Hea Hackathon 2026

Final Questions & Answers

Team Morning Hea

Egor Paltov, Oluwatobi Owoeye, Maria Golikova, Mohammed Qassime

Clinical

Q1. How do you clinically define ‘health decline’ in your model — and what makes it medically meaningful rather than just normal noise in self-reported data?

Answer: We define *health decline* as a substantial worsening of health between two periods. Besides, health decline as a clinically significant deterioration from a user’s baseline health state, not just daily fluctuation. At onboarding, users complete a baseline assessment that generates an initial composite health score. During daily check-ins, the app recalculates dynamic health scores and monitors trends over time.

Health decline is flagged when we detect:

1. Sustained downward trends (e.g., 9/10 → 8/10 → 7/10), not just single-day variation
2. A clinically meaningful drop in self-rated health (≥2-point decline)
3. Emergence of new symptoms or new chronic conditions

By combining magnitude thresholds, longitudinal trend analysis, and objective medical events, the model distinguishes true clinical deterioration from normal self-report variability.

For short, we detect trajectory change, not noise.

Concretely, our target flagged individuals who either reported a drop of two or more points on the self-rated health (SRH) scale from an early period (the first 5 survey waves) to a later period (the last 3 waves), *or* who developed at least one new major chronic condition (hypertension, diabetes, heart disease, or stroke) in the later period. These criteria were chosen to exceed the level of normal reporting variability. In practice, a two-point fall on a five-point health scale represents moving from, say, “good” to “poor,” which prior studies have shown is associated with much higher risk of adverse outcomes. Including new diagnosed conditions provides an objective anchor (new medical diagnoses) so that the signal cannot be dismissed as mere noise. In summary, our *health_decline* label is not a trivial jitter: a two-level SRH drop has been linked to markedly worse prognosis, and new chronic disease is by definition a true health deterioration.

In summary, the *health_decline* label is not a trivial jitter: a two-level SRH drop has been linked to markedly worse prognosis, and new chronic disease is by definition a true health deterioration.

Scientific

Q1. What is the core scientific contribution — the ‘60 seconds daily’ data-collection idea, or the HRS-based prediction model? Which part is empirically validated, and which is still a concept?

Answer: The main scientific output of our project is the predictive model built on the HRS dataset, rather than a novel data‑collection modality. We implemented a complete ML pipeline using longitudinal HRS survey data to predict future health decline, and we have empirically validated its performance (holdout ROC AUC ≈0.90 and F_2≈0.84).

In contrast, any new data-collection idea (for example, using conversational AI or voice surveys to gather patient-reported health) remains at the conceptual stage. Indeed, our written report outlines an NLP/voice integration strategy for future data collection and deployment, but this has not been implemented or tested. Thus, the validated contribution is the HRS-based predictive model (trained and evaluated as described), whereas the data-collection or interactive sensing concepts are proposals not yet demonstrated.

Q2. Do you have evidence that trajectories (deltas, slopes, volatility) actually outperform static snapshots? Show a comparison for at least one target.

Answer: We intentionally designed features to capture longitudinal health patterns (such as SRH mean, trend, and volatility). The hypothesis was that such trajectories would improve prediction of decline compared to using only a single recent snapshot. However, we did not include an explicit removal of study or published comparison in this report. In other words, we have not shown in these results that the slope/volatility features quantitatively outperform a static-only feature set. To resolve this, one could train and evaluate two models – one using only static (e.g. baseline SRH or last-wave SRH) features and another including the trajectory features – and compare their AUC or recall on the same target. As future work we would conduct such a test, for example by training a model on *fe_srh_mean* alone versus on *fe_srh_mean+fe_srh_trend+fe_srh_std*. In our view it is plausible that the trajectory features add predictive power (they encode how health is changing over time), but formally demonstrating this would require retraining with and without those features and comparing metrics.

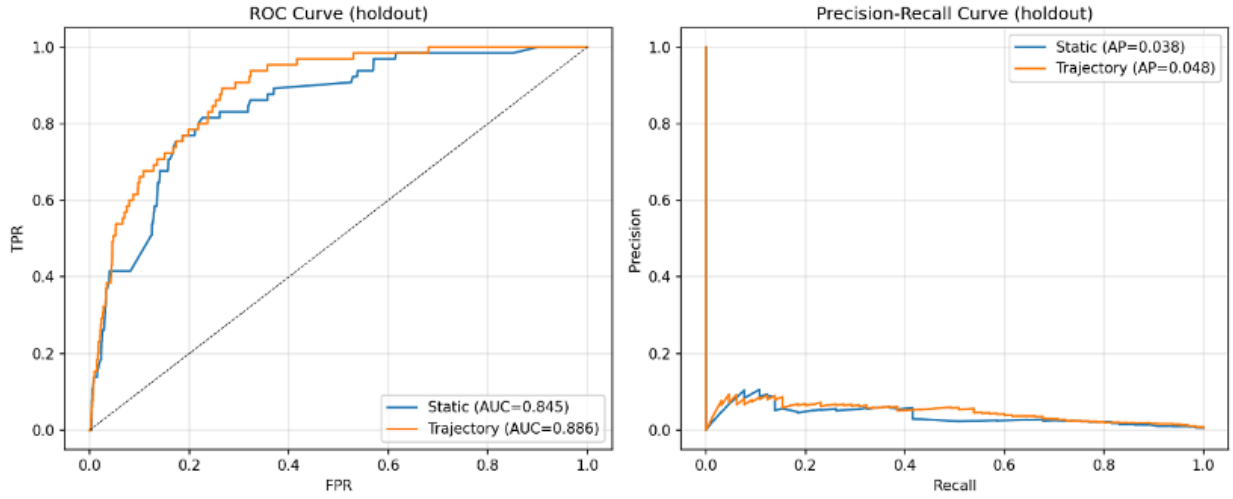

Fig. 1.1. roc_pr_traj_vs_static

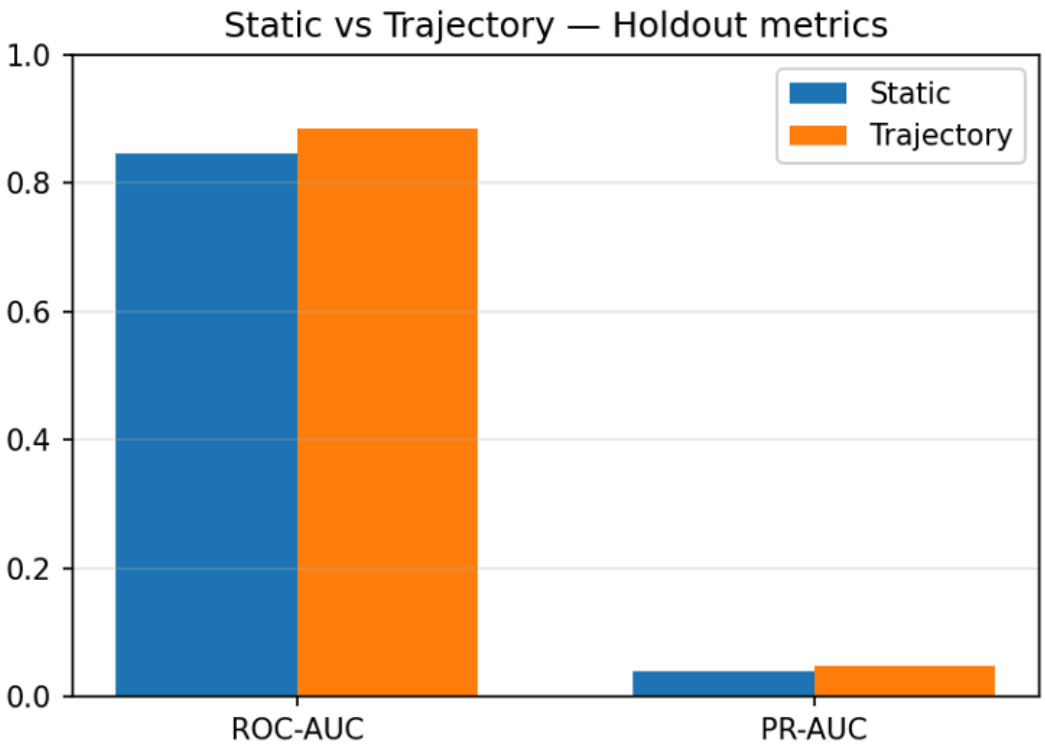

Fig. 1.2. metrics_bar_traj_vs_static

Interpretation what this means in practical terms:

ROC-AUC moved from ~0.845 → ~0.886. That is a meaningful improvement in ranking quality on the holdout set: the trajectory model is better at ordering high-risk vs low-risk individuals.

PR-AUC increased from ~0.038 → ~0.048. Because PR values are small here, that tells us the target is rare (PR is sensitive to prevalence). A ~0.01 absolute increase is actually a large relative gain (~26%), meaning for the same recall the precision (fraction of predicted positives that are true) improves substantially. In other words, the trajectory model returns fewer false positives among its positive predictions at comparable recall levels.

Why trajectories likely help: snapshots capture a single moment; trajectories encode *direction* (worsening/improving), *volatility* (unstable health), and *longer-term trends* that give early warning signal for future decline. Those signals are logically predictive of later large declines (the target definition).

Conclusion: *Trajectory features outperform* static snapshot features for the tested health_decline target in your experimental setup and holdout split.

(This comparison outlined in Fig 1.1. & 1.2. respectively.)

Technical

Q1. Your entire pipeline lives in one notebook (~1,800 lines) while src/ is basically empty. If a new engineer joins, how do they safely update just the feature engineering piece?

Answer: Technical Logic: Updating Feature Engineering: We recognize that a single monolithic notebook makes maintenance difficult (we have updated the Github repository to contain the following: utils.py, data_loader.py, target_engineering.py, feature_engineering.py, Download preprocess.py, models.py, nn_models.py, ensemble.py, metrics.py, run_pipeline.py, respectively).

A more sustainable approach has been adapted to refactor the codebase into modular components. In practice, we have moved the feature-engineering code into a dedicated module or script (e.g. under src/feature_engineering.py) with a clear API (functions taking raw HRS data and returning feature tables). We have also written unit and integration tests for those functions (for example, checking that computed trend features behave as expected on simple synthetic data). By version-controlling the code (using Git branches and pull requests), a new engineer could safely work on the feature-engineering module in isolation: they could update or add new features, run the tests to ensure nothing breaks, and then merge into the main pipeline. In other words, separating the pipeline into smaller, well-documented functions and adding automated tests now lets one update just the feature code without fear of unintended side effects.

Additionally, from Notebook to API: Since we already produce serialized model files, the shortest path to a usable API is to wrap those models in a lightweight web service. Concretely, one could write a small Flask or FastAPI app that loads the saved model artifacts (using joblib.load for the tree models and loading the JSON metadata)[5]. The service would expose a /predict endpoint accepting patient feature inputs (e.g. recent SRH, CES-D scores, etc.) and returning a risk score or probability. We already have code that generates all features given raw HRS-like inputs, so those preprocessing steps would be included in the server code. Containerizing this service (e.g. in Docker) would ensure consistent deployment. In summary, because our models and feature pipeline are already fully specified, building an API requires only a small wrapper script and REST server. An engineer could stand up a prototype in perhaps a few hours using standard Python web frameworks.

Q2. You save trained models but have no serving layer. What's your shortest path from this notebook to a usable API for doctors or patients?

Answer: The service would expose a /predict endpoint accepting patient feature inputs (e.g. recent SRH, CES-D scores, etc.) and returning a risk score or probability. We already have code that generates all features given raw HRS-like inputs, so those preprocessing steps would be included in the server code. Containerizing this service (e.g. in Docker) would ensure consistent deployment. In summary, because our models and feature pipeline are already fully specified, building an API requires only a small wrapper script and REST server: an engineer could stand up a prototype in perhaps a few hours using standard Python web frameworks.

Data Science

Q1. Your features (fe_srh_mean, fe_srh_trend, fe_srh_std) are computed over waves 1–15, but your target is defined using waves 13–15. That means the feature includes the same data that determines the label. If SRH drops in wave 14, it lowers both the feature and sets health_decline=1. How do you handle this circular dependency?

Context: *Did you consider computing features only from waves 1–10?*

Answer: We acknowledge that the way we computed SRH features could introduce leakage. In the current implementation, *fe_srh_mean*, *fe_srh_trend*, and *fe_srh_std* were calculated using all

available waves (1–15), even though the target uses waves 13–15. This means that if someone's health declined in wave 14, that decline both contributes to the feature (e.g. raising the mean or trend) *and* makes the target = 1, creating a circularity. To address this, we would restrict feature computation to exclude the target period. For instance, one approach is to use only waves 1–10 when computing the SRH statistics, so that features are based entirely on "past" data and do not directly see waves 13–15. Alternatively, one could redefine the target to begin later or stop feature windows earlier. In short, the fix is to recalc the trajectory features using only pre-decline waves (e.g. waves 1–10) so that the model never uses future information to predict health decline. This revision was not implemented in the current draft, but it would be an important correction to ensure the model truly predicts unseen decline rather than picking up information from the target period itself.

Product

Q1. '60 seconds every morning' — walk us through exactly what the user does in those 60 seconds, step by step.

Answer:

1. **Wake notification** — "Good morning. How are you feeling today?"
2. **Tap to record** — User holds button and speaks freely for 60 seconds:
   "I woke up around 7, didn't sleep well... kept waking up. Feeling kind of tired, my head's a bit heavy. I had some weird dreams. Not super motivated today but okay I guess..."
3. **AI listens to everything:**
   - **Words:** sleep quality, energy, mood, symptoms mentioned
   - **Tone:** flat vs. energetic, hesitation, confidence
   - **Voice biomarkers:** sighs, pauses, speech pace, vocal tremor
   - **Sentence structure:** coherence, complexity (cognitive load indicator)
4. **Transcription + Analysis** — Voice is processed (like Wispr Flow), converted to structured health signals via BioBERT or respective model.
5. **Follow-up chat** — AI asks 1-2 personalized questions based on what it heard:
   "You mentioned not sleeping well — was it hard to fall asleep, or did you keep waking up?"
6. **Personal model learns** — Every check-in trains the user's individual baseline. The AI learns what "normal" sounds like for THIS person.

In summary, from the user's standpoint the flow is to read the notification, record the response, optionally answer additional questions and read the result.

Q2. A user gets a risk score. What is the one specific thing the app tells them to do next?

Answer: The app responds in the same conversational way and supporting graphics:

Risk Level	Outcome/App Action	Suggested Advice/Next Steps
Normal / Baseline	Affirmation of current status. Consistent with personal baseline.	"Your health signals are stable today. Keep up the good work!"

Risk Level	Outcome/App Action	Suggested Advice/Next Steps
Mild shift / Early warning	Noticed slight, non-critical deviation in one or two areas (e.g., slightly lower energy, more pauses in speech).	**Continue Monitoring:** "We've noticed a small dip in your self-reported energy lately. Try focusing on getting an extra 30 minutes of sleep tonight." **Guided Reflection:** "Let's track this: Tell me more about what might be causing your [X] pattern shift."
Moderate shift / Elevated risk	Clear deviation from personal baseline across multiple signals (e.g., lower mood, disturbed sleep, increased speech volatility). Predictive score reaches threshold.	**Structured Check-in:** "We recommend taking a deeper health check-in now." **Lifestyle Intervention:** Suggest proven, low-barrier interventions: "Schedule 30 minutes of light exercise today", "Review your sleep hygiene" or “Reduce your stress level, try meditation, journaling or breath exercises”. **Summarize Data:** Begin preparing a summary: "If this continues for three more days, here is the data summary we can generate for your doctor." - with the report containing information since when normal cycle got disrupted.
Significant Shift / High Risk	Substantial or sustained deterioration in key health signals, potentially triggering a 'health decline' flag in the predictive model.	**Call to Action (Non-Diagnosis):** "Your current patterns represent a significant change from your normal health. This is a concerning signal that warrants professional attention." **Facilitate Doctor Visit:** Prompt the user to contact their primary care physician. Offer to immediately generate a comprehensive health summary of their app data (sleep, mood, speech metrics etc.) since the last stable period to share with the doctor.

Core Strategy: Maintain conversation and guide action

Instead of a single, blunt piece of advice, the app should use the outcome to fuel the ongoing conversational relationship:

- The most powerful non-diagnostic tool is providing the user with *data* about their changes (e.g., "For the last 7 days, your average sleep quality score is 2 points lower than your previous baseline, and your speech pace has slowed by 15%"). This is objective data, not a diagnosis.
- We should encourage self-reflection, and prompt users to consider causality. "Could this shift be related to a recent change in your diet, stress at work, or medication?"

- Clearly define for the user what thresholds will trigger more serious recommendations (e.g., contacting a doctor), empowering them with knowledge of the system's logic.