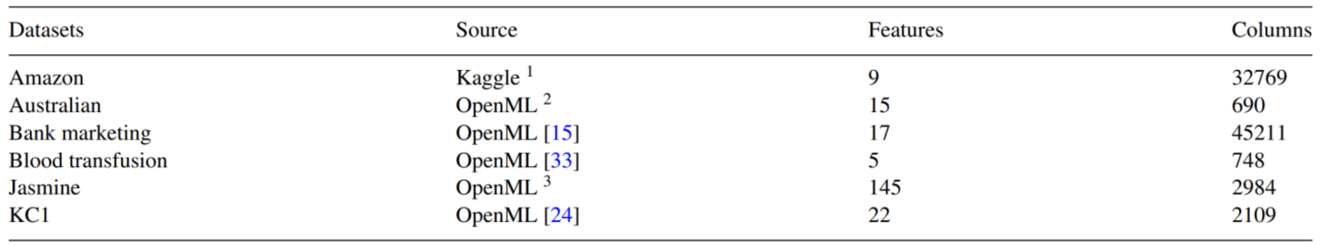
The **final metric score** is the most significant.

Time penalty and penalty to prevent of using a ***Placeholder*** or ***End of Pipeline*** also present in the Reward.

Experimental study

The proposed approach evaluated over six binary classification datasets with various specifics. Each dataset was separated into **train**, **test**, and **valid subsamples**. This was done for the **veracity of the experiment**:

- The Agent fitting at train;
- Getting reward at test;
- Validating with baselines models at valid.



Datasets	Source	Features	Columns
Amazon	Kaggle [1]	9	32769
Australian	OpenML [2]	15	690
Bank marketing	OpenML [15]	17	45211
Blood transfusion	OpenML [33]	5	748
Jasmine	OpenML [3]	145	2984
KC1	OpenML [24]	22	2109

The experiment was run at **Intel core i7** with parallel at **NVIDIA GeForce RTX 3060** for laptops.

The Agent was trained on **30000 episodes** on a train, and **every 1000 episodes** the Agent was verified on test data.

In the evaluation experiment, for each validation data it was required to construct **25 pipelines**. These pipelines were compared with the **baseline**.

Baseline pipelines were constructed using the **SkLearn Decision Tree model**.

Evaluations and Results

Datasets	Baseline Metric	Generated Pipeline Best Metric	Generated Pipeline Mean Metric	Number of valid Pipelines
Amazon	0.6807	**0.8622**	0.8071	25
Australian	0.8439	**0.9505**	0.9140	24
Bank marketing	0.7038	**0.9269**	0.8770	22
Blood transfusion	0.6543	**0.7308**	0.7197	24
Jasmine	0.4747	0.6088	**0.5297**	24
KC1	0.5575	0.8409	**0.8177**	23

Episodes	1000		3000		10000		15000		25000		30000	
Datasets	Valid	Reward	Valid	Reward	Valid	Reward	Valid	Reward	Valid	Reward	Valid	Reward
Amazon	10	-0.42	19	0.15	25	0.68	25	0.69	25	0.64	23	0.56
Australian	10	-0.26	18	0.27	25	0.82	25	0.79	25	0.76	25	0.80
Bank	8	-0.49	19	0.25	25	0.74	25	0.77	25	0.72	25	0.79
Blood	5	-0.76	20	0.19	25	0.60	25	0.61	25	0.59	25	0.56
Jasmine	7	-0.64	22	0.16	25	0.41	25	0.41	25	0.41	25	0.42
KC1	10	-0.42	19	0.17	25	0.66	25	0.67	24	0.57	21	0.37

Conclusion

Summary:

- Conducted a **research on the automatic generation of ensemble ML pipelines** using the Policy-Based RL method and presented a proof-of-concept. The proposed methods for the **environment**, **actions**, **reward**, and **states** were strong enough to **generate ensemble pipelines**.

- The successful generation proves the **applicability of RL methods to pipeline generation problems**.

- For future work, we plan to extend our approach to **generate non-fixed structure pipelines**.

Source code of the experiment research

YSC young scientists conference

Thanks for your attention