Gym Wellness & Retention AI Agent: Combining Machine Learning, LLM Coaching, and Marketing Automatio

Project Report

Joyce Ghosn

Elie Haouch

Master's in Data Science – USJ

Contents

1.Introduction and Purpose

This project implements a Gym Wellness AI Agent that brings together traditional machine learning, large language models, and workflow automation to support gym coaches, marketing teams, and management in a practical and actionable way.

At its core, the system analyzes each member's recent wellbeing and attendance patterns, including mood, sleep, stress, fatigue, and activity levels, to estimate the likelihood that the following day will be a difficult one. A logistic regression model transforms seven-day behavioural summaries into a probability that "tomorrow will be a bad day." Based on this probability, the agent determines an appropriate risk level and selects a corresponding coaching strategy, such as recovery, balanced, or performance focus.

Once the risk level is defined, the system passes the member's profile and recent behavioural summary to an LLM powered by OpenAI. The model then generates a concise, empathetic explanation of the member's current situation, followed by a personalized three-day micro-plan that offers practical actions related to sleep, physical activity, and stress management. In addition, the LLM recommends one to three gym service from a fixed list, allowing the gym to track which offers are being suggested most frequently.

On top of this wellness-coaching layer, the project integrates an operational marketing automation workflow built in n8n. This workflow reads member data from Google Sheets and automatically sends personalized SMS/WhatsApp-style messages for different scenarios:

- New members (welcome & first-month retention),
- Old members with low attendance (gentle re-engagement),
- Old members with high attendance (congratulatory & promotional),
- Members whose birthday is today (birthday message with an offer).

These messages are generated with OpenAI, logged into Google Sheets, and delivered via Twilio, turning the AI outputs into real member-facing communication.

The business purpose of this combined agent is to help gyms:

- Improve member retention and engagement by proactively detecting individuals at risk of fatigue, stress, or disengagement.
- Provide coaches with ready-to-use, personalized plans instead of raw dashboards.
- Enable the marketing team to run data-driven, segmented campaigns based on risk level, attendance behaviour, and special events (such as birthdays).
- Track which services and offers are being recommended and promoted most often across different member segments.

Although the current implementation is based entirely on synthetic data, the dataset is designed to reflect realistic patterns of human behaviour so that the machine learning model learns meaningful relationships. The architecture can easily be transferred to real-world environments. A gym could replace the synthetic check-in data with real member data and run the full pipeline daily or weekly. The enriched outputs (risk, micro-plan, offer) can then

be written to a Google Sheet and consumed by the n8n workflow, which sends messages and logs activity automatically—making the system fully operational in a production setting.

2. Dataset & Data Generation Process

2.1 Member profile data

The first dataset, profiles.csv, contains one row per gym member and describes static characteristics and personality baselines:

- Demographics & identifiers
  - member_id: unique ID (1–30).
  - name: fake first name.
  - age: integer between 18 and 60.
  - sex: M / F.
- Goal & behavioural style
  - goal: high-level wellness goal (e.g. fat loss, muscle gain, general wellness), chosen from a fixed set.
  - activity_type: typical activity habit (low / moderate / high).
- Baseline wellbeing (personality-like traits)
  - baseline_mood (~3.2 on a 1–5 scale).
  - baseline_stress (~3.0 on a 1–5 scale).
  - baseline_sleep (~7 hours).
  - volatility: how much their wellbeing fluctuates; higher values mean more "chaotic" patterns.

These baselines are generated with controlled randomness (e.g. normal distributions around realistic means), so each member behaves differently but remains plausible. The LLM later uses these baselines to adapt the tone and recommendations.

2.2 Daily check-in data

The second dataset, checkins.csv, contains daily wellbeing data for each member over a time window (roughly 40–70 days per member). Each row represents one member-day with:

- member_id, date
- Wellbeing indicators (1–5 scale): mood, stress, fatigue
- Sleep & activity: sleep_hours, activity_minutes
- Label: bad_day_tomorrow – the target for the machine learning model.

The daily values are not independent. They are generated via rules that connect variables over time, for example:

- Poor sleep increases fatigue and stress, and can reduce mood.
- Higher activity tends to decrease stress and fatigue.
- High fatigue and stress push mood down.

These rules, plus volatility-scaled noise, create human-like time series where patterns such as "several nights of short sleep → higher fatigue → lower mood" are visible and consistent.

2.3 Label: bad_day_tomorrow

The label bad_day_tomorrow is derived from the simulated data through a rule-based definition of a bad day:

- low average sleep over recent days, or
- a combination of high stress and low mood, or
- high fatigue.

A binary bad_today is computed from these conditions, then shifted by one day:

bad_day_tomorrow at date t = bad_today at date $t+1$.

3. Feature Engineering & Data Preparation

3.1 From daily time series to 7-day windows

The machine learning model does not work on raw daily values alone. Instead, the code constructs a feature table where each row summarizes the previous seven days of behaviour for a given member-date.

Starting from checkins_df (one row per member-day), the pipeline:

1. Makes a working copy: feat_df = checkins_df.copy().
2. Groups by member_id so all rolling operations stay within each member: grouped = feat_df.groupby("member_id", group_keys=False).
3. For each member, computes 7-day rolling features (window=7, min_periods=7).

The main engineered features are:

- mood_avg_7d: average mood over the last 7 days.
- stress_avg_7d: average stress over the last 7 days.
- fatigue_avg_7d: average fatigue over the last 7 days.
- sleep_avg_7d: average sleep hours over the last 7 days.
- activity_days_7d: number of days (0–7) in the last week with activity_minutes > 0.
- sleep_debt_7d: defined as max(7 − sleep_avg_7d, 0) – the cumulated lack of sleep relative to a 7-hour target.

For the first six days of each member's timeline, these 7-day features are NaN (insufficient history), and such rows are excluded from model training.

3.2 ML-ready table

A separate ML table (e.g. ml_df) is built by:

- Selecting rows where all 7-day features exist (day ≥ 7 per member).
- Including the engineered features listed above as input variables.
- Keeping bad_day_tomorrow as the target label y.

Thus, one ML training row corresponds to:

"For member *i* at date *t*, given the pattern of the last 7 days, will day *t+1* be a bad day?"

3.3 Why these features are useful

The chosen features encode key behavioural and physiological drivers of readiness and risk:

- Average mood and fatigue capture emotional and physical load over the week, not just today's state.
- Average stress reflects ongoing pressure that may not be visible in a single day.
- Sleep averages and sleep debt represent chronic under-recovery.
- Activity days warn about sedentary behaviour or, in some cases, excessive training without recovery.

Together, they give the model a compact yet rich representation of member wellbeing over a full week, which is exactly what coaches care about when deciding how to adjust training and support.

---

4. Machine Learning Component

4.1 Model choice and training setup

The ML component is implemented using a logistic regression model trained on the engineered 7-day features to predict the binary label bad_day_tomorrow (0 = good/normal day, 1 = bad day).

- Input features (X):
  - mood_avg_7d
  - stress_avg_7d
  - fatigue_avg_7d
  - sleep_avg_7d
  - activity_days_7d
  - sleep_debt_7d
- Target (y):
  - bad_day_tomorrow

The data is split into training and test sets (classical train/test split), and the model is fitted on the training set. Performance is evaluated on both partitions to check for reasonable accuracy and potential overfitting.

Logistic regression is a good initial choice here because:

- It outputs probabilities (risk scores between 0 and 1).
- It is relatively easy to interpret (feature coefficients and direction of effect).
- It works well with a moderate number of features and binary outcomes.

4.2 Interpretation of model output: risk_score

For a given feature row x, the model is queried via:

```
risk_score = model.predict_proba(x)[0, 1]
```

This yields:

```
risk_score = P(bad_day_tomorrow = 1 | last 7 days of behaviour)
```

Example interpretations:

- risk_score = 0.15 → very low probability that tomorrow will be a bad day.
- risk_score = 0.52 → borderline; tomorrow could go either way.
- risk_score = 0.82 → high probability that tomorrow will be a bad day given how the last week looked.

This probabilistic score is the core numeric signal produced by the ML model.

4.3 From risk_score to risk_level and strategy

To make the model output actionable for coaches and LLM prompts, the project defines a simple business rule that maps the continuous risk_score to a discrete risk_level and a strategy:

- If risk_score < 0.4 →
  - risk_level = "LOW"
  - strategy = "performance"
- If 0.4 ≤ risk_score < 0.7 →
  - risk_level = "MEDIUM"
  - strategy = "balanced"
- If risk_score ≥ 0.7 →
  - risk_level = "HIGH"
  - strategy = "recovery"

This mapping is implemented in a helper function (e.g. choose_strategy(risk_score)) and is not learned by the model; it is a human-designed policy that aligns with intuitive coaching logic.

A wrapper function (e.g. compute_risk_and_strategy) applies the logistic regression and this business rule to each ML row, returning series with risk_score, risk_level, and strategy. The result is stored in a new DataFrame (e.g. ml_with_risk), which augments each (member, date) with the predicted risk and recommended high-level strategy.

4.4 Role of the ML output in the overall agent

In the final agent, the ML component answers one question per member and date:

"Given the last 7 days, how likely is it that tomorrow will be a bad day, and what overall strategy should we follow?"

This output then conditions everything that follows:

- It determines the tone and focus of the LLM explanation (e.g. gentle and recovery-oriented vs more motivational).
- It influences the type of exercises and recommendations in the 3-day plan.
- It guides which service offer is most appropriate (e.g. stress-coaching for high-risk vs performance-oriented sessions for low-risk).

---

5. From Numbers to Words: LLM Prompting & OpenAI Usage

5.1 Converting features into natural language summaries

Before calling the LLM, the pipeline builds a human-readable summary of the member's recent state from the numerical features and profile information. A function like summarize_features_for_llm(row, profile_row) reads:

- Profile: name, age, goal.
- 7-day features: mood_avg_7d, stress_avg_7d, fatigue_avg_7d, sleep_avg_7d, sleep_debt_7d, activity_days_7d.

It then applies simple threshold-based descriptions, for example:

- Mood: "generally good", "moderate", or "rather low" depending on the average.
- Stress & fatigue: "low", "moderate", or "high".
- Sleep: "quite short", "adequate", or "quite long" based on average hours.
- Activity: "very little", "some", or "regular" depending on active days.

This becomes context for the LLM. It is much easier for the model to reason in terms of "short sleep, high fatigue, low mood" than raw floating-point numbers.

5.2 LLM tasks: explanation + 3-day micro-plan + offer

Two main LLM functions are defined:

1. Risk Explanation (llm_explain_risk)
    - Input:
        - the numerical row with risk_level & strategy,
        - the natural language summary,
        - the member profile.

    - Prompt: instructs GPT to act as a compassionate gym wellness coach, read the summary, and:
        - briefly explain to the member why they might feel this way,
        - refer to stress, sleep, fatigue, activity patterns,
        - avoid mentioning probabilities, scores, or technical terms like "risk".
    - Output: a short paragraph or a few bullet points in simple, empathetic language.
2. 3-Day Micro-Plan (llm_generate_plan)
    - Input:
        - the same summary and profile,
        - the risk_level and strategy ("performance", "balanced", "recovery"),
        - the risk explanation text.
    - Prompt**:** instructs GPT to:
        - speak directly to the member by name,
        - open with 1–2 gentle sentences acknowledging their situation,
        - propose a 3-day plan with:
            - sleep targets,
            - activity level (e.g. light walk vs structured training),
            - relaxation / stress-management techniques,
        - keep the text concise (≈150 words), pragmatic, and encouraging.

The result is a personalized coaching message that transforms the abstract output of the ML model into a concrete, human-friendly plan.

5.2.1 Fixed list of service offers

In addition to the explanation and plan, the LLM is also instructed to choose one offer from a fixed list of gym services, for example:

- "yoga class"
- "mobility class"
- "pilates session"
- "personal training session"
- "nutrition consultation"
- "stress-coaching session"

This list is hard-coded and fixed in the prompt so that:

- The LLM cannot invent arbitrary services.
- The chosen offer can be stored in a dedicated column (e.g. offer_suggestion) in the output CSV.
- Power BI or other BI tools can later count how often each service is recommended, by risk level, by member profile, etc.

This design makes LLM outputs trackable and auditable, bridging the gap between free-form text and structured analytics.

6. Technical Integration: Google Colab + Google Sheets + n8n

6.1 Environment and OpenAI connection in Colab

The full ML + LLM pipeline runs inside a Google Colab notebook, which orchestrates:

- Data loading and synthetic data generation.
- Feature engineering and ML training.
- LLM prompting and output collection.

The OpenAI client is initialized as:

```
from openai import OpenAI
client = OpenAI()
```

An API key (stored securely in the Colab environment) authenticates the calls.

A central WellnessAgent class encapsulates the end-to-end logic:

- It is instantiated with:
  - the ML table with risk outputs (ml_with_risk),
  - the profiles_df with member profiles,
  - the trained ML model,
  - and the list of feature columns.
- For a given member_id and today_date, run_for_member(...) does:
  - Locate the row in ml_with_risk corresponding to that member and date.
  - Retrieve risk_score, risk_level, and strategy.
  - Fetch the profile row for that member.
  - Call llm_explain_risk to generate a personalized explanation.
  - Call llm_generate_plan to produce the 3-day plan text and the chosen offer.
  - Return a dictionary with all relevant fields.
- A helper run_for_all_members(today_date) loops over all members and produces a daily output DataFrame (e.g. daily_plans_df) with:
  - member_id, name, date_for_plan,
  - risk_score, risk_level, strategy,
  - risk_explanation, plan_text, offer_suggestion.

This class acts as the orchestrator that connects data, ML, and LLM into a coherent wellness agent.

6.2 n8n retention & messaging workflow

The second technical component is an n8n workflow created by your friend, which focuses on membership marketing and communication.

Key elements:

- Input:
  - A Google Sheet ("AI Fitness dataset") is read by a node called Members. Each row in Sheet1 corresponds to a gym member and contains fields like MemberID, Name, Age, Gender, Attendance_W4, Goal, Tone, Preferred_Workout, New_or_Old, Birthday.
- Branching logic:
  - If New_or_Old = "New" → member goes to a New Member path.
  - If New_or_Old = "Old" and Attendance_W4 < 2 → Low Attendance path.
  - If New_or_Old = "Old" and Attendance_W4 ≥ 2 → High Attendance path.
  - Independently, if today matches the day and month of Birthday → Birthday path.
- Message generation (LLM in n8n):
  - For new members, the workflow calls OpenAI with a prompt:

    > "Create a warm, welcoming message for a new gym member after their first month. Make it personal and motivating based on their fitness goal and preferred workout style. Use their preferred tone. Keep it brief (Trainup Gym retention)."

  - For low attendance, it asks for a supportive, non-pushy re-engagement message, linked to goal and preferred workout.
  - For high attendance, it creates congratulatory messages, possibly with rewards or promotions.
  - For birthdays, it generates short birthday messages with a promotional offer.

  These prompts interpolate variables such as Name, Age, Gender, Attendance, Goal, Tone, Category, and Preferred_Workout. OpenAI is accessed via the OpenAI Chat Model nodes, using models like gpt-4.1-mini and gpt-4o-mini.

- Logging:
  - All generated messages are appended to a logging sheet (Sheet3) via nodes like Messages Log, Messages Log2, Logs, and Logs1. Each row stores:
    - Timestamp, MemberID, Name, Category (New / Low / High / Birthday), and Message.
- Delivery (Twilio):
  - Twilio nodes (Send an SMS, Send an SMS1, Send an SMS2, Send an SMS3) send the generated message to a fixed phone number, simulating WhatsApp/SMS delivery to the member.
- Memory:
  - LangChain-style Simple Memory nodes store a short history per MemberID, allowing conversations to be contextual over time.

6.3 How both components connect

In the integrated architecture:

1. The Colab WellnessAgent generates daily_plans_df with risk level, explanation, 3-day plan, and offer_suggestion.
2. This DataFrame is exported to a Google Sheet (for example, an extra tab in the same "AI Fitness dataset" spreadsheet).
3. The n8n workflow is configured to watch this sheet (or join profile + plan sheets) and use the enriched information (risk level, offer suggestion) as additional context for the retention messages.
4. n8n then sends out channel-specific messages (SMS/WhatsApp) and logs them, closing the loop between analytics, coaching, and communication.

7. Member-Level Outputs & Practical Use

7.1 Structure of the enriched output

The final daily output (e.g. daily_plans_df or its exported Google Sheet) typically contains, for each member and plan date:

- Core identifiers: member_id, name, date_for_plan.
- Model outputs: risk_score (numeric), risk_level (LOW/MEDIUM/HIGH), strategy (performance/balanced/recovery).
- LLM outputs:
    - risk_explanation: short text explaining why the next day might be challenging or not.
    - plan_text: the 3-day micro-plan with sleep, activity, and relaxation recommendations.
    - offer_suggestion: one of the fixed services (yoga, mobility, pilates, PT, nutrition, stress-coaching).

The n8n workflow adds:

- Messaging context: category tags like New Member, Low Attendance, High Attendance, Birthday.
- Operational logs: the exact message text, timestamps, and delivery status stored in Sheet3 for future BI analysis.

8. Automation & Future Extensions

8.1 End-to-end automated workflow

The combined system naturally supports an end-to-end automated pipeline:

1. Data ingestion

    - The gym's membership or check-in system exports daily data (mood, stress, attendance, sleep proxies from wearables, etc.) into a database, warehouse, or Google Sheet.
    - A scheduled Python job or notebook prepares the raw input for the AI agent.
2. Scheduled ML + LLM job (Colab or server)
    - On a daily or weekly schedule, the script:
        - Loads the latest data.
        - Applies feature engineering and uses logistic regression to compute risk_score, risk_level, and strategy for each active member.
        - Calls the OpenAI API to generate risk_explanation, plan_text, and offer_suggestion.
    - The enriched output is written to a Google Sheet such as daily_plans_YYYY-MM-DD.
3. n8n automation layer (friend's workflow)
    - n8n watches the relevant sheets ("AI Fitness dataset").
    - For each member row, it:
        - Segments members into New, Low Attendance, High Attendance, or Birthday categories using IF nodes.
        - Sends personalized retention messages via OpenAI nodes and Twilio SMS.
        - Logs all messages (timestamp, category, text) in Sheet3.
    - Optionally, it can incorporate the risk_level and offer_suggestion columns from daily_plans_df to adapt message tone and highlight the recommended service.
4. BI & monitoring (Power BI / Looker / Data Studio)
    - Power BI imports the log sheets and enriched tables to produce dashboards covering:
        - Offer frequency and distribution by risk_level.
        - Evolution of risk across the member base over time.
        - Number and type of messages sent (new/low/high/birthday) and their association with future attendance and churn.

8.2 Next steps and enhancements

Natural extensions include:

- Real data integration: replace synthetic check-ins with real member data while keeping the same feature and modeling pipeline.
- Model refinement: explore tree-based models, add trend or variability features, or move towards survival analysis for churn prediction.
- Closed-loop feedback: log whether members attend the recommended class, follow the plan, or see improved risk later; use this outcome data to refine both the ML model and LLM prompts.
- Multi-channel messaging**:** extend n8n to send push notifications, app messages, or emails in addition to SMS/WhatsApp.
- Coach-in-the-loop tools: build a small web dashboard where coaches can review, edit, and approve the AI-generated plans and messages before sending.
- Deeper integration between risk and marketing: dynamically adjust promotional intensity and discount levels based on risk_level and recent engagement.

9. Conclusion

This project demonstrates how synthetic data, classical machine learning, LLMs, and workflow automation can be combined into a coherent, production-ready pipeline for the fitness industry.

On the data side, the system builds a realistic synthetic dataset with structured temporal relationships between sleep, mood, stress, fatigue, and activity, plus a meaningful label bad_day_tomorrow. On the modeling side, a logistic regression learns to map 7-day behavioural patterns to a probability of a bad day, which is then transformed into interpretable risk levels and strategies.

On top of this numerical core, the system uses an LLM (OpenAI GPT) to translate numbers into natural language explanations, personalized 3-day micro-plans, and structured service offers from a fixed list—making the outputs both human-friendly and analytically trackable.

The integration with your friend's n8n workflow shows how these insights can be operationalized: member data is pulled from Google Sheets, segmented into new/low/high attendance and birthday paths, enriched with AI-generated messages, logged to a central sheet, and pushed out via Twilio as SMS/WhatsApp-style messages.

Although the current implementation runs on simulated data, the design intentionally mirrors real-world flows. Swapping in actual check-in and wellbeing data, connecting to gym CRMs, and running the Colab + n8n chain on a schedule would be technically straightforward. This makes the project not just a school exercise, but a credible blueprint for how gyms could scale personalized, proactive coaching and retention campaigns to hundreds or thousands of members with limited human time.

Together, they form an intelligent layer that can help gyms protect member wellbeing, reduce churn, and create new opportunities for targeted, high-value services.