**Astraal LXP**

**Content Effectiveness Prediction Model (CEPM)**

*(Standalone Early Quality Intelligence Module)*

**1️⃣ Problem Framing (Correct & Institutional-Safe)**

**Current Situation**

Most platforms identify poor-performing content only after:

* Low completion rates
* Poor assessment outcomes
* Negative feedback
* Instructor complaints

By then:

* Many learners are affected
* Engagement has dropped
* Faculty intervention is delayed

**Objective**

Predict potential content effectiveness issues early using:

* Initial engagement patterns
* Time-to-completion signals
* Drop-off points
* Early assessment outcomes

Without:

* Blaming instructors
* Labeling learners
* Triggering automatic penalties

This is early detection, not automatic correction.

**2️⃣ What "Content Effectiveness" Means**

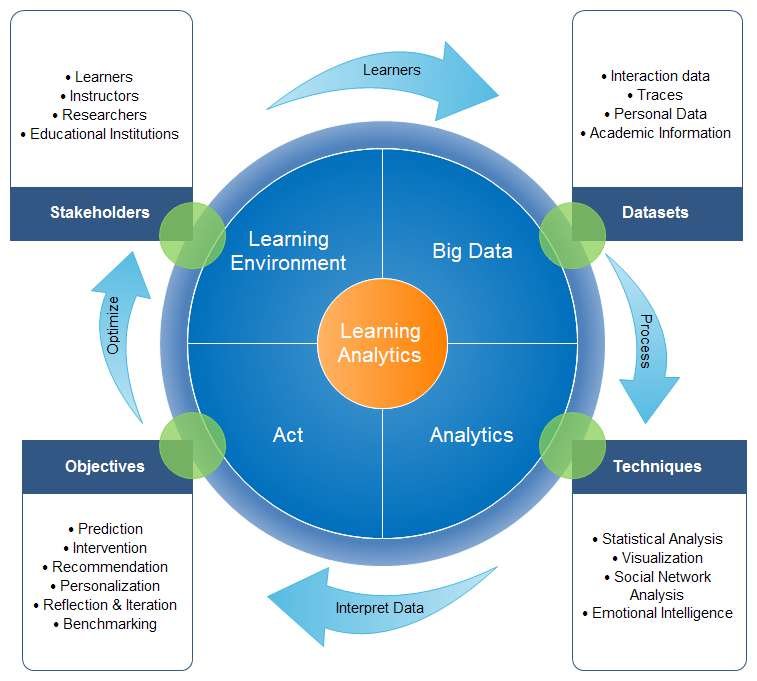
Effectiveness is not:

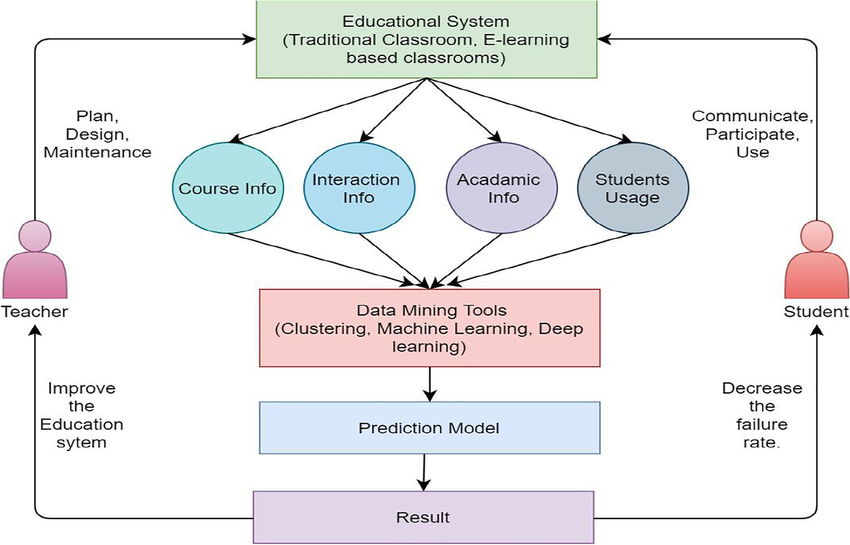
* Popularity
* Time spent
* Number of clicks

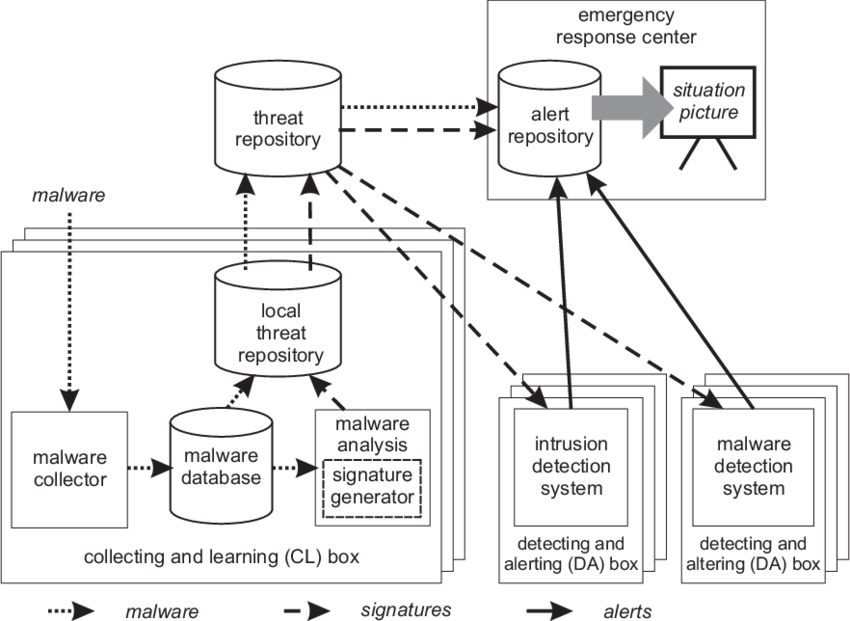
Effectiveness =

The ability of a content unit to sustain engagement and support successful learning outcomes relative to expectations.

**3️⃣ System Architecture (UWAMP + GoDaddy Compatible)**







**Architecture Overview**

Learner Activity Logs (PHP)

↓

Engagement & Assessment Signals (MySQL)

↓

Feature Aggregation Engine (Python CLI)

↓

Prediction Model

↓

Content Risk Scores (MySQL)

↓

Faculty Quality Dashboard (PHP)

Constraints:

* No real-time streaming
* Python runs via cron
* Shared-hosting compliant
* No external ML servers

**4️⃣ Data Model**

**4.1 Content Units (Existing)**

content\_units

**4.2 Content Effectiveness Signals**

CREATE TABLE content\_signals (

unit\_id INT,

avg\_time\_spent FLOAT,

dropoff\_rate FLOAT,

early\_assessment\_score FLOAT,

engagement\_variance FLOAT,

sample\_size INT,

calculated\_on DATETIME

);

**4.3 Prediction Output**

CREATE TABLE content\_effectiveness\_predictions (

unit\_id INT,

predicted\_effectiveness FLOAT,

risk\_level VARCHAR(50),

confidence\_score FLOAT,

calculated\_on DATETIME

);

**5️⃣ Feature Engineering (Critical Section)**

We use early-stage signals:

| **Feature** | **Meaning** |
| --- | --- |
| Avg Time Ratio | Actual vs expected |
| Drop-off Rate | % leaving before completion |
| Early Assessment Score | First quiz result average |
| Engagement Variance | Inconsistent activity |
| Revisit Rate | Learners returning |

**Example Feature Vector**

[

avg\_time\_ratio,

dropoff\_rate,

early\_score,

engagement\_variance,

revisit\_rate

]

**6️⃣ ML Model Choice**

Because environment is shared hosting:

Recommended:

* Logistic Regression (binary effective / at-risk)
* Random Forest (optional)
* Gradient Boosted Trees (offline training)

Training offline.  
Inference via CLI.

**7️⃣ Python Prediction Engine**

**7.1 Training Script (Offline)**

from sklearn.ensemble import RandomForestClassifier

import pickle

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

pickle.dump(model, open("content\_model.pkl","wb"))

**7.2 Prediction Script (CLI)**

content\_effectiveness\_engine.py

import mysql.connector

import pickle

import pandas as pd

model = pickle.load(open("content\_model.pkl","rb"))

db = mysql.connector.connect(...)

cursor = db.cursor(dictionary=True)

cursor.execute("SELECT \* FROM content\_signals")

rows = cursor.fetchall()

df = pd.DataFrame(rows)

X = df[['avg\_time\_spent','dropoff\_rate','early\_assessment\_score','engagement\_variance']]

predictions = model.predict\_proba(X)[:,1]

for i, row in df.iterrows():

risk = "Stable"

if predictions[i] < 0.4:

risk = "Needs Review"

cursor.execute("""

INSERT INTO content\_effectiveness\_predictions

VALUES (%s,%s,%s,%s,NOW())

""",(row['unit\_id'], predictions[i], risk, 0.8))

**8️⃣ Faculty Dashboard UI Design**

**Page: Content Quality Insights**

**Table Layout**

| **Content Unit** | **Predicted Effectiveness** | **Risk Level** | **Confidence** | **Action** |
| --- | --- | --- | --- | --- |

**Risk Categories (Neutral Language)**

Instead of:

* Poor
* Weak
* Failing

Use:

* Stable
* Monitor
* Needs Review

**Drill-Down View**

Faculty sees:

* Drop-off graph
* Engagement distribution
* Early assessment distribution
* Comparison to course average

**9️⃣ Governance Safeguards**

* Minimum sample size required
* Confidence score displayed
* No automatic content removal
* Faculty review required
* No learner-visible labels

**🔟 Integration with Other Modules**

| **Module** | **Integration** |
| --- | --- |
| Engagement Engine | Supplies engagement signals |
| Sequencing Engine | Can optionally avoid high-risk units |
| Knowledge Graph | Identifies skill clusters impacted |
| Career Readiness | Flags if key skill-building unit is weak |

Still standalone.

**11️⃣ Deployment Model**

* Nightly cron job
* Weekly recalibration
* Model retraining quarterly
* Admin download of signals possible

**12️⃣ Ethical Positioning**

This module evaluates:

Content quality  
Not instructor quality  
Not learner ability

Microcopy for faculty:

These predictions indicate patterns in engagement and outcomes. They support instructional review and refinement.

**13️⃣ Strong Module Names**

* Content Effectiveness Intelligence™
* Learning Unit Quality Insights
* Early Content Risk Detection Engine
* Instructional Effectiveness Predictor

**PART 1️⃣**

**Integrating Content Effectiveness with Graph-Based Skill Impact Analysis**

This integration leverages your existing **Astraal Knowledge Graph Backbone**.

**1.1 Core Idea**

If a content unit is predicted as:

“Needs Review”

We must ask:

* Which skills does this unit build?
* Which competencies are affected?
* Which career roles depend on those skills?
* Which certifications validate them?

We move from:

Content Risk  
→ Skill Impact  
→ Career Impact

**1.2 Graph Relationships Used**

From your graph schema:

Course → BUILDS → Skill

Skill → REQUIRES → Career Role

Assessment → VALIDATES → Skill

Certificate → CERTIFIES → Skill Cluster

**1.3 New Integration Flow**

Content Effectiveness Prediction

↓

Fetch Impacted Skills (Graph)

↓

Fetch Dependent Career Roles

↓

Compute Structural Impact Score

↓

Faculty Impact Visualization

**1.4 Required Graph Queries**

**Step 1 – Get Skills Built by Content**

SELECT to\_node

FROM kg\_edges

WHERE from\_node = :content\_unit\_node

AND relationship\_type = 'BUILDS';

**Step 2 – Get Career Roles Dependent on Those Skills**

SELECT DISTINCT from\_node

FROM kg\_edges

WHERE to\_node IN (:skill\_list)

AND relationship\_type = 'REQUIRES';

**1.5 Structural Impact Score**

We calculate:

Impact Score =

Predicted Risk \*

Skill Criticality Weight \*

Career Dependency Weight

Where:

* Skill criticality = weight in REQUIRES edge
* Career dependency = number of roles depending on skill

**1.6 New Table for Skill Impact**

CREATE TABLE content\_skill\_impact (

unit\_id INT,

skill\_node INT,

career\_node INT,

structural\_impact FLOAT,

calculated\_on DATETIME

);

**1.7 Python Graph Impact Engine**

content\_impact\_engine.py

# Pseudocode logic

risk\_score = predicted\_effectiveness

for skill in skills\_built\_by\_unit:

fetch career\_roles

for role in roles:

impact = risk\_score \* skill\_weight \* role\_dependency

store impact

Run nightly after prediction engine.

**1.8 What This Enables**

Faculty now sees:

* “Unit A underperforms”
* “It impacts Skill X”
* “Skill X is critical for Data Analyst role”
* “3 career pathways affected”

This turns analytics into structural insight.

**PART 2️⃣**

**Faculty Quality Console UI Design**

This is the control center for instructional governance.

**2.1 Console Architecture**

Content Risk Overview

↓

Skill Impact Mapping

↓

Career Dependency View

↓

Remediation Suggestions

↓

Version & Audit History

**2.2 Page 1: Content Risk Overview**

**Layout**

| **Unit** | **Effectiveness** | **Risk Level** | **Impacted Skills** | **Career Impact** | **Action** |
| --- | --- | --- | --- | --- | --- |

Risk levels:

* Stable
* Monitor
* Needs Review

No red flashing indicators.

**2.3 Page 2: Skill Impact Drill-Down**

When faculty clicks “View Impact”:

**Panel Layout**

**Section A – Affected Skills**

| **Skill** | **Criticality Weight** | **Impact Score** |
| --- | --- | --- |

**Section B – Career Pathways Impacted**

| **Career Role** | **Dependency Level** | **Impact Severity** |
| --- | --- | --- |

**2.4 Visual Graph Overlay**

Interactive graph:

[Content Unit]

↓ BUILDS

[Skill Cluster]

↓ REQUIRES

[Career Role]

Highlight:

* Weak content node
* Impacted skills
* Downstream roles

Use D3.js force graph.

**2.5 Page 3: Diagnostic Signals**

Show raw indicators:

* Drop-off graph
* Assessment distribution
* Engagement variance
* Comparison to course average

Purpose: Evidence-based review.

**2.6 Page 4: Remediation Controls**

Faculty can:

* Flag for revision
* Add supplemental content
* Adjust difficulty
* Request peer review
* Recompute effectiveness

**2.7 Audit Trail**

CREATE TABLE content\_revision\_log (

unit\_id INT,

action\_taken TEXT,

revised\_by INT,

timestamp DATETIME

);

This ensures governance.

**2.8 UX Principles**

* No blame language
* No instructor ranking
* No performance labels
* Focus on instructional improvement

**2.9 Faculty Microcopy (Important)**

Header:

Content Quality Insights

Subtext:

These insights are generated from early engagement and assessment patterns. They indicate potential areas for instructional review and refinement.

Impact Section:

Structural impact reflects how this unit contributes to skill development and downstream career pathways.

**3️⃣ Full Integrated Architecture**

Learner Activity

↓

Content Effectiveness Model

↓

Graph Skill Mapping

↓

Career Dependency Analysis

↓

Structural Impact Score

↓

Faculty Quality Console

All running on:

* MySQL graph schema
* Python CLI processing
* PHP rendering layer
* Cron-based recalculation

Shared-hosting compliant.

**4️⃣ Strategic Power of This Integration**

This now enables Astraal to claim:

* Early content risk detection
* Structural skill impact visibility
* Career-aligned instructional refinement
* Evidence-based curriculum improvement

Very few platforms do structural impact analysis.