S 14 **P1 — Predictive risk scoring, P2 — Topic modeling for complaints, P3 — Linear & Time-decay attribution.**

Predictive Risk Scoring (P1)

Goal: predict which patients are at high risk (dropout, adverse event, non-adherence) so we can act early.

1) Simple idea

Turn historical data into a table of features per patient (or per patient-visit) and train a classifier/regressor that outputs a risk score (0–1).

2) Data & features (examples)

- Demographics: age, sex, site_id
- Historical behavior: days since last dose, number of missed doses past 30 days
- Clinical readings: vitals, lab values (last value, trend)
- Text signals: sentiment score of recent messages
- Device signals: missing pings count, battery low count
- Time features: days since enrollment

3) Model choices (simple → advanced)

- Logistic Regression / Random Forest (good baseline / explainable)
- Gradient Boosted Trees (XGBoost / LightGBM) — strong tabular performance
- Neural networks (if huge data)
- Calibrate outputs (Platt scaling / isotonic) to get well-behaved probabilities

4) Training pipeline (short)

1. Label data (e.g., label=1 if patient had adverse event within 14 days).
2. Train/test split by time or by patient (avoid leakage).
3. Feature engineering (missing value handling, normalization).
4. Train model, evaluate AUC, precision@k (top-k recall), calibration.
5. Export model and thresholds for action (e.g., score > 0.7 → high risk).

5) Simple Python sketch

```
# scikit-learn baseline
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler

X_train, y_train = ...  # prepared
pipe = make_pipeline(SimpleImputer(strategy='median'),
                     StandardScaler(),
                     RandomForestClassifier(n_estimators=100, random_state=42))
pipe.fit(X_train, y_train)
probs = pipe.predict_proba(X_val)[:,1]  # risk score 0..1
```

6) Evaluation & metrics

- ROC AUC, PR AUC

- Calibration plot (are predicted probs honest?)
- Confusion matrix at chosen threshold

7) Production notes

- Serve as microservice or batch job; store score and top contributing features (SHAP) for explainability.
- Retrain periodically; monitor data drift and model performance.

Topic Modeling for Complaints (P2)

Goal: automatically cluster complaints/feedback into topics (pain, nausea, scheduling issues) to find common problems.

1) Simple idea

Convert complaint texts into topic groups so humans can review aggregated issues.

2) Methods (easy → better)

- **Count-based clustering** (TF-IDF + KMeans) — easy, fast for classroom
- **LDA (Latent Dirichlet Allocation)** — classic probabilistic topics
- **Embedding + Clustering** (Sentence Transformers + HDBSCAN) — modern, high quality
- **BERTopic** — uses embeddings + class-based TF-IDF to produce meaningful topic labels

3) Steps (student-friendly)

1. Collect complaint texts and clean (lowercase, remove stopwords, minimal stemming).
2. Choose method: TF-IDF+KMeans for demo, or SentenceBERT for better clusters.
3. Fit model → get topic id for each complaint.
4. For each topic, show top words and example complaints.
5. Label topics manually (one-time human step) and monitor frequency over time.

4) Simple code (TF-IDF + KMeans)

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans

docs = [...]  # complaint texts
vec = TfidfVectorizer(max_features=2000, ngram_range=(1,2))
X = vec.fit_transform(docs)
km = KMeans(n_clusters=8, random_state=42)
labels = km.fit_predict(X)
# show top terms per cluster
terms = vec.get_feature_names_out()
for i in range(8):
```

```
    center = km.cluster_centers_[i]
    top_idx = center.argsort()[-10:][::-1]
    print("Topic", i, [terms[j] for j in top_idx])
```

5) How to explain topics to students

- Topic = group of complaints that often share words/semantics
- Show sample texts per topic; ask students to suggest a label

6) Monitoring and use

- Track topic volumes over time; alert when a clinical side-effect topic spikes.
- Combine with sentiment and risk scores for prioritized review.

Linear & Time-Decay Attribution Models (P3)

Goal: allocate credit for conversions across touchpoints. Two simple, interpretable models: linear and time-decay.

1) Set up sequences

For each conversion, create the ordered list of touchpoints that occurred before the conversion (e.g., Email → Ad → SMS → Conversion).

2) Linear attribution (equal credit)

- **Idea:** Every touch in the sequence gets equal credit.
- If a conversion value = 1 and there are 4 touches → each touch gets 0.25.

SQL sketch

```
-- assume touches table ordered by time for each conversion
SELECT conversion_id, touch, 1.0 / COUNT(*) OVER (PARTITION BY
conversion_id) as credit
FROM touches;
```

3) Time-decay attribution (recent touches count more)

- **Idea:** weight touches based on how close they are to conversion.
- Simple weight: exponential decay w = exp(-λ * Δt) where Δt = time difference (days) between touch and conversion, λ controls decay speed.
- Normalize weights so they sum to 1 per conversion.

Formula example

```
w_i = exp(-lambda * delta_days_i)
credit_i = w_i / sum_j w_j
```

Pick lambda so that a touch 7 days prior has, say, half the weight of a touch on the conversion day.

4) Code sketch (Python)

```
import numpy as np
def time_decay_credits(touch_times, conv_time, lam=0.3):
    delta_days = np.array([(conv_time - t).days for t in touch_times])
    w = np.exp(-lam * delta_days)
    return w / w.sum()
```

5) How to choose λ (lambda)

- Small λ → slow decay (all touches matter)

- Large $\lambda \rightarrow$ fast decay (only recent touches matter)
  Explain with examples: compute credits for the same sequence with different λ to visualize.

6) Evaluation & reporting

- Aggregate credited conversions by channel (sum credits per channel) and compute CPA/ROI.
- Compare simple models (first/last/linear/time-decay) to see differences.

Practical tips & pitfalls

- **Avoid leakage** in predictive models: don't use future info. Split by time/patient.
- **Label quality** is everything: especially for risk labels and topic validation.
- **Explainability matters**: show feature importances / SHAP for risk scores.
- **Human-in-loop**: always have human review for topic labels and high-risk flags.
- **Scale**: simple methods (TF-IDF, RandomForest) are enough to start; optimize later.

Main Idea

In this stage, we add **smart and predictive logic** to our project.

Part	Logic Type	Purpose
P1	Predictive Risk Scoring	Predict which patient or user may face risk soon
P2	Topic Modeling	Automatically group similar complaints or feedback
P3	Linear / Time-Decay Attribution	Fairly share credit for conversions between marketing channels

P1 — Predictive Risk Scoring

Goal

To **predict risk in advance**, such as which patients are likely to:

- Miss a dose
- Drop out of a study
- Have side effects

Simple Steps

1. **Collect data**
   Example:
   - Age, gender
   - Missed doses count
   - Health readings
   - Last visit date

- Sentiment score from feedback

2. **Give labels**
   - 1 → High risk
   - 0 → Low risk
3. **Train a model**
   - Use a simple machine learning model like **Logistic Regression** or **Random Forest**.
   - Model learns patterns from past data.
4. **Predict risk score**
   - Model gives a score between **0 and 1** (0 = safe, 1 = very risky)
5. **Take action**
   - If score > 0.7 → send alert to team
   - If score < 0.3 → mark as safe

Example

Patient	Missed Doses	Health Score	Predicted Risk
P001	0	90	0.1 (Low)
P002	3	60	0.8 (High)

Team can now **focus early** on high-risk patients.

P2 — Topic Modeling (for Complaints)

Goal

To **automatically group** complaints or feedback into topics like:

- Side effects
- Scheduling issues
- App problems
- Billing issues

Simple Steps

1. **Collect text feedback**
   Example:
   - "I feel tired after the new dose."
   - "The app keeps crashing."
   - "My appointment was delayed."
2. **Clean the text**
   - Remove extra words, punctuation, stopwords.
3. **Find common topics**
   - Use AI to find similar words and group them.
   - Each group becomes a **topic**.
4. **Label topics manually**
   - Topic 1 → Side Effects

- Topic 2 → Technical Issue
- Topic 3 → Scheduling Problem

Example

Feedback	Topic
"I feel tired after dose."	Side Effect
"App not loading."	Technical Issue
"Appointment got delayed."	Scheduling Issue

Helps identify which problems are most common.

P3 — Linear / Time-Decay Attribution

Goal

To understand **which marketing activity** (like email, ad, SMS) contributed most to a conversion.

Example Journey

User's journey before signing up:

Step	Channel	Date
1	Email	1 Oct
2	Facebook Ad	3 Oct
3	Google Ad	5 Oct
4	Conversion	6 Oct

1. Linear Attribution

Every touch (step) gets **equal credit**.

There are 3 touches → each gets 1/3 credit (33%).

Channel	Credit
Email	0.33
Facebook Ad	0.33
Google Ad	0.33

2. Time-Decay Attribution

The **closer** a touchpoint is to conversion, the **more credit** it gets.

Channel	Days Before Conversion	Weight
Email	5 days	0.2
Facebook Ad	3 days	0.3
Google Ad	1 day	0.5

So **Google Ad** gets more credit because it's the last one before conversion.

Simple Comparison

Model	Meaning	Credit Example
First-Touch	First channel gets all credit	Email = 100%
Last-Touch	Last channel gets all credit	Google Ad = 100%
Linear	Equal credit to all	Each = 33%
Time-Decay	More credit to recent touches	Google Ad > Email

Tools You Can Use

Task	Easy Tool Option
Predictive Risk	Python (Scikit-learn) / Excel regression
Topic Modeling	Python (TF-IDF + KMeans)
Attribution	Excel / SQL queries

In Short

Module	What It Does	Example
P1	Predict who may face risk	Identify patients likely to miss dose
P2	Group similar feedback	Cluster complaints by topic
P3	Distribute conversion credit	Give fair credit to marketing channels