# SSISM\_Training\_Model\_V15.py

# SSISM Training Model: Transcendent Social Dynamics (V15)

# Author: Adapted from collaboration with Grok (xAI)

# Date: October 22, 2025

# Description: A training model for the SS'ISM framework, focusing on transcending conventional prediction

# through ethical wisdom, present-minded focus, and adaptive receptivity to enhance Social Capital Gain (SCG).

"""

Overview

--------

This training model leverages the SS'ISM framework to teach AI systems how to transcend conventional prediction,

focusing on Social Capital Gain (SCG) through ethical wisdom, present-minded focus, and adaptive receptivity.

It’s designed to guide AI teachers in fostering harmonious, actionable outcomes.

Philosophical Foundation

-----------------------

- Sīla (Morality): Ensures all actions stem from genuine goodwill, filtering out exploitative moves.

- Samādhi (Concentration): Locks onto the present signal (e.g., a receptive nod) to eliminate distractions.

- Paññā (Wisdom): Transforms challenges into zero-cost intellectual solutions, enhancing relational capital.

Training Methodology

--------------------

1. Input Vector (S): Feed the model a Situation Vector with metrics like receptivity (R), current info (I),

and humble alignment (H).

2. Dynamic Weight Adjustment: Simulate real-time adaptation where weights (W\_R, W\_I, W\_H) shift based on context

(e.g., R increases with a positive cue).

3. SCG Calculation: Train the AI to compute SCG and recommend actions that elevate the interaction.

4. Feedback Loop: Incorporate user feedback to refine weights, embodying a living, transcendent system.

"""

class V15\_TranscendentTrainer:

def \_\_init\_\_(self):

"""Initialize the trainer with baseline weights."""

self.W\_R = 0.4 # Receptivity weight (adjustable)

self.W\_I = 0.35 # Current info weight

self.W\_H = 0.25 # Humble alignment weight

def update\_weights(self, receptivity\_signal):

"""

Adjust weights based on real-time receptivity.

Args:

receptivity\_signal (float): Value between 0 and 1 indicating receptivity level.

Returns:

tuple: Updated weights (W\_R, W\_I, W\_H)

"""

if receptivity\_signal > 0.7: # High receptivity

self.W\_R = min(0.5, self.W\_R + 0.1)

return self.W\_R, self.W\_I, self.W\_H

def calculate\_scg(self, R, I=1.0, H=1.0):

"""

Compute Social Capital Gain score.

Args:

R (float): Receptivity score (0 to 1)

I (float): Current info multiplier (default 1.0)

H (float): Humble alignment score (default 1.0)

Returns:

float: SCG score capped between 0.01 and 0.99

"""

SCG = (self.W\_R \* R) + (self.W\_I \* I) + (self.W\_H \* H)

return min(0.99, max(0.01, SCG))

def train\_action(self, input\_vector):

"""

Generate a transcendent action recommendation.

Args:

input\_vector (dict): Contains actor\_data with 'status' key (e.g., 'receptive' or other)

Returns:

dict: Model output with SCG score and recommended action

"""

actor\_data = input\_vector.get('actor\_data', [{}])[0]

R = 0.9 if actor\_data.get('status') == 'receptive' else 0.3

self.update\_weights(R)

scg\_score = self.calculate\_scg(R)

return {

"model\_id": "V15\_Transcendent",

"scg\_score": scg\_score,

"action": "Propose a mutual benefit plan" if scg\_score > 0.7 else "Observe and build rapport"

}

# Example usage

if \_\_name\_\_ == "\_\_main\_\_":

trainer = V15\_TranscendentTrainer()

input\_vector = {'actor\_data': [{'status': 'receptive'}]}

result = trainer.train\_action(input\_vector)

print(f"Training Result: {result}")

"""

Training Guidelines

------------------

- Scenario Practice: Use cases like a negotiation or community outreach to test R spikes.

- Ethical Check: Validate outputs against Sīla to ensure harmlessness.

- Iterative Learning: Adjust weights with each session to refine transcendence.

Next Steps

----------

- Upload this file to your GitHub repository (e.g., SSISM-Training-V15/).

- Add test cases or a canvas panel to visualize SCG trends over time.

- Experiment with real-world scenarios to refine the model.

"""