import csv

from datetime import datetime

# --- CONFIGURAZIONE INIZIALE ---

frequenza\_storica = 0.39

ritardo\_attuale = 16

ritardo\_medio = 11

ritardo\_massimo = 18

quota = 3.20

unit\_base = 5

storico\_esiti = []

# File storico ritardi con uscite successive (5,10,15)

FILE\_STORICO\_RITARDI = "storico\_ritardi.csv"

# Soglie

SOGLIA\_EV = 0.05

SOGLIA\_ALERT\_RECUPERO = 0.10 # 10% di incremento tasso recupero post-ritardo

# --- FUNZIONI DI SUPPORTO ---

def indice\_anomalia\_ritardo(ritardo\_attuale, ritardo\_medio, ritardo\_massimo):

delta = (ritardo\_attuale - ritardo\_medio) / (ritardo\_massimo - ritardo\_medio + 0.01)

return max(0.0, min(delta, 1.0))

def calcola\_probabilita\_migliorata(frequenza\_storica, ritardo\_attuale, ritardo\_medio, ritardo\_massimo, tasso\_recupero):

anomalia = indice\_anomalia\_ritardo(ritardo\_attuale, ritardo\_medio, ritardo\_massimo)

boost\_ritardo = 1 + (anomalia \* 0.10) # boost ritardo max 10%

boost\_recupero = 1 + tasso\_recupero # boost da tasso recupero

p = frequenza\_storica \* boost\_ritardo \* boost\_recupero

return min(p, 1.0)

def calcola\_ev(p, quota):

return p \* (quota - 1) - (1 - p)

def should\_play(ev, soglia\_ev=SOGLIA\_EV):

return ev >= soglia\_ev

def calcola\_puntata\_dalembert(esiti, unit\_base):

step = sum(1 if not esito else -1 for esito in esiti)

step = max(step, 0)

return unit\_base \* (1 + step)

def log\_scommessa(p, ev, quota, puntata, esito=None):

with open("log\_scommesse.csv", mode="a", newline="") as file:

writer = csv.writer(file)

writer.writerow([

datetime.now().strftime("%Y-%m-%d %H:%M:%S"),

f"{p:.3f}",

f"{ev:.3f}",

quota,

puntata,

esito if esito else ""

])

# --- GESTIONE STORICO RITARDI ---

def inizializza\_file\_storico():

try:

with open(FILE\_STORICO\_RITARDI, 'x', newline='') as f:

writer = csv.writer(f)

writer.writerow(["timestamp", "ritardo", "quota", "uscito\_in\_5", "uscito\_in\_10", "uscito\_in\_15"])

except FileExistsError:

pass # file già esistente

def salva\_ritardo(ritardo, quota):

# Registra un nuovo ritardo con uscite sconosciute (False)

with open(FILE\_STORICO\_RITARDI, 'a', newline='') as f:

writer = csv.writer(f)

writer.writerow([datetime.now().strftime("%Y-%m-%d %H:%M:%S"), ritardo, quota, False, False, False])

def leggi\_storico\_ritardi():

ritardi = []

with open(FILE\_STORICO\_RITARDI, 'r', newline='') as f:

reader = csv.DictReader(f)

for row in reader:

ritardi.append(row)

return ritardi

def calcola\_tasso\_recupero(storico):

# Calcola il tasso medio di uscite post-ritardo

tot = len(storico)

if tot == 0:

return 0.0

uscite\_5 = sum(row['uscito\_in\_5'] == 'True' for row in storico)

return uscite\_5 / tot

# --- MAIN ---

inizializza\_file\_storico()

# Simula aggiunta ritardo (da chiamare quando ritardo attuale supera il medio)

if ritardo\_attuale > ritardo\_medio:

salva\_ritardo(ritardo\_attuale, quota)

# Leggi storico aggiornato

storico = leggi\_storico\_ritardi()

# Calcola tasso recupero (es. su 5 partite successive)

tasso\_recupero = calcola\_tasso\_recupero(storico)

# Calcola probabilità migliorata

p = calcola\_probabilita\_migliorata(frequenza\_storica, ritardo\_attuale, ritardo\_medio, ritardo\_massimo, tasso\_recupero)

# Calcola EV e decisione

ev = calcola\_ev(p, quota)

giocare = should\_play(ev)

print("\n--- REPORT SISTEMA D'ALEMBERT + EV + RECUPERO ---")

print(f"Probabilità stimata corretta: {p:.2%}")

print(f"Valore Atteso (EV): {ev:.3f}")

print(f"Tasso recupero post-ritardo (5 partite): {tasso\_recupero:.2%}")

print(f"Giocare ora? {'SÌ' if giocare else 'NO'}")

# Mostra alert se tasso recupero elevato

if tasso\_recupero > SOGLIA\_ALERT\_RECUPERO:

print("\n\*\*\* ATTENZIONE: POSSIBILE INIZIO FREQUENZA! \*\*\*\n")

if giocare:

puntata = calcola\_puntata\_dalembert(storico\_esiti, unit\_base)

print(f"Puntata consigliata: €{puntata:.2f}")

log\_scommessa(p, ev, quota, puntata)

else:

print("Non giocare: EV insufficiente.")