*All input variables of the classification model are included in the previous Data-Feature table. There's no new variables.
*Several features in the Data-Feature table is not necessary now, for details please check the latest Data-Feature table in the end of this file.

Input description of the classification model

Input variables		Index	Shape/type	Transfer to X_features	Numbers of labels	Feature importance
1st gain order sequence		0	[A list of 18 int]	float (list->a clustered label)	3 (can be 0,1, or 2)	0.041
1st gain time sequence		1	[A list of 18 int]	float (list->a clustered label)	3 (can be 0,1, or 2)	**0.049**
Accumulated amount of each item		2	[A list of 18 int]	float (list->a clustered label)	3 (can be 0,1, or 2)	0.018
~~Sparse reward accumulation sequence~~		3	[A list of int] ~~length= duration_steps~~	float (list->a clustered label)	3 (can be 0,1, or 2)	**0.050**
~~Dense reward accumulation sequence~~		4	[A list of int] ~~length= duration_steps~~	float (list->a clustered label)	3 (can be 0,1, or 2)	0.026
If useless tools was crafted (3 booleans)	iron_axe	5	boolean	float (1 if True else 0)		0
	stone_axe	6	boolean	float (1 if True else 0)		0.003
	wooden_axe	7	boolean	float (1 if True else 0)		0.015
Sparse total reward		8	int	float		**0.055**
Dense total reward		9	int	float		**0.120**
Attack efficiency		10	float	same		**0.097**
Attack ratio		11	float	same		**0.102**
Equipped attack ratio		12	float	same		**0.100**
Camera moving ratio		13	float	same		**0.111**
Position moving ratio		14	float	same		**0.100**
Placed_items (4 items)	torch_placed	15	int	float		0.038
	cobblestone_placed	16	int	float		0.028

批注 [z1]: Didn't use Sparse reward accumulation sequence, use varied form of 1st gain time sequence instead.

批注 [z2]: Didn't use Dense reward accumulation sequence, use varied form of 1st gain order sequence instead.

	dirt_placed	17	int	float		0.021
	stone_placed	18	int	float		0.019
If_smelt_coal		19	boolean	float (1 if True else 0)		0.008

*The order of 18 items in inventory list and how to calculate reward :

Order index (from 0)	**Item**	**Can get reward or not? ('x' for not)**	**How many reward?** **Sparse** reward: The reward will be accumulated whenever one item has been obtained at the first time; **Dense** reward: The reward will be accumulated every time when one item has been obtained, even if there are already same items.
0	'coal'	x	
1	'cobblestone'		16
2	'crafting_table'		4
3	'dirt'	x	
4	'furnace'		32
5	'iron_axe'	x	
6	'iron_ingot'		128
7	'iron_ore'		64
8	'iron_pickaxe'		256
9	'log'		1
10	'planks'		2
11	'stick'		4
12	'stone'	x	
13	'stone_axe'	x	
14	'stone_pickaxe'		32
15	'torch'	x	
16	'wooden_axe'	x	
17	'wooden_pickaxe'		8

Output description of the classification model:

Function name	Operation number	Description	Shape/type of return value	Other things
print_decision_tree	1	Print structure of decision tree	/	
check_input_tuple	2	Check if there are errors in input values and print	/	
predict	3	Prediction of the evaluated label (0 for under average ,1 for around average, 2 for over average)	A list of labels	Do the prediction of one input

Data-Feature table

Original Data	Features	Possible or not
'duration_steps' *All the other actions, inventory sequences, reward sequences and other features are based on the state of each "step".	Total time steps. 1 time step is 0.05 s (50 ms)	
'inventory_seq' -> Inventory information	**Inventory keeping sequence** ~~The amount of each item in the inventory in each step. This **Inventory keeping~~**	

	sequence is basic and very important, and has the highest priority. Type: a list of Ordered Dictionary [inventory1, inventory2, …] len(list)= duration_steps Here is an example of inventory: * **'inventory'** is an ordered dictionary looks like below (include 18 items, please keep this specific order, value of each key is the current amount of each item): ('coal', 0), ('cobblestone', 0), ('crafting_table', 0), ('dirt', 0), ('furnace', 0), ('iron_axe', 0), ('iron_ingot', 0), ('iron_ore', 0), ('iron_pickaxe', 0), ('log', 7), ('planks', 0), ('stick', 0), ('stone', 0), ('stone_axe', 0), ('stone_pickaxe', 0), ('torch', 0), ('wooden_axe', 0), ('wooden_pickaxe', 0)])),	

批注 [z3]: This is NOT necessary any more.
But we still need:
Gain **order** sequence of the 1st of each item
Gain **time_step** sequence of the 1st of each item
If useless tools was crafted
Accumulated amount of each item

	~~<- So, this inventory means at this time step, the player has 7 logs, other items are all 0.~~	
	Gain **order** sequence of the 1st of each item Type: list of int [1, 8, 0 , …] -> 18 items in same order as above item=0 if it wasn't gained in the episode; item=i(1<=i<=18) means it was the i_th gained item	
	Gain **time_step** sequence of the 1st of each item Type: list of int [100, 805, 0 , …] -> 18 items in same order as above item=0 if it wasn't gained in the episode; item=t(1<=i<= duration_steps) means it was gained at the t_th time_step	
	~~Gain **time_step** sequence of **all items**~~ ~~Type: 2-dimensional array of int~~ ~~[100, 805, 0 , …] -> 18 items in same order as above (columns)~~ ~~[200, 1000, 0,…]~~ ~~[……]~~ ~~->1562 rows (1562 is upper limit of the inventory. But the real data usually will not achieve the upper limit)~~ ~~item=0 if it wasn't gained in the episode;~~	

	~~item=t(1<=i<= duration_steps) means it was gained at the t_th time_step;~~ ~~the r_th row shows when the r_th same item was gained~~	
	If useless tools was crafted Type: a list of 3 booleans [boolean1, boolean2, boolean3] (Assume the following 3 tools are useless) boolean1: Whether the player crafted the **iron_axe**? True = yes, craft boolean2: Whether the player crafted the **stone_axe**? True = yes, craft boolean3: Whether the player crafted the **wooden_axe**? True = yes, craft	
	Accumulated amount of each item List of int: [10, 5, 8,…] -> 18 items in same order as above i.e. how many amount of each item the player got in total, include the items that was used. *the amount of each inventory item might decrease during the game, so the state of the last time step is NOT what we want.	
Sparse_reward information	**Sparse reward accumulation sequence** i.e. how many rewards the player get up till each step. The reward will be accumulated whenever one item has been obtained **at the**	

批注 [z4]: This won't be used

	first time (e.g. when getting the first log, reward+=1). If the rewards don't change in step t then reward(t)= reward(t-1)
	
	list of float (or int): [0.0,…, 1.0, 1.0, 2.0, 3.0,…]
	len(list)= duration_steps
	
	*Please check the following list for 'how much reward to give the player for each item'. Only items shown in this list will get a reward.
	
	`<Item reward="1" type="log" />`
	`<Item reward="2" type="planks" />`
	`<Item reward="4" type="stick" />`
	`<Item reward="4" type="crafting_table" />`
	`<Item reward="8" type="wooden_pickaxe" />`
	`<Item reward="16" type="cobblestone" />`
	`<Item reward="32" type="furnace" />`
	`<Item reward="32" type="stone_pickaxe" />`
	`<Item reward="64" type="iron_ore" />`
	`<Item reward="128" type="iron_ingot" />`
	`<Item reward="256" type="iron_pickaxe" />`	

	sparse_total_reward: The total reward = the last value in the above reward accumulated sequence. i.e. if duration_steps = ds, Total reward = reward_seq(ds) (if ds counts from 0, then it's ds-1)	

Dense_reward information	**Dense reward accumulation sequence**
	
	*Similar with **Sparse_reward** above EXCEPT that the reward will be accumulated **every time** when one item has been obtained, even if there are already same items (e.g. when getting the first log, reward+=1; when getting the second log, reward+=1…).
	
	list of float (or int): [0.0,…, 1.0, 1.0, 2.0, 3.0,…]
	len(list)= duration_steps
	
	*Please check the following list for 'how much reward to give the player for each item'. Only items shown in this list will get a reward.
	
	`<Item reward="1" type="log" />`
	`<Item reward="2" type="planks" />`
	`<Item reward="4" type="stick" />`
	`<Item reward="4" type="crafting_table" />`
	`<Item reward="8" type="wooden_pickaxe" />`
	`<Item reward="16" type="cobblestone" />`
	`<Item reward="32" type="furnace" />`
	`<Item reward="32" type="stone_pickaxe" />`
	`<Item reward="64" type="iron_ore" />`
	`<Item reward="128" type="iron_ingot" />`
	`<Item reward="256" type="iron_pickaxe"`	

	dense_total_reward: The total reward = the last value in the above reward accumulated sequence. i.e. if duration_steps = ds, Total reward = reward_seq(ds) (if ds counts from 0, then it's ds-1)	
'attack' information	attack_steps (int) If an attack action is performed: this is an attack_step. (i.e. how many steps the player did attack action)	
	Attack efficiency (float) = total_excavable_inventory / attack_steps *total_excavable_inventory = the total amount of **log**, **cobblestone** and **iron_ore** the player got. This value can be abstracted from '**Accumulated amount of each item**'.	
	Attack ratio (float) =attack_steps/ duration_steps	
	equipped_and_attack_steps (int) Number of steps that the player equipped **wooden_pickaxe** or **stone_pickaxe** and did attack actions at the same time	
	Equipped attack ratio (float) = equipped_and_attack_steps /attack_steps	

	*If a **wooden_pickaxe** or **stone_pickaxe** is equipped and an 'attack' action is performed, this is an equipped_and_attack_step.	
'camera' information	Camera position sequence Type: list of 2d array of float [array([0., 0.]), array([0., 0.1]), …] len(list)= duration_steps * **'camera'** is an array of 2 floats range between [-180.0, 180.0]	
	Camera moving ratio (float) =camera_moving_steps/ duration_steps if (camera_t) – (camera_t-1) ≠(0., 0.), it means the camera moved at t_th step	
7 move actions: **back, forward, left, right, jump, sneak, sprint**	move_steps (int) *It is a move_step, when the action is one (or more than one) of '**back, forward, left, right, jump, sneak, sprint'.** move_steps is the number of move_step.	
	Position moving ratio = move_steps/ duration_steps	
~~**Misuse action information**~~	~~**misuse_action_steps (int)**~~ ~~* if a player uses wooden_pickaxe to dig iron_ore, or uses stone_pickaxe to dig log,~~	

	~~this will be regarded as misuse of equipment.~~ ~~* misuse_action_steps is the number of steps that the player did equipment misuse~~	
	~~The ratio of equipment misuse.~~ ~~=misuse_action_steps/ duration_steps~~ ~~PS: In Action Space and Observation Space of MineRL:~~ ~~- **'equip'**(wooden_pickaxe, stone_pickaxe)~~ ~~- **'inventory'**(log, iron_ore)~~ ~~- **'attack'**=true~~	
Place item information	**placed_items** Type: a list of 4 integers [torch_placed, cobblestone_placed, dirt_placed, stone_placed] i.e. Total number of **torch, cobblestone**, **dirt**, and **stone** that were placed. PS: In Action Space of MineRL: **'place'** (torch, cobblestone, dirt, stone)	
'if_smelt_coal' information	if_smelt_coal=True (Boolean type), i.e. did the player once smelt coal? if the player once smelt coal, instead of dig the coal from mineral.	

批注 [z5]: This won't be used

批注 [z6]: This won't be used