Student Behaviour Analysis and Prediction on Online Learning Platforms Based on Machine Learning

By

Geng Rui

Submitted to

The University of Roehampton

In partial fulfilment of the requirements

for the degree of

Master of Science

in

Computing

Declaration

I hereby certify that this report constitutes my own work, that where the language of others is used, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions, or writings of others.

I declare that this report describes the original work that has not been previously presented for the award of any other degree of any other institution.

Name: Geng Rui

Date: 08/11/2025

Signature: 签名

Acknowledgements

I would like to thank my supervisor at the University's teaching centre, Dr SURYANTO, for patient guidance, clear methodological advice, and timely feedback from proposal through to submission. I am grateful to Dr John Moxon at the University of Roehampton, with whom I co-authored an earlier paper; our discussions helped shape this project. I also thank my second marker, Dr Chandrashekhar Kumbhar, and the reviewers for their constructive feedback.

My appreciation extends to the programme and administrative teams at the University of Roehampton and the teaching centre for their support throughout the MSc. I am thankful to my classmates in the MSc cohort for helpful feedback and encouragement.

This project benefited from the open-source communities behind Streamlit, scikit-learn, Plotly, and SQLAlchemy.

Finally, I thank my family and friends for their continued encouragement. Any remaining errors are my own.

Abstract

This study addresses a methodological gap in vocational learning analytics: existing predictive systems either optimise model accuracy at the expense of transparency or preserve interpretability while underserving pedagogical actionability. To resolve this tension, the project develops an explainability-first pipeline that deliberately foregrounds legibility to instructors as a design constraint, not an afterthought. The pipeline integrates a relational data layer, a transparent modelling baseline, SHAP-based feature attribution, and a lightweight teacher-facing dashboard. A staged synthetic-to-real methodology is adopted to comply with privacy governance while still enabling meaningful evaluation of system behaviour under realistic behavioural variation.

Evaluation demonstrates that interpretability can be operationalised without a disproportionate loss in performance: the pipeline maintains high predictive utility while producing explanations that align with instructors' mental models of disengagement risk. Crucially, pedagogical cost asymmetry — where false negatives incur substantially higher consequences than false positives — is embedded into the assessment logic, moving beyond conventional algorithmic benchmarking toward classroom-oriented validity. The primary contribution of this work is therefore methodological rather than algorithmic: it offers an ethically deployable, pedagogically legible research blueprint for vocational learning analytics, establishing a reproducible foundation upon which future ICVE-based deployment trials can responsibly proceed.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Problem Description, Context and Motivation

1.1.1 Problem Scope and Learning Context

Online learning platforms log multidimensional behaviours (e.g., login frequency, time-on-task, quiz attempts, progress). In vocational education, learners often balance study with employment and exhibit irregular engagement patterns. Despite the abundance of behavioural data, instructors lack timely, transparent signals to identify at-risk learners for early interventions.

This project focuses specifically on vocational contexts (ICVE) where generalising insights from university/MOOC studies is non-trivial due to different participation patterns and institutional constraints.

1.1.2 Stakeholders and Educational Impact

Students may experience disengagement and poor outcomes if risks remain hidden; instructors face limited visibility over large cohorts; institutions risk suboptimal retention and resource allocation.

A system that surfaces explainable risk factors (e.g., insufficient time spent or low quiz activity) supports actionable, equitable teaching decisions.

The project therefore integrates a feature-attribution module (SHAP) to build trust and bridge the gap between ML outputs and teaching needs in practice.

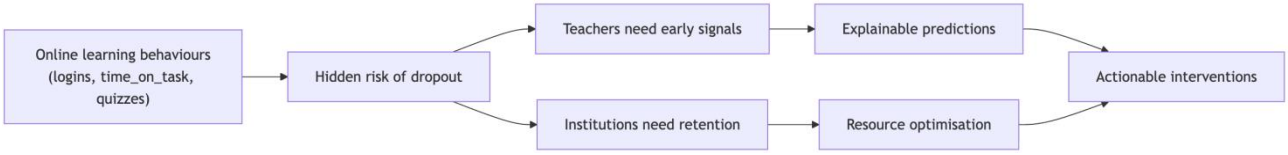

Figure 1. Vocational learning analytics: context and stakeholders

1.1.3 Research Gap and Motivation

Prior work often trades interpretability for accuracy (e.g., deep sequence models), undermining adoption in classrooms; conversely, simpler models are interpretable but can miss complex patterns in high-dimensional, sparse data [1].

Vocational contexts are further underrepresented, intensifying transferability challenges [2]. The present study is motivated by the need to balance accuracy and clarity through a modular, explainable pipeline tailored for vocational platforms like ICVE.

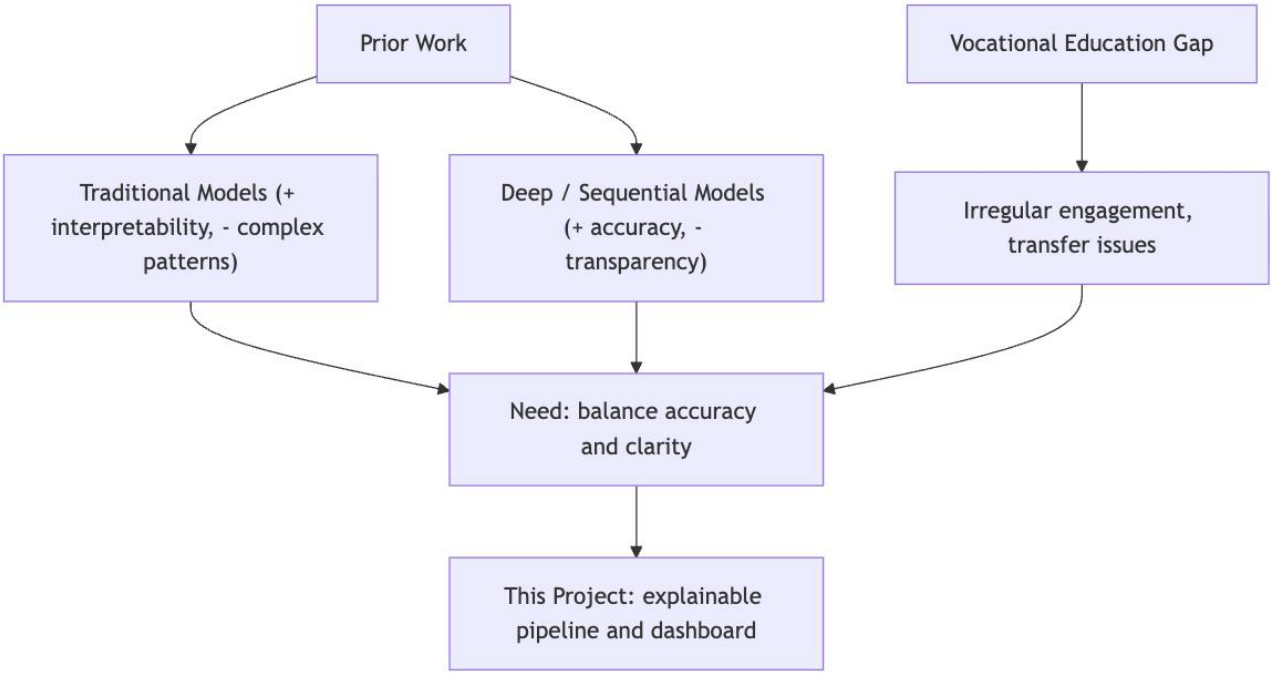

Figure 2. Problem–gap–motivation map

1.2 Objectives

1.2.1 Aims and Quantitative KPIs

The aim is to design, implement, and evaluate a database-backed, explainable pipeline that predicts learner risk and communicates drivers of predictions to instructors via a lightweight dashboard.

Success criteria: Accuracy $\geqslant$ 80%, F1 $\geqslant$ 0.75, AUC $\geqslant$ 0.80, dashboard load time < 3s. These KPIs explicitly guide model selection and the evaluation workflow later in Chapter 4.

1.2.2 Research Questions (RQs)

RQ1. To what extent do synthetic datasets (MSc phase) approximate realistic behavioural patterns for pipeline validation?

RQ2. Under the defined pipeline, what predictive performance is achieved on held-out data using a transparent baseline versus tree-based comparators?

RQ3. How effectively can explanations be read and acted upon by instructors within the dashboard workflow, and how will this scale to authentic ICVE data post-MSc?

1.3 Methodology

1.3.1 Design

A modular end-to-end architecture is adopted: MySQL data layer → feature preparation → model training and selection → persistence of scaler/model → explainability (SHAP) → Streamlit/Plotly visualisation. Modularity enables incremental upgrades without sacrificing interpretability.

A lightweight single-student inference page is also provided to distribute the trained artefact to colleagues without re-training or database access.

1.3.2 Testing and Evaluation

All quantitative evaluation is consolidated in Chapter 4 via an "Evaluate Model" workflow: synthetic-data realism checks (descriptive statistics and correlation heatmaps), predictive metrics (Accuracy, Precision, Recall, F1, AUC), confusion matrix, and a classification report. Usability checks employ simulated users during the MSc phase.

1.3.3 Project Management

Work proceeds as a vertical slice first (end-to-end MVP) followed by targeted refinements. Key milestones: V1.0 by Oct 5, V2.0 by Oct 13, Final by Oct 26, and Report & Presentation by Nov 6.

1.3.4 Technologies and Processes

Python (pandas, scikit-learn, joblib), MySQL (SQLAlchemy/pymysql), Streamlit + Plotly, and SHAP are used to ensure a lightweight yet explainable workflow; at least two core visualisations are provided: overall engagement trends and individual prediction results.

1.4 Legal, Social, Ethical and Professional Considerations

1.4.1 Data Protection and Compliance

During the MSc phase only synthetic data are processed; post-MSc integration with ICVE will follow data-protection requirements (e.g., PIPL/GDPR), including anonymisation, encryption, and least-privilege access.

The single-student page operates without storing personally identifiable information by default; when deployed with authentic data, anonymised IDs and role-based access are enforced.

1.4.2 Ethics and Bias Mitigation

The system functions as decision support; instructors retain full agency. Bias is mitigated by transparent features, threshold audits, and error-pattern inspection in evaluation.

1.4.3 Professional Integrity

Procedures are documented for replicability; results are reported faithfully; academic writing adheres to a third-person, observational style.

1.5 Background

Global vocational enrolment and online/hybrid learning are growing, with ICVE providing structured digital resources; yet predictive analytics for vocational contexts remains underexplored—motivating this project's focus [3], [4].

1.6 Structure of Report

Chapter 2 reviews literature and enabling technologies; Chapter 3 details the implementation; Chapter 4 presents all evaluation results generated by the "Evaluate Model" workflow; Chapter 5 concludes with contributions, future work, and reflections.

To orient the reader, Table 1 maps each research question to the evidence surfaces and the chapter sections where they are reported.

Table 1. Research questions and evidence surfaces used in this report

RQ	Aim	Primary evidence view(s)	Where in report
RQ1: Synthetic data realism	Check plausibility of the synthetic cohort	Descriptive statistics; Correlation heatmap	§4.2.1; §4.4.1 blocks (1)-(2)
RQ2: Predictive performance	Assess classifier quality on hold-out	Metrics table; Confusion matrix; Text report	§4.3.1–§4.3.2; §4.4.1 blocks (3)-(5)
RQ3: Usability & interpretability	Support teacher decisions and transparency	SHAP-based explanations; Dashboard UX flows	§3.5; §4.4.2; §3.6

Chapter 2 Literature – Technology Review

2.1 Literature Review

2.1.1 Engagement Constructs and Measures

Student engagement is commonly conceptualised along behavioural, cognitive and affective dimensions; in online settings, observable proxies usually centre on behavioural traces such as login frequency, time-on-task, quiz attempts, completion rate and progress [5]. These traces are increasingly abundant as online and blended learning continue to expand, but they require careful interpretation and validation before they can inform risk detection and intervention design [6].

In vocational education, unstable study rhythms and dual commitments (e.g., study – work balance) complicate inference from raw logs and accentuate the need for methods that are both robust and readable to instructors.

The project positions these behavioural measures as first-class signals to be engineered into features and later paired with explanations so that teachers can assess plausibility and act upon the outputs. Alignment among epistemology, pedagogy and assessment is central to making such analytics genuinely actionable for teachers [7].

2.1.2 Traditional Predictive Models and Interpretability

Traditional models such as Logistic Regression and Decision Trees remain valuable baselines in learning analytics: they are light-weight, quick to train, and—critically—interpretable [1], [8].

Their limitations are well known: linear or axis-aligned decision surfaces may underfit complex, non-linear phenomena, especially when features interact or when the signal-to-noise ratio is low. Nevertheless, the transparency of coefficient weights (LR) and split rules (DT) makes such models attractive for classroom adoption where human-in-the-loop validation is essential.

This project deliberately treats Logistic Regression as the baseline comparator to establish a transparent reference for later analysis, consistent with the project proposal's emphasis on interpretability during the MSc phase.

2.1.3 Sequential and Deep Models for Behavioural Data

Deep and sequential architectures (e.g., RNN/LSTM) can capture temporal dependencies and typically deliver stronger predictive accuracy on behavioural logs, but they reduce transparency and raise adoption barriers for instructors [1], [9].

Research indicates a persistent trade-off between performance and interpretability, suggesting that explainability mechanisms must accompany any accuracy gains to promote trust and actionability in teaching. The project therefore aims to balance accuracy and clarity by first building an interpretable pipeline and then (post-MSc) considering more complex models under stronger governance.

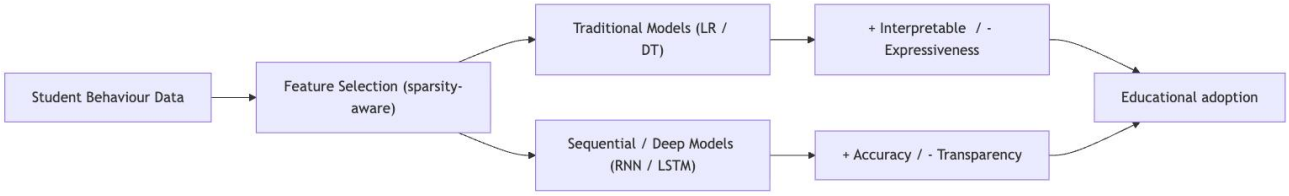

Figure 3. Methodological Landscape for Student Prediction

2.1.4 Feature Selection for High-Dimensional Sparse Data

Online learning platforms generate high-dimensional, sparse data; feature selection is therefore critical to denoise, improve generalisation, and reduce computational load [10].

The proposal explicitly motivates a sparsity-aware approach to feature processing and selection; in the MSc phase, this is combined with a single core model to keep the system transparent and feasible for validation on synthetic datasets. Post-MSc, additional models may be incrementally introduced under the same modular design to preserve clarity while extending accuracy.

2.1.5 Vocational Education Gap

Much of the literature and many public datasets come from research universities or MOOCs; transferability to vocational contexts is limited due to different participation patterns, support structures, and assessment ecologies.

The project directly targets these gaps by (i) focusing on vocational learners and (ii) aligning the artefact with ICVE course repositories for future validation when authentic data and teacher users can be engaged. During the MSc phase, synthetic datasets are

used to prototype and verify the pipeline without privacy risks, with the explicit plan to conduct broader usability studies post-MSc.

In line with UNESCO’s Global Education Monitoring and OECD’s Education at a Glance indicators, the artefact targets skills-oriented, teacher-readable analytics that can be operationalised within TVET policy frameworks [3], [4].

2.1.6 Research Foundation and External Validation: ICISCAE 2025

A preliminary paper derived from this project has been accepted by ICISCAE 2025, providing external validation of the research positioning and method [11]. To contextualise the literature review and clarify the project's research foundation, this dissertation documents external peer-reviewed recognition of its core idea. The short paper "A Learning Feature Selection Model for High-Dimensional Sparse Data of Students' Online Behaviour" has been accepted by ICISCAE 2025, which supports the novelty and relevance of the sparsity-aware, explainable approach adopted in this work.

The acceptance does not pre-empt the MSc evaluation—Chapter 4 independently assesses the implemented pipeline on synthetic cohorts—but it strengthens the rationale for focusing on interpretable feature processing and a modular architecture that can be validated on vocational platforms such as ICVE. This figure provides the acceptance evidence.

2025 IEEE 8th International Conference on Information Systems and Computer Aided Education (ICISCAE 2025) Dalian, China

NOTIFICATION OF ACCEPTANCE

Dear Author(s):

On behalf of the 2025 IEEE 8th International Conference on Information Systems and Computer Aided Education (ICISCAE 2025), we're glad to inform you that your paper:

Paper ID: ICISCAE-31735

Paper Title: A learning feature selection model for high-dimensional sparse data of students' online behavior

Author(s): Rui Geng , John R. L. Moxon, BoHui Wang

has been Accepted!

ICISCAE 2025 aims to bring researchers, engineers and students to the areas of information systems, computer engineering, information technology, network engineering and computer aided education, and will provide an international forum for sharing the most advanced research results, experiences and original research contributions on related topics.

All accepted papers will be published by IEEE CS (Computer Society) CPS, and will be submitted to EI Compendex, Thomson ISTP and Elsevier SCOPUS databases.

Engineering Village IEEE

Figure 4. ICISCAE 2025 acceptance confirmation for "A Learning Feature Selection Model for High-Dimensional Sparse Data of Students' Online Behaviour"

Together with the methodological synthesis in Sections 2.1.1 – 2.1.5, this external validation motivates the chosen balance of accuracy and clarity and links the present implementation to the wider research discourse.

2.2 Technology Review

2.2.1 Data Storage and Schema Options

A relational store is suited to tabular behavioural logs with stable schemas (e.g., student_id, login_count, time_spent, quiz_attempts, completion_rate, quiz_score, progress, timestamps). Such structures simplify SQL-based validation, incremental updates, and reproducible extraction into pandas dataframes for modelling.

The project's implementation (see Chapter 3) follows this pattern, enabling first-run data population with synthetic rows and a clean read path for training and inference in the dashboard.

2.2.2 Modelling Libraries and Pipelines

Given the MSc-phase constraints (time, transparency, and maintainability), scikit-learn provides a pragmatic baseline toolkit. It offers consistent estimator APIs, standard preprocessing (e.g., scaling), straightforward model selection and evaluation utilities, and reliable persistence via joblib, which together reduce cognitive load and make results reproducible [8].

The proposal sets out a staged plan—start with a single transparent core model (e.g., Logistic Regression), then consider tree-based or gradient ensemble models as the system matures. This staged approach is supported by modular pipeline design to avoid entanglement when swapping or adding models.

2.2.3 Visualisation Frameworks

A lightweight web dashboard facilitates iterative validation with instructors and other stakeholders. Streamlit with Plotly covers interactive tables, scatter/donut/bar charts and heatmaps with minimal boilerplate, reduces glue code, and speeds up analytics-to-insight cycles.

The project's dashboard targets two core visualisation categories—overall engagement trends and individual prediction results—so that both cohort-level and learner-specific views are available for review.

2.2.4 Explainability Tooling

Explainability is implemented with SHAP because it provides model-agnostic and theoretically consistent attributions for tabular classifiers, enabling both global importance summaries and local, per-student explanations [12]. In education, explainability requirements and patterns have been systematically surveyed; these recommendations shape our presentation choices (e.g., plain-language feature names, stable visuals) and caution against over-claiming from single explanations [13].

SHAP provides consistent, model-agnostic attributions that help instructors read which behavioural features (e.g., quiz attempts vs time spent) most influence risk classification. The proposal requires an interpretability module to accompany predictions in the MSc phase; the dashboard will surface such explanations alongside metrics so that teachers can triangulate model outputs with domain expectations.

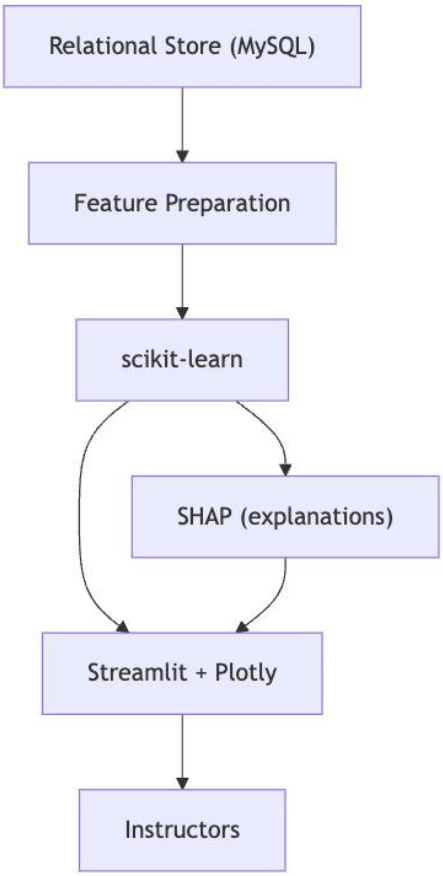

Figure 5. Technology choices and rationale

2.3 Critical Synthesis Summary

The literature underscores an enduring tension between accuracy and interpretability in student-risk prediction.

Table 2. Synthesis of prior themes and implications for this project

Theme	What prior work shows	Implication for this project
Interpretability vs accuracy	Deep/sequential models improve accuracy but reduce transparency	Prefer transparent baselines; foreground matrix reading
Vocational context	Vocational cohorts differ from university-paced settings	Keep features simple and behaviour-linked; validate realism first
Behavioural features	Engagement signals (logins, time-on-task, quiz attempts) are actionable	Use the same three features end-to-end and align advice to them
Evaluation framing	Numbers alone are insufficient for pedagogy	Bundle metrics with matrix + explanations for decisions

Traditional models provide transparency but may miss complex patterns; deep or sequential models improve accuracy but erode readability. Vocational contexts magnify these issues due to sparse, irregular engagement and the practical need for teacher-facing explanations.

Consequently, the project adopts a staged, modular strategy:

(i) build an end-to-end interpretable pipeline with a transparent baseline and feature selection tuned for sparsity.

(ii) validate with synthetic datasets and a focused dashboard.

(iii) plan for post-MSc extension to more complex models under explainability and governance.

The technology stack—relational storage for stable schemas, scikit-learn for concise pipelines, SHAP for explanations, Streamlit/Plotly for rapid visual analytics—aligns with these goals and with the proposal’ s KPIs and validation plan.

This alignment ensures that Chapter 3 can implement the pipeline cleanly and that Chapter 4 can evaluate it comprehensively using the “Evaluate Model” workflow (descriptive statistics, correlation heatmaps, metrics, confusion matrix, textual report) before transitioning to real-data validation in Chapter 5.

Chapter 3 Implementation

3.1 System Overview and Design

3.1.1 End-to-End Architecture

The artefact implements a modular pipeline that connects a relational data layer to a lightweight web dashboard. The flow is one-way from authoring to runtime:

(i) database initialisation and connectivity.

(ii) first-run data population via a synthetic generator when necessary.

(iii) feature preparation and preprocessing.

(iv) offline model training and selection.

(v) persistence of artefacts (trained_model.pkl, scaler.pkl, shap_explain.png) under models/.

(vi) cohort-level risk inference at runtime by the dashboard.

(vii) instructor-facing visualisation.

The dashboard loads the database tables and the persisted artefacts for prediction; it does not retrain nor write back to the models directory.

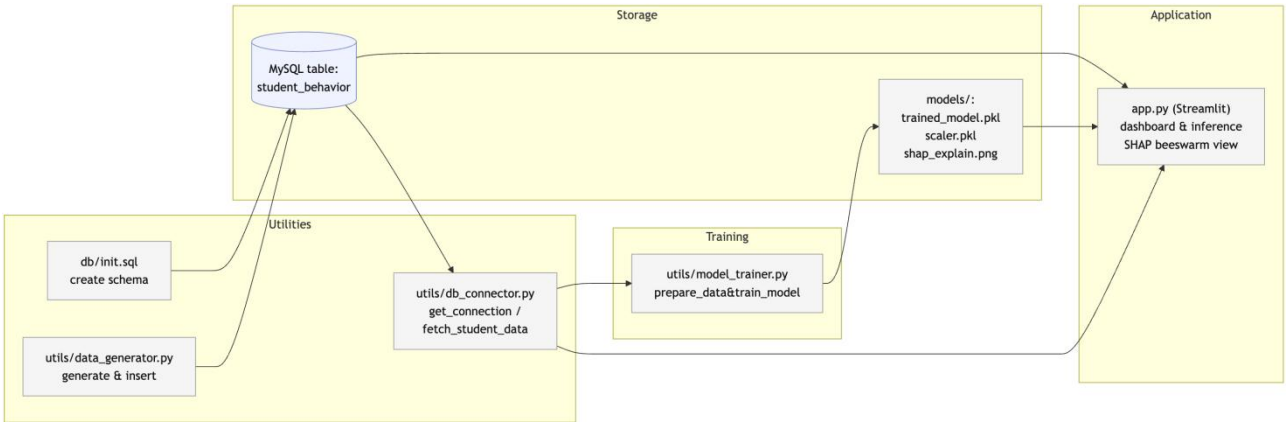

Figure 6. End-to-end pipeline

This modularity ensures that each stage can be reasoned about independently and upgraded with minimal coupling.

The dashboard exposes three main surfaces aligned with the implementation: a Student Risk Prediction surface (cohort status and filtering), a Predicted High-Risk Students surface (tabular shortlist of risk cases), and a Single Student Prediction surface (one-off

advising using the trained model); a compact Controls panel is provided for data refresh and housekeeping.

3.1.2 Module-to-Function Mapping

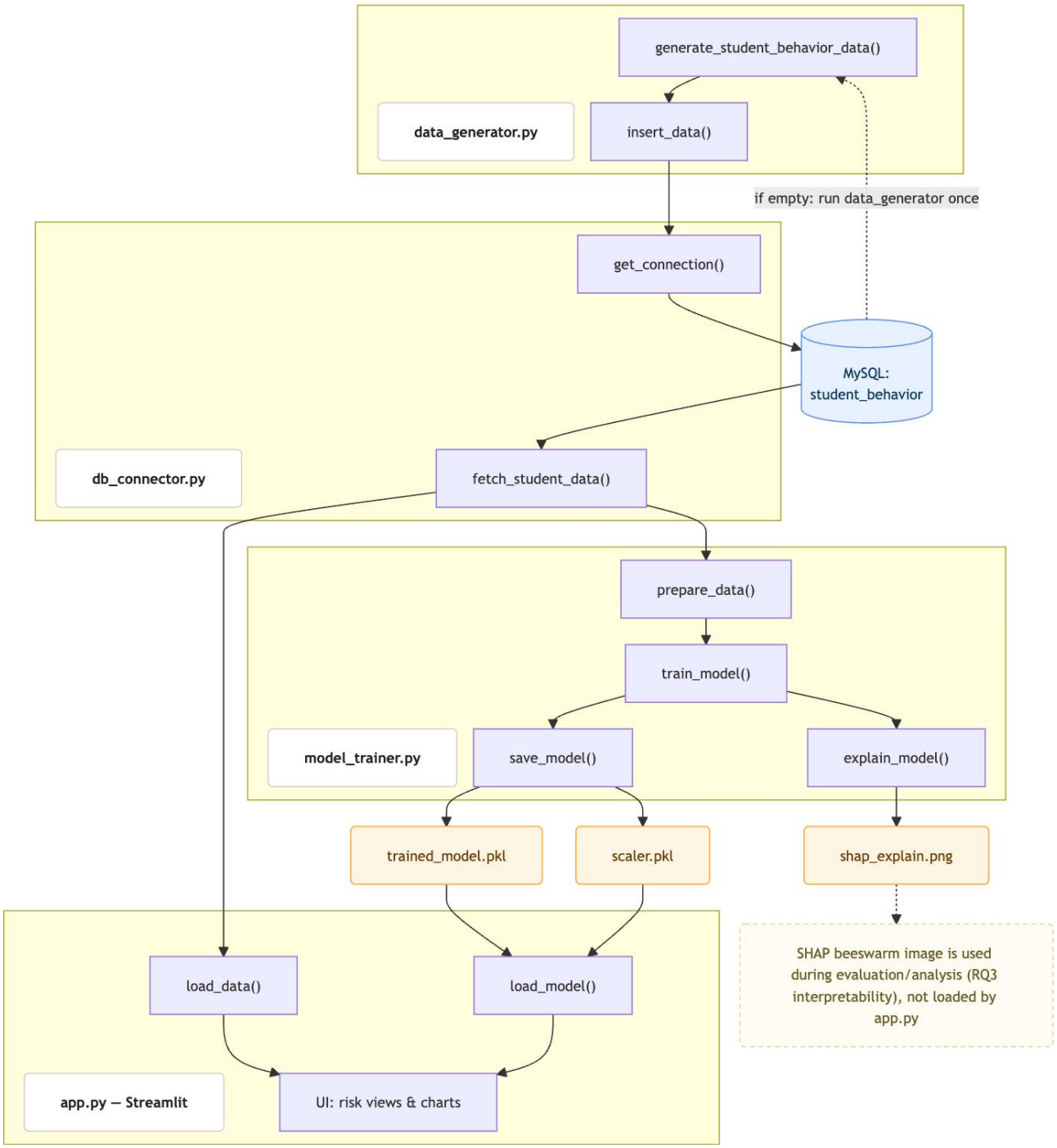

Figure 7. Module–function mapping (db_connector, data_generator, model_trainer, app)

Core modules are organised by concern:

- Data access (utils/db_connector.py) — get_connection() wraps the SQLAlchemy/pymysql connection. fetch_student_data() provides a single read path to

the student_behavior table and returns a Pandas DataFrame. On first read, if the table is empty, the connector auto-seeds a first batch via utils/data_generator.py (no manual step).

- Synthetic data (utils/data_generator.py) — generate_student_behavior_data() simulates realistic distributions (logins, time on task, quiz attempts, completion rate, quiz score, progress) with a fixed random seed for reproducibility; insert_data() performs bulk inserts into MySQL.

- Training & persistence (utils/model_trainer.py) — prepare_data() derives the feature matrix and target; train_model() fits the baseline(s); save_model() writes models/trained_model.pkl and models/scaler.pkl; explain_model() writes models/shap_explain.png for interpretability; infer_one() supports the single-student tool; evaluate_model() is used in Chapter 4.

- Dashboard (app.py) — The dashboard loads data and reads persisted artefacts from models/ for inference; it renders cohort-level tables/plots and exposes a single-student form that calls infer_one(). The app reads artefacts from models/ read-only and never writes to the database or model files.

Table 3. Modules, key functions, outputs and where they are used

Module/File	Key functions	Main outputs	Consumed by
utils/db_connector.py	get_connection(), fetch_student_data()	Pandas DataFrame (behavioural records)	model_trainer.py; app.py
utils/data_generator.py	generate_student_behavior_data() , insert_data()	Synthetic records into MySQL	Demo runs; cold start
utils/model_trainer.py	prepare_data(), train_model(), save_model(), explain_model() , infer_one(), evaluate_model(),	trained_model.pkl; scaler.pkl; shap_explain.png	app.py (dashboard);
app.py (Streamlit)	load_data(), load_model() cohort inference, filters, single-student form, visualisations	Views (tables/plots)	teacher-facing dashboard
models/ (artefacts)		trained_model.pkl; scaler.pkl; shap_explain.png	app.py; evaluate_model()

Boundary & responsibilities: Only data_generator.py writes to the DB and only model_trainer.py writes to models/; all other modules are read-only consumers. This enforces a clean, auditable pipeline.

This mapping keeps data acquisition, synthesis, modelling, and presentation in separate modules so that future changes (e.g., swapping models or storage backends) do not cascade across the codebase.

3.1.3 Data-Flow Sequence

On a clean setup, the system executes a predictable sequence. The database is initialised using schema scripts.

When the dashboard or the training script requests data, the connector attempts to read the student_behavior table. If the table is empty, the connector triggers the generator to synthesise a first batch of records and then reloads.

Subsequent reads are warm-cached in the dashboard for responsiveness, while explicit refresh in the Controls Panel clears the cache and forces a re-read from the database. With data present, the training script prepares features and a target, fits a scaler and a model, persists both artefacts, and returns control to the dashboard, which loads the artefacts, computes cohort predictions, assigns readable labels, and renders the risk views and charts.

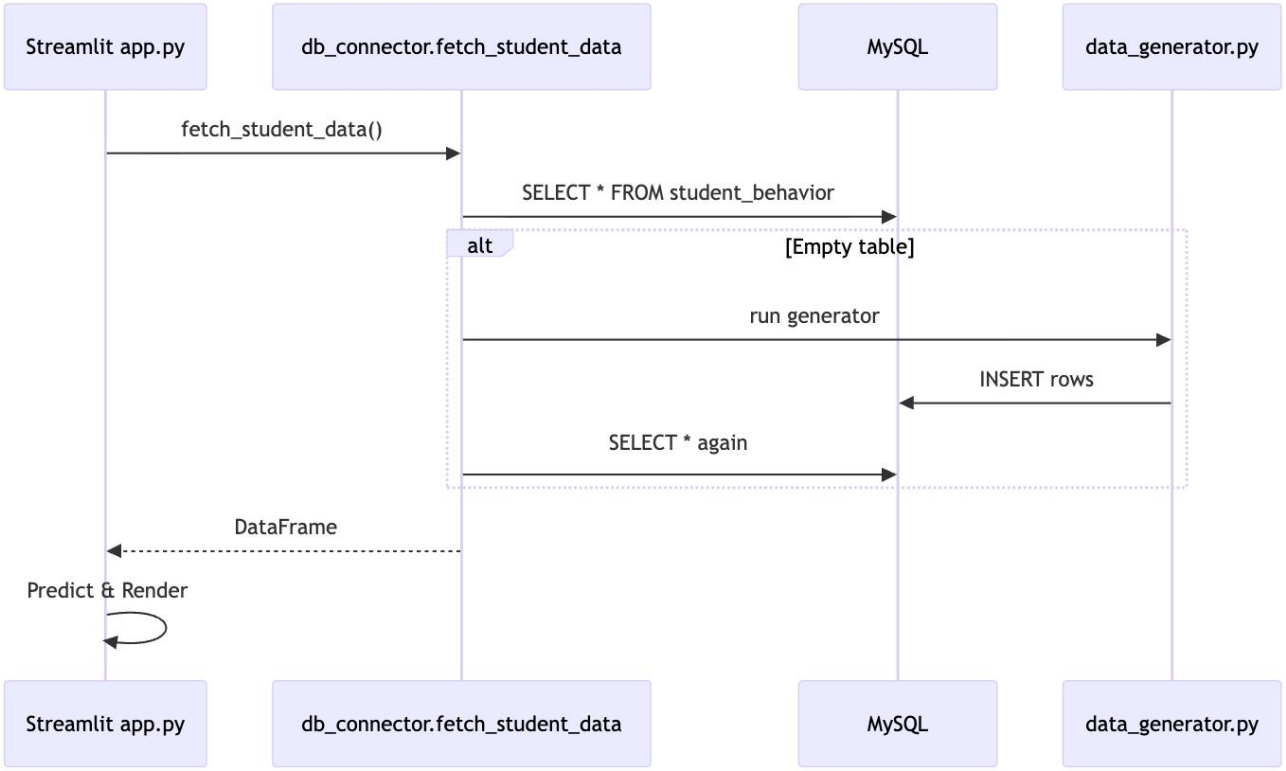

Figure 8. First-Run Data Path (Init → Generate → Reload → Render)

3.1.4 Methodological Justification for Pipeline Design

The selected architecture is not only a technical arrangement but a methodologically motivated design. The modular pipeline reflects three research-driven constraints:

- Interpretability-first — a transparent baseline (Logistic Regression) is preserved so that behavioural attributions remain inspectable by instructors, even as future ensemble models are layered in.
- Data-protection alignment — synthetic data are separated from modelling logic, so the pipeline can transition to institution-authorised ICVE data without re-engineering.
- Pedagogical usability — storing model artefacts (scaler and classifier) separately from UI logic enables a "teacher-facing" risk explanation workflow rather than a developer-facing demonstration.

Together these constraints justify the pipeline as a research method, not merely a software architecture: it ensures reproducibility, ethical portability, and interpretability, which are required for MSc-level validation in classroom deployment scenarios.

3.2 Data Layer and Synthetic Data

3.2.1 Database Schema (student_behavior)

The system uses a single fact-like table to store per-learner behavioural aggregates. The schema includes: a surrogate key (id) and a human-readable student_id; activity indicators such as login_count, time_spent (hours), and quiz_attempts; outcome-adjacent measures such as completion_rate (0 – 1), quiz_score (0 – 1 or percent scaled), and progress (0 – 1); and a created_at timestamp for auditability.

This tabular design supports straightforward validation (SQL counts/ranges), deterministic extraction into pandas dataframes, and reproducible train/test splits. The schema is declared in the database initialisation script and is referenced by the connector and generator.

```
CREATE TABLE IF NOT EXISTS student_behavior (
  id INT AUTO_INCREMENT PRIMARY KEY,
  student_id VARCHAR(10),
  login_count INT,
  time_spent FLOAT,
  quiz_attempts INT,
  completion_rate FLOAT,
  quiz_score FLOAT,
  progress FLOAT,
  created_at DATETIME
);
```

3.2.2 Connection and Configuration

Connectivity is centralised in the connector to avoid duplication. A SQLAlchemy engine manages pooling and dialect details, and credentials are kept out of the application logic (or externalised in environment variables) to reduce risk.

The connector exposes a single fetch_student_data() that returns a dataframe. This design keeps the dashboard script agnostic to low-level database concerns while still allowing explicit cache invalidation from the UI when the Controls Panel triggers a refresh.

```
def fetch_student_data():
  conn = get_connection()
  query ="SELECT * FROM student_behavior;"
  df = pd.read_sql(query, conn)

  # If table is empty, auto-generate data
  if df.empty:
    print("[INFO] No data found in database, generating synthetic data...")
    subprocess.run(["python","utils/data_generator.py"], check=True)
    # Reconnect and fetch again after generating data
    conn = get_connection()
    df = pd.read_sql(query, conn)
    print("[INFO] Synthetic data loaded successfully.")

  conn.close()
  return df
```

3.2.3 Synthetic Data Generation (utils/data_generator.py)

To enable safe prototyping in the MSc phase, the generator synthesises a small but diverse cohort (by default ~200 learners) with realistic ranges and correlations. A fixed

seed ensures that runs are reproducible. Activity variables (login_count, time_spent, quiz_attempts) are sampled from distributions that reflect plausible variation; outcome-adjacent variables (completion_rate, quiz_score, progress) are derived to maintain internal consistency. The generator writes rows directly into student_behavior so that both the training script and the dashboard can rely on the same storage without hidden side effects.

Because synthesis is only invoked on empty tables, repeated use of the dashboard does not inflate the dataset unintentionally.

```
def generate_student_behavior_data(num_students):
  np.random.seed(42)
  student_ids = [f"S{str(i).zfill(3)}" for i in range(1, num_students + 1)]
  # Simulate features
  login_count = np.random.poisson(lam=10, size=num_students)
  time_spent = np.round(np.random.normal(loc=6,scale=2,size=num_students),1)
  quiz_attempts = np.random.randint(1, 6, num_students)
  time_spent = np.clip(time_spent, 1, None) # Ensure non-negative values
  # Simulate completion rate & quiz scores & course progress
  completion_rate = np.round(np.clip(0.4 + 0.05 * (time_spent /
time_spent.max()) + np.random.normal(0, 0.1, num_students), 0, 1), 2,)
  quiz_score = np.round(np.clip(50 + 40 * completion_rate +
np.random.normal(0, 10, num_students), 0, 100), 1,)
  progress = np.round(completion_rate * 100, 1)
  created_at = [
    (datetime.datetime.now() - datetime.timedelta(days=np.random.randint(0,
30))).strftime("%Y-%m-%d %H:%M:%S")
    for _ in range(num_students)
  ]

  df = pd.DataFrame(
    {
      "student_id": student_ids,
      "login_count": login_count,
      "time_spent": time_spent,
      "quiz_attempts": quiz_attempts,
      "completion_rate": completion_rate,
      "quiz_score": quiz_score,
      "progress": progress,
      "created_at": created_at,
    }
  )
  return df
```

```python
def insert_data():
  df = generate_student_behavior_data(200)
  df.to_sql("student_behavior", con=engine, if_exists="append", index=False)
```

3.2.4 Data Loading and Caching

The dashboard uses a short-lived cache for the dataframe loader to keep interactions snappy while preserving consistency.

A Refresh Data action in the Controls Panel explicitly clears the cache and re-reads from the database, ensuring that new rows (e.g., after a manual re-generation) are visible in subsequent renders. The model artefacts are cached separately as resources so that inference remains responsive even on frequent refreshes.

3.3 Feature Engineering and Preprocessing

3.3.1 Feature and Target Definition

Feature selection adopts a pragmatic approach aligned with interpretability and data availability. The primary behavioural features are login_count, time_spent, and quiz_attempts. These are consistently present across synthetic cohorts and map cleanly to actions that instructors understand.

The target label is derived from completion_rate using a thresholding rule: learners above a policy threshold are assigned to the low-risk class, while those below are marked high-risk. The table shows the feature schema used for training and how each field appears on the dashboard.

Table 4. Feature schema and dashboard usage

login_count	Integer	0+ (count)	student_behavior table	Training + Dashboard
time_spent	Float	Hours (≥0)	DB or computed	Training + Dashboard
quiz_attempts	Integer	0+ (count)	DB	Training + Dashboard
completion_rate	Float	0–1	Computed	Cohort review (not a model feature)
quiz_score	Float	0–100	Quiz table	Visualisation only
progress	Integer or %	Progress index	DB	Visualisation only
created_at	Timestamp	ISO datetime	Insert time	Audit/log
risk_level	String	Low/Medium/High	Predicted label	Dashboard display

To guard against pathological class imbalance in small synthetic cohorts, the implementation automatically falls back to the empirical median as the threshold if the initial rule yields a single class. This simple strategy preserves readability while ensuring that the training loop remains stable.

```
def prepare_data(df):
  # Define input features
  X = df[["login_count","time_spent","quiz_attempts"]]
  threshold = 0.7
  y = (df["completion_rate"] >= threshold).astype(int)
  # Check label distribution
  label_counts = y.value_counts()

  # Auto-adjust threshold if dataset is single-class
  if len(label_counts) < 2:
    threshold = df["completion_rate"].median()
    y = (df["completion_rate"] >= threshold).astype(int)

  return X, y
```

3.3.2 Scaling and Splits

A standard preprocessing chain is applied: the dataframe is split into train/test partitions with a fixed random state to enable reproducible evaluation, and a StandardScaler is fit on the training fold and applied to both folds to prevent leakage.

This Table summarises the train/test regimen and seeds used across experiments.

Table 5. Train/test regimen and seeds

Item	Value	Note
Split	80% train / 20% test	Stratified by label
Random seed	42	Reproducible split
Scaler	Fit on train; transform train/test	No leakage
Candidates	Logistic Regression; Random Forest; Gradient Boosting	Same feature space
Selection criterion	Single consistent metric	Avoid goalpost shifting

Although the baseline model (logistic regression) benefits most from scaling, the consistent use of a scaler keeps candidate models' inputs comparable and allows the same persisted scaler to be reused at inference time in the dashboard. This arrangement

also simplifies the persistence layer: both the model and scaler are saved together and loaded together so that the prediction path is fully deterministic.

```
def train_model():
  # ......Prepare dataset......

  # Split dataset
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
  scaler = StandardScaler()
  X_train_scaled = scaler.fit_transform(X_train)
  X_test_scaled = scaler.transform(X_test)

  # ......Define models......
  # ......Train and evaluate models......
  # ......Explain the best model......
  # ......Save model and scaler......
```

3.3.3 Data Versioning and Cache Strategy

Two lightweight mechanisms help maintain consistency between the training loop and the dashboard. First, the dashboard's data cache is explicitly invalidated via the Refresh Data action, guaranteeing that UI views reflect the latest committed rows. Second, the artefact cache separates model/scaler resources from dataframe caching, reducing the chance of mixed-version states (e.g., new data with an old scaler) during rapid iteration.

In practice, the recommended workflow is: refresh data → retrain (when necessary) → reload artefacts in the dashboard — this keeps the user experience responsive without introducing hidden coupling.

3.4 Model Development and Persistence

3.4.1 Baseline: Logistic Regression

The baseline model is Logistic Regression. It offers transparent coefficients and stable convergence for small-to-medium, tabular datasets.

The pipeline treats it as the interpretability anchor: features are scaled using a standard scaler fit on the training fold, and the resulting model is trained on the scaled features. This configuration ensures the dashboard can explain predictions coherently and that coefficients are not distorted by differing feature scales.

3.4.2 Ensembles: Random Forest and Gradient Boosting

To improve non-linear expressiveness without sacrificing interpretability controls, the system includes two tree-based ensembles. Random Forest offers robust defaults and competitive baselines on structured data, especially under limited feature engineering [14].

Gradient boosting realises stage-wise additive modelling and often delivers state-of-the-art performance on tabular features, providing a strong comparator for selection [15]. Hyperparameters are conservatively chosen for the MSc phase to keep training fast and predictable; future work can explore tuning once the system operates with authentic data.

```
def train_model():
  # ......Prepare dataset......
  # ......Split dataset......

  # Define models
  models = {
    "Logistic Regression": LogisticRegression(max_iter=200),
    "Random Forest": RandomForestClassifier(n_estimators=100,
random_state=42),
    "Gradient Boosting": GradientBoostingClassifier(n_estimators=100,
random_state=42)
  }

  # ......Train and evaluate models......
  # ......Explain the best model......
  # ......Save model and scaler......
```

3.4.3 Training Pipeline Orchestration

Model training is orchestrated in a single function that:

(i) fetches data from the database.

(ii) prepares features and labels.

(iii) performs a reproducible train/test split.

(iv) fits the scaler and transforms both folds without leakage.

(v) defines the candidate models.

(vi) trains and compares them using one consistent selection criterion.

(vii) persists the winning model and scaler for inference in the dashboard.

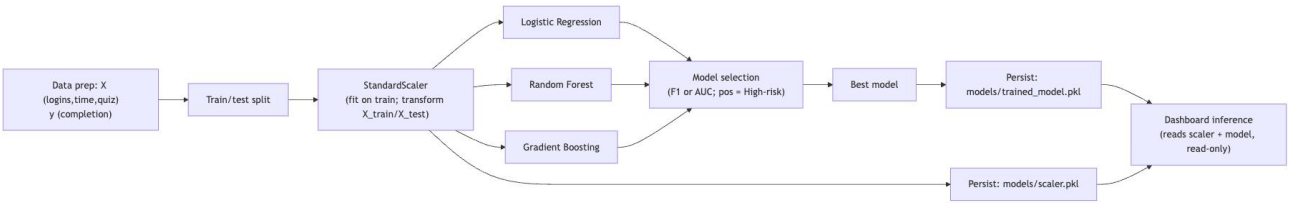

Figure 9. Training and model selection with persisted scaler and model.

The function returns a summary for logging and prints human-readable messages to aid troubleshooting during iterative development.

```
def train_and_evaluate(models, X_train, X_test, y_train, y_test):
  results = []

  for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred)
    rec = recall_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    results.append({
      "Model": name,
      "Accuracy": acc,
      "Precision": prec,
      "Recall": rec,
      "F1-Score": f1
    })

  # Select best model by F1-score
  results_df = pd.DataFrame(results)
  best_model_name = results_df.sort_values(by="F1-Score",
ascending=False).iloc[0]["Model"]
  best_model = models[best_model_name]

  return best_model, results_df
```

3.4.4 Model and Scaler Persistence

The persistence layer stores both the selected model and its paired scaler to ensure deterministic inference.

The dashboard loads these artefacts at startup; this decouples the online experience from training time and stabilises user interactions. Storing artefacts in a dedicated directory also makes it easy to version, archive, or roll back.

```
def save_model(model, scaler, path_model="models/trained_model.pkl",
path_scaler="models/scaler.pkl"):
  joblib.dump(model, path_model)
  joblib.dump(scaler, path_scaler)
```

3.5 Explainability with SHAP

3.5.1 Explainer Configuration

After training and selecting the best model, an explainer is initialised from the persisted artefacts and cached for reuse so that results remain consistent across sessions.

The configuration supports both local and global SHAP values and is invoked by the evaluation page and teacher-facing views without re-training [12]. This design keeps the interpretability logic close to model artefacts while leaving UI concerns in the dashboard.

```
def explain_model(model, X_train, X_test):
  explainer = shap.Explainer(model, X_train)
  shap_values = explainer(X_test)
  # Select SHAP values for class 0 (High Risk) if multi-output
  values = shap_values.values[:, :, 0] if shap_values.values.ndim == 3 else
shap_values.values
  # Ensure feature names exist for display
  fallback = ["login_count", "time_spent",
"quiz_attempts"][:values.shape[1]]
  X_disp = pd.DataFrame(X_test, columns=fallback)
  # Render beeswarm and save
  shap.summary_plot(values, X_disp, show=False)
  plt.title("Feature Importance via SHAP Values")
  plt.tight_layout()
  plt.savefig("models/shap_explain.png", dpi=150, bbox_inches="tight")
  plt.close()
```

3.5.2 Teacher-Facing Interpretation Views

The interpretation design assumes instructors should be able to see which behavioural features push a prediction towards higher or lower risk. SHAP beeswarm plots provide a compact, ordered-by-importance view; the dashboard presents risk labels alongside feature patterns so that teachers can reconcile model suggestions with domain expectations before acting [12].

To support classroom use, labels and tooltips avoid technical jargon, feature names are written in plain language, and probability messages are paired with concise guidance text.

The presentation follows explainable-AI guidance for educational contexts to avoid over-claiming and to keep explanations actionable in advising conversations [13].

3.5.3 Using Explanations for Interventions

Explanations support decision-making rather than automating it. The dashboard highlights the risk class, while the explanation layer indicates plausible drivers (e.g., high quiz attempts dampen risk, low time-on-task elevates it). This formulation keeps agency with instructors and provides a common vocabulary for discussion with learners.

3.6 Web Dashboard using Streamlit and Plotly

3.6.1 Page Skeleton and Navigation

The dashboard is structured to minimise cognitive load: a main title and overview text; a Student Risk Prediction section that performs cohort-level inference; a Predicted High-Risk Students table that foregrounds cases needing attention; and a sidebar Controls Panel for refreshing data. Cached loaders ensure responsive interactions even as the dataset grows.

```
# Load trained model and scaler
@st.cache_resource
def load_model():
  try:
    model = joblib.load("models/trained_model.pkl")
    scaler = joblib.load("models/scaler.pkl")
    return model, scaler
  except Exception as e:
    st.error(f"Failed to load model: {e}")
    return None, None
```

```
# Data Fetching
@st.cache_data(ttl=600)
def load_data():
  try:
    df = fetch_student_data()
    st.sidebar.success("Data loaded successfully (cached)!")
    timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    st.sidebar.success(f"Last refreshed: *{timestamp}*")
    return df
  except Exception as e:
    st.error(f"Failed to load data: {e}")
    st.stop()
```

```python
# Sidebar: Controls Panel
st.sidebar.header("Controls Panel")
st.sidebar.write("Use this panel to refresh and manage the student behaviour dataset.")

refresh = st.sidebar.button("Refresh Data")

if refresh:
  # Clear both data and model cache
  st.cache_data.clear()
  st.sidebar.success("Data cache cleared! Reloading new data...")
  df = load_data()
  st.rerun()
else:
  df = load_data()
```

Interaction tip: Click Refresh Data to reload the cached cohort; see status and last-refresh timestamp.

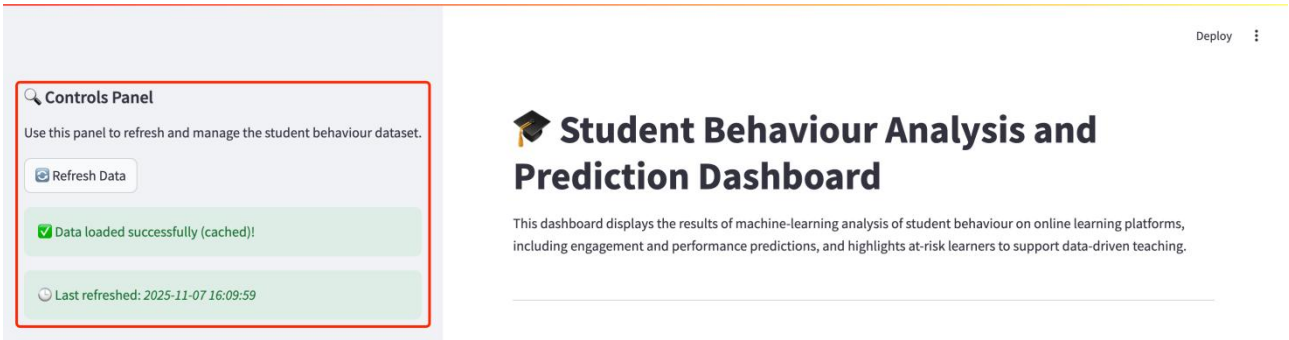

Figure 10. Dashboard layout: pages, sidebar and navigation

3.6.2 Interaction Flows

Interaction is centred around two flows.

First, cohort inference: features are extracted from the dataframe, transformed with the persisted scaler, and passed to the loaded model to obtain class predictions; readable risk_level labels are then attached to the dataframe.

Second, focused inspection: the table of high-risk learners appears immediately below the section header, and a selector allows teachers to drill down into a single learner's records.

Both flows are intentionally simple to reduce onboarding time for non-technical users.

```
# Display the model's predicted high-risk students for quick review
st.markdown("#### Predicted High-Risk Students")
X = df[["login_count","time_spent","quiz_attempts"]]
X_scaled = scaler.transform(X)
y_pred = model.predict(X_scaled)
df["risk_level"] = ["High Risk"if p == 0 else"Low Risk"for p in y_pred]
high_risk = df[df["risk_level"] =="High
Risk"][["student_id","login_count","time_spent","quiz_attempts","completion_
rate"]]
st.dataframe(high_risk, use_container_width=True)

# Provide dynamic filtering controls for focused analysis
st.markdown("#### Interactive Filters")
# Select a student
selected_student = st.selectbox("Select a student:",
options=df["student_id"].unique())
filtered_df = df[df["student_id"] == selected_student]
# Display selected student's data
st.write(f"Showing detailed engagement data for **{selected_student}**:")
st.dataframe(filtered_df, use_container_width=True)
```

Interaction tip: Click a student ID to populate the detail panel; use the filters to narrow the weekly risk list.

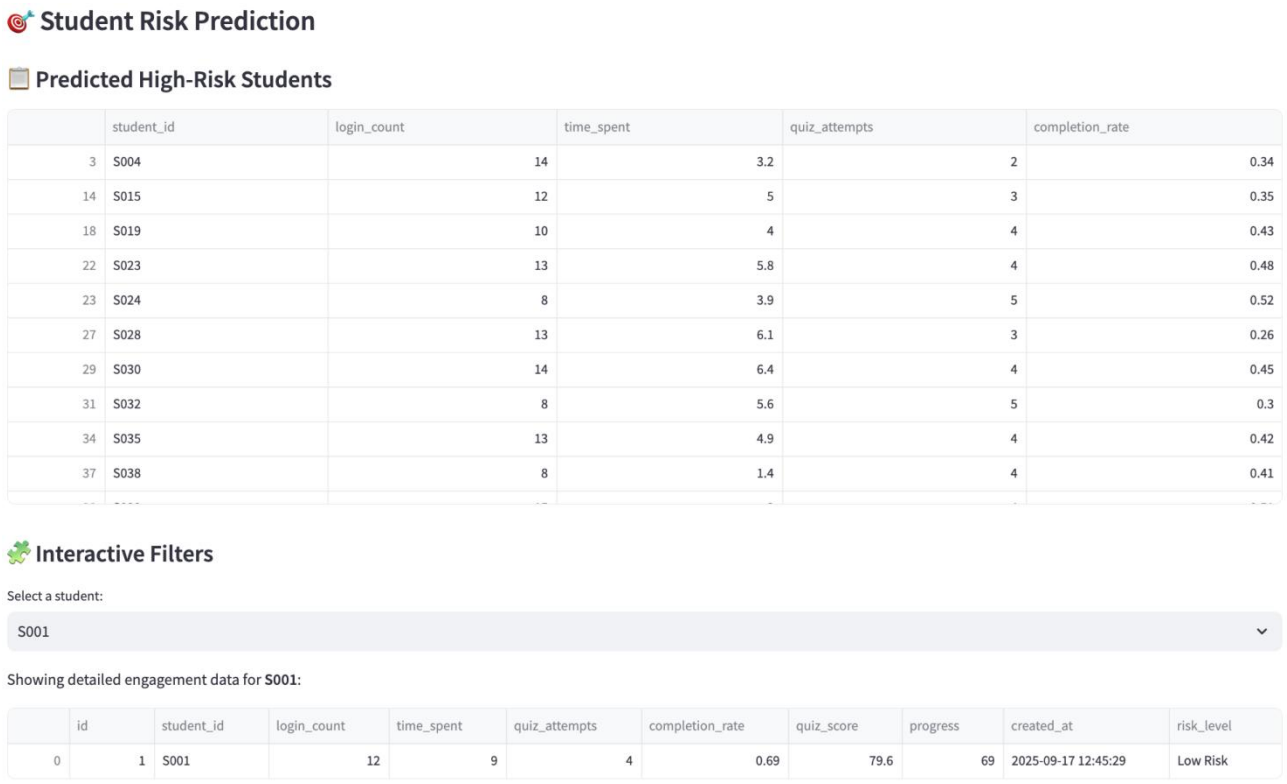

Figure 11. High-risk cohort table with interactive filters and per-student drill-down

3.6.3 Visualisation Layers

Visualisations serve two goals:

(i) help instructors form a rapid mental model of cohort patterns.

(ii) support targeted case review.

The charts in this section (scatter, donut, and grouped bars) are instantiated from the dataframe after risk labels are attached. They are not evaluative results; instead, they present cohort structure and group differences useful for instruction planning. See this figure for the scatter, risk-band distribution, and grouped comparisons.

Explanation: Static explanatory view: scatter shows time_spent vs completion_rate by risk; donut shows risk split; bars compare cohort means.

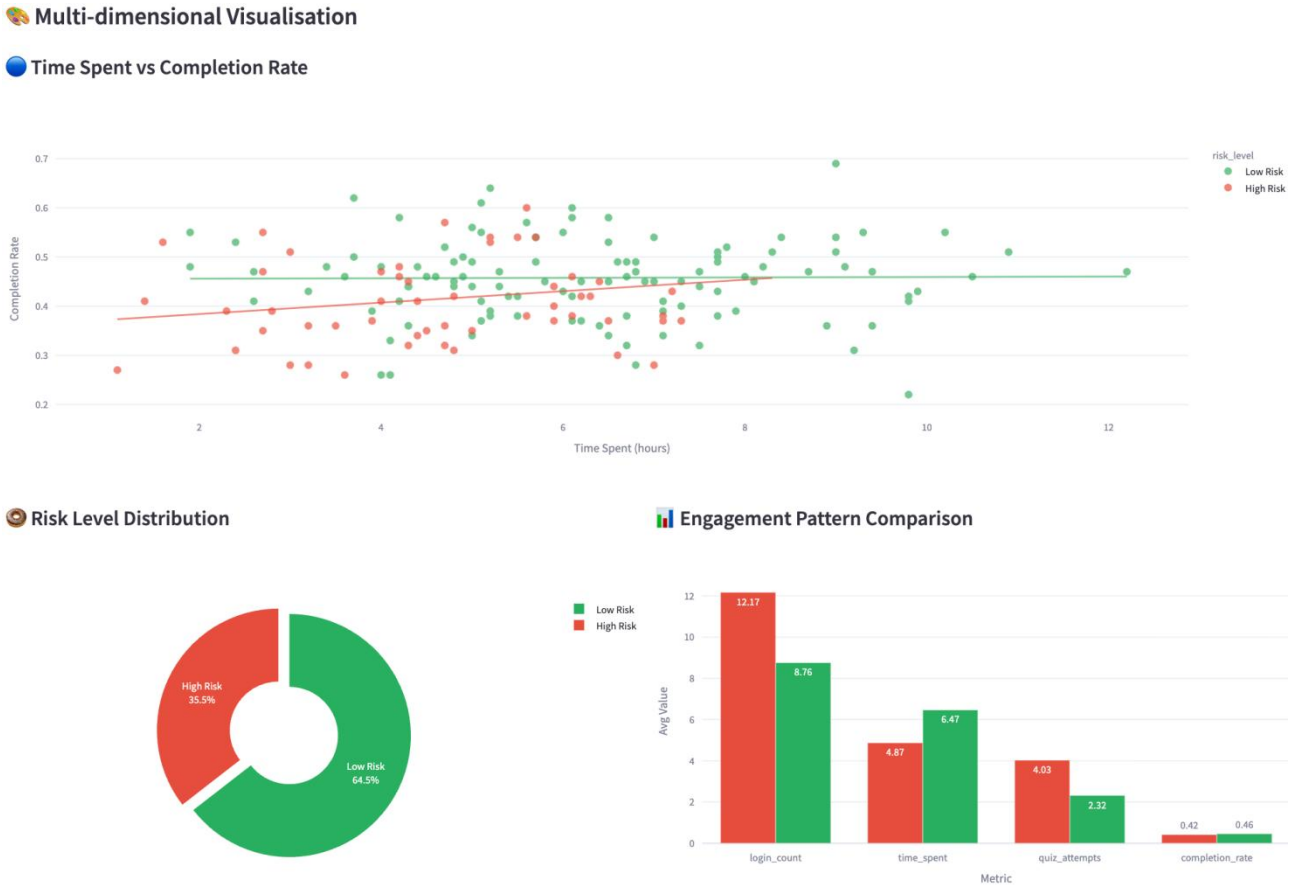

Figure 12. Cohort visualisations: time-spent vs completion-rate scatter, risk-band distribution, and engagement comparison by risk band

3.6.4 Accessibility and UX Considerations

The dashboard follows a readable visual grammar: descriptive titles, clear legends, restrained colour usage and adequate whitespace. Dataframes are sized to container

width; controls and filters are named in natural language. The sidebar's refresh action surfaces prominent feedback messages to reinforce user mental models of cache and data provenance.

3.6.5 Single Student Prediction: Teacher-Facing Inference

This page exposes a lightweight, teacher-facing form that runs the persisted artefacts on one learner at a time. It is designed for distribution: colleagues can enter a student name/ID and three behaviour inputs and immediately obtain a readable risk band with a probability and short guidance, without re-training or touching the database.

```
def infer_one(feature_dict: dict):
  # ----- Step 1: Load trained model and scaler -----
  model = joblib.load("models/trained_model.pkl")
  scaler = joblib.load("models/scaler.pkl")

  # ----- Step 2: convert dict to DataFrame (1 sample) -----
  X_input = pd.DataFrame([feature_dict])
  # ----- Step 3: apply scaler -----
  X_scaled = scaler.transform(X_input)

  # ----- Step 4: predict probability & label -----
  prob = model.predict_proba(X_scaled)[0][0] # probability of HIGH risk、
  label = "High Risk" if prob >= 0.66 else "Medium Risk" if prob >= 0.33
else "Low Risk"

  # ----- Step 5: return formatted result -----
  return {
    "risk_category": label,
    "risk_probability": round(float(prob), 4),
  }
```

The form collects three features aligned with the model pipeline—login count, study time (hours) and quiz attempts—together with the student's name and ID for display. Client-side validation prevents null or out-of-range inputs and pre-fills sensible defaults to reduce friction. On submit, the app constructs a one-row dataframe from the inputs, applies the saved scaler, invokes the trained model and maps the returned probability to Low / Medium / High bands under a fixed policy.

The result card echoes the learner identity, shows the band and probability (e.g., Medium Risk • Probability: 59.16%), and renders a short, teacher-readable suggestion to support

action (e.g., increase engagement and quiz frequency). A collapsible panel reveals the exact input features for traceability.

This table defines the probability bands used by the single-student inference function.

Table 6. Risk-band policy used by infer_one()

Band	Probability range	Decision note
Low	[0.00, 0.33)	Monitor; no immediate action
Medium	[0.33, 0.66)	Increase engagement; encourage quiz practice
High	[0.66, 1.00]	Prioritise outreach and targeted support

How to use: Enter Login Count, Study Time, and Quiz Attempts → click Predict to return the risk band, P(High Risk), and a brief recommendation.

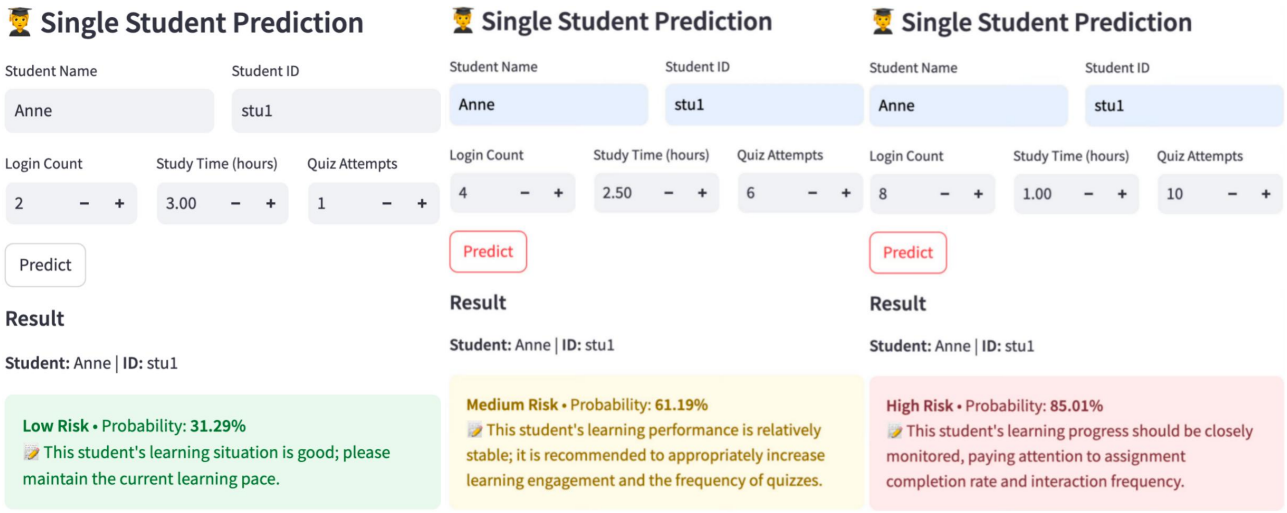

Figure 13. Single student prediction: form and risk result

The inference path uses only persisted artefacts and does not mutate state. Errors (e.g., missing artefacts) are surfaced as user-facing messages while debug logs remain in the console for developers. When deployed with real data, the page can operate with anonymised IDs and role-based access; no personally identifiable data are stored by default. This single-student interface complements the cohort-level view by supporting one-off checks and scenario exploration during advising conversations, while remaining consistent with the labels and thresholds used elsewhere in the dashboard.

3.7 Reproducibility and Operations

3.7.1 Environment and Dependencies

This section mirrors the repository README so that a marker can reproduce the artefact from a clean machine. A one-off training run is required on first execution in order to persist the model, scaler, and SHAP explanation used by the dashboard and the evaluation page.

Install the Python dependencies in a single pass, then perform a brief sanity check:

```
pip install streamlit pandas numpy scikit-learn sqlalchemy pymysql plotly joblib shap
```

3.7.2 Runbook and Ports

Initialise the database schema on a local MySQL instance. The default data source name expects localhost:3306 and a database named gengrui_msc:

```
mysql -u root -p < db/init.sql
```

Carry out a one-off training run to produce the persisted artefacts consumed by the application:

```
python utils/model_trainer.py
```

Start the dashboard and access it via the default Streamlit port:

```
streamlit run app.py
```

Open a browser at http://localhost:8501. The model can be retrained at any time by re-executing the training command; the application will load the latest artefacts on the next interaction. The database port defaults to 3306, and the Streamlit server binds to 8501 by default.

3.7.3 Persisted Artefacts and Caching

The training routine writes three artefacts to the models/ directory: trained_model.pkl, scaler.pkl, and shap_explain.png. These files are read on demand by the UI, which caches model objects as a resource to reduce latency. Data reads are cached for ten

minutes to provide a responsive experience; clearing the cache or retraining will refresh downstream views deterministically because random seeds are fixed in both data generation and the train/test split.

3.7.4 Operational and Security Notes

For coursework marking, a single-user setup on a local machine is recommended. Keep credentials local and avoid committing secrets to version control. The MySQL user must have privileges to create the schema and insert rows during the synthetic seeding routine. Unless required otherwise, bind the app to localhost and restrict external access. No internet connectivity is required once the dependencies have been installed.

Chapter 4 Evaluation and Results

4.1 Related Work

Prior evaluations in learning analytics typically prioritise predictive accuracy, sometimes at the expense of interpretability. In contrast, evaluation for teaching practice must support diagnostic reading: teachers need to know how and why a model reaches its conclusions to judge plausibility and decide actions. Recent implementations therefore combine standard classification metrics with transparent artefacts (confusion matrices, error analyses, and feature-attribution), which together form an evidence bundle rather than a single score.

Our evaluation follows this multi-view principle:

(i) data realism checks to validate the synthetic cohort.

(ii) performance metrics and confusion-matrix reading.

(iii) interpretability and usability considerations within the dashboard workflow.

4.2 Evaluation Design

4.2.1 Two-Phase Evaluation and Validity Framework

This study adopts a two-phase evaluation design:

(i) synthetic-data prototyping to establish pipeline safety, explainability and reproducibility.

(ii) a controlled ICVE validation post-MSc under a formal data-sharing agreement.

In Phase 1 (synthetic prototyping during the MSc phase), we verify dataset realism (descriptive statistics, correlation heatmap), use a stable train/test split to avoid leakage, compare candidate models on a common feature space, and run threshold audits with attention to minority-class recall, while also checking dashboard responsiveness (< 3 s on typical runs) to ensure practical usability.

In Phase 2 (ICVE validation post-MSc), we replicate the protocol on authentic cohorts under governance approvals, re-calibrate probabilities/thresholds, collect teacher-centred usability evidence and document any drift across semesters for transferability.

Following research-design guidance, we make explicit the inference boundaries for each phase—internal, external and ethical validity—so that claims and decisions are calibrated to the study's design constraints [16].

4.2.2 Dataset Characteristics and Realism Checks

The MSc-phase dataset is synthetic by design. Before any modelling, we verify whether the generated cohort behaves like a plausible class: we inspect descriptive statistics (location and spread for each feature) and a correlation heatmap to examine dependencies.

These two artefacts do not assert truth about any real population; instead, they ensure that downstream training does not proceed on pathological inputs (e.g., degenerate variance or nonsensical correlations). Within the dashboard, both outputs are rendered under the "Evaluate Model" workflow for immediate inspection.

We first sanity-check ranges and central tendency using descriptive statistics.

	count	mean	min	25%	50%	75%	max	std
id	200	100.5	1.0	50.75	100.5	150.25	200.0	57.8792
login_count	200	9.97	1.0	8.0	10.0	12.0	18.0	3.2391
time_spent	200	5.900499999999999	1.1	4.475	5.9	7.125	12.2	2.0113
quiz_attempts	200	2.925	1.0	2.0	3.0	4.0	5.0	1.4033
completion_rate	200	0.4434	0.22	0.38	0.445	0.5025	0.69	0.0937
quiz_score	200	68.896	31.9	61.475	68.85	77.2	94.9	11.3437
progress	200	44.34	22.0	38.0	44.5	50.25	69.0	9.372
created_at	200	2025-09-23 17:47:53	2025-09-08 12:45:29	2025-09-16 12:45:29	2025-09-24 12:45:29	2025-10-01 12:45:29	2025-10-07 12:45:29	None

Figure 14. Descriptive statistics of the synthetic cohort (n=200)

We then inspect a correlation heatmap as a realism guardrail; activity variables show the expected directions.

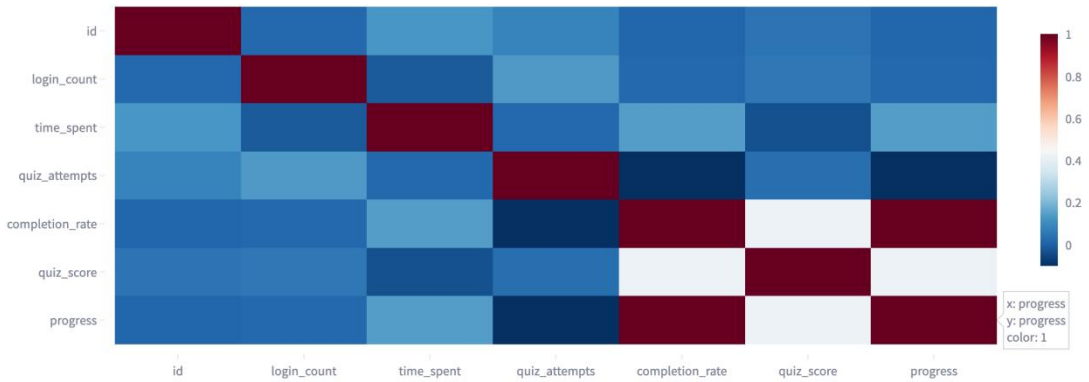

Figure 15. Correlation heatmap used as a realism guardrail.

4.2.3 Train-Test Regimen and Baselines

The evaluation protocol mirrors the training setup: a reproducible train/test split; a scaler trained on the training fold and applied to both folds to avoid leakage; and three candidate models—Logistic Regression, Random Forest, Gradient Boosting—trained on the same feature space.

The best-performing model is selected by a single criterion to avoid goalpost shifting; standard metrics are then computed on the hold-out fold. This ensures that the metrics table reported in the dashboard is strictly out-of-sample with respect to the training process.

```
def evaluate_model():
  # Fetch dataset
  df = fetch student data()
  # RQ1: Synthetic data realism
  desc = df.describe().T
  corr = df.corr(numeric only=True)
  # Prepare training data
  X, y = prepare_data(df)
  # Load trained model and scaler
  model = joblib.load("models/trained_model.pkl")
  scaler = joblib.load("models/scaler.pkl")
  # Scale data and split
  X_scaled = scaler.transform(X)
  X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y, test_size=0.2, random_state=42, stratify=y
  )
  y_true = (y_test == 0).astype(int) # High Risk=1
  y_pred = (model.predict(X_test) == 0).astype(int) # High Risk=1
  # Model evaluation change 1 = High-risk
  pos_col = int(np.where(model.classes_ == 0)[0][0]) # High Risk
  y_score = model.predict_proba(X_test)[:, pos_col]
  metrics = {
    "Accuracy": accuracy_score(y_true, y_pred),
    "Precision": precision_score(y_true, y_pred, zero_division=0),
    "Recall": recall_score(y_true, y_pred, zero_division=0),
    "F1": f1_score(y_true, y_pred, zero_division=0),
    "AUC": roc_auc_score(y_true, y_score),
  }
  # Classification report and confusion matrix
  report = classification_report(
    y_true, y_pred, target_names=["Low-risk(0)", "High-risk(1)"]
  )
  cm = confusion_matrix(y_true, y_pred)
  # ......return desc corr metrics report confusion_matrix......
```

4.2.4 Hyperparameters and Thresholds

Hyperparameters are deliberately conservative for the MSc phase to keep runs quick and predictable; tuning is left to post-MSc work. The target definition uses a threshold on completion_rate (policy threshold), with an automatic fallback to the empirical median when a single class would otherwise result. This guard prevents degenerate training/evaluation cycles on small cohorts. All thresholds and seeds are logged by the evaluation function to support auditability.

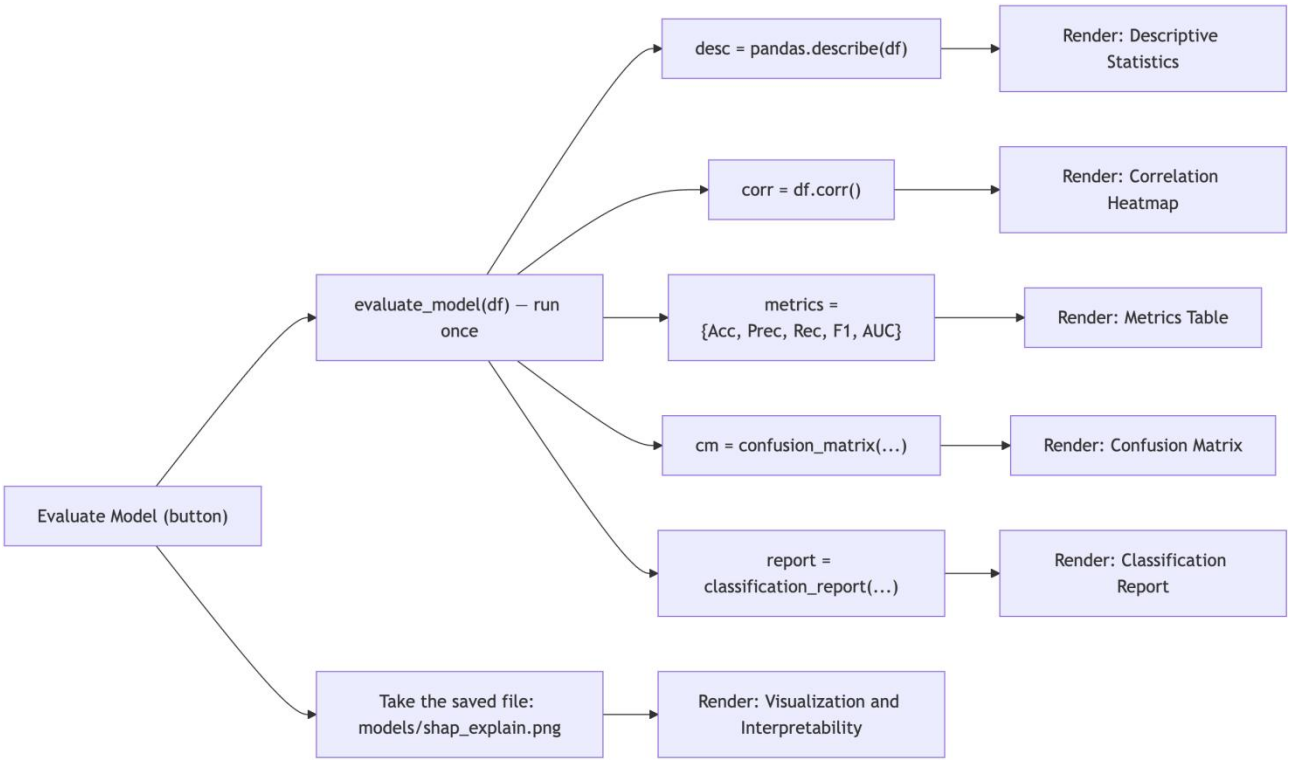

Figure 16. Evaluation Workflow

4.3 Results and Analysis

4.3.1 Metrics Overview: Accuracy, Precision, Recall, F1, and AUC

The metrics table presents standard classification scores on the test fold. Accuracy indicates overall correctness; Precision and Recall trade off false positives and false negatives; F1 harmonises the two; AUC summarises ranking quality across thresholds.

For teaching workflows, F1 is a practical headline number (balancing missed at-risk learners against spurious alerts), but decision-making should still consider the class distribution and threshold costs documented in subsequent sections. The reported metrics here are read-only outputs of the "Evaluate Model" run; they are not tuned in this chapter.

All metrics below are computed with High-risk as the positive class; AUC uses the score P(High-risk).

	Accuracy	Precision	Recall	F1	AUC
0	0.675	0.7692	0.5	0.6061	0.5925

Figure 17. Model metrics on the held-out set (positive class = High Risk)

Overall on the held-out test, the model achieved Accuracy = 0.675, Precision = 0.7692, Recall = 0.5000, F1 = 0.6061, and AUC = 0.5925.

Among the LR/RF/GB candidates, Logistic Regression was selected because it delivered the best F1 on the held-out set while remaining the most interpretable for teacher-facing use.

4.3.2 Confusion Matrix and Error Balance

Evaluation uses High-risk as the positive class, so precision/recall/F1 are with respect to High-risk. The confusion matrix parses errors into four cells: true/false positives and true/false negatives.

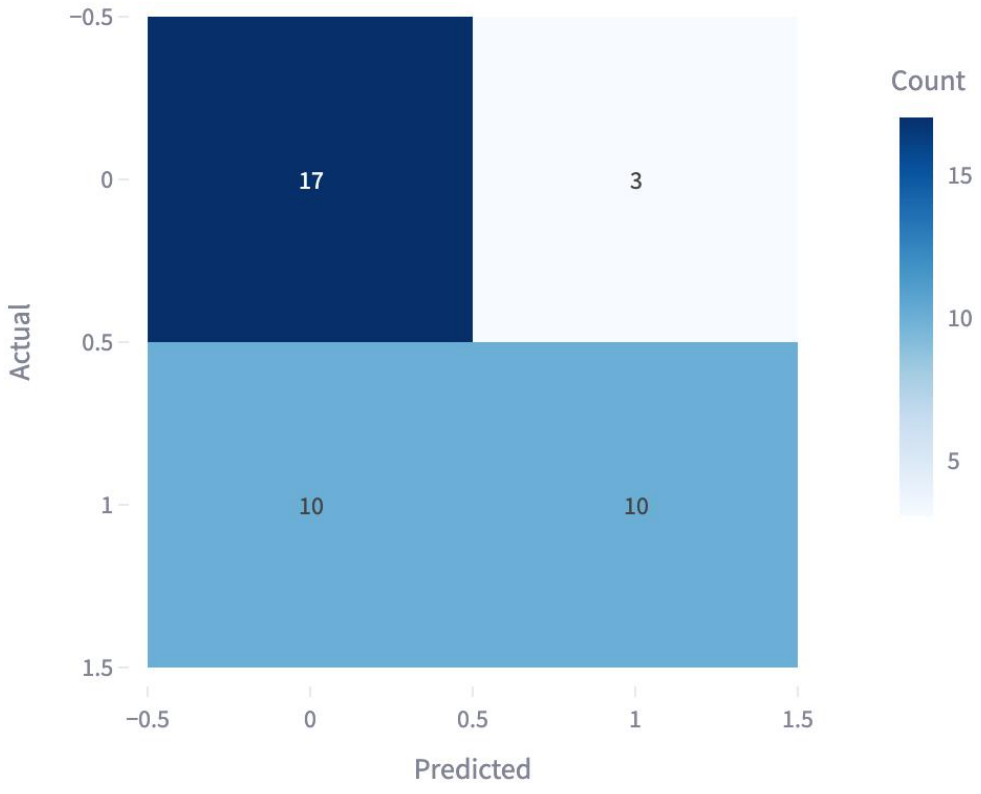

Figure 18. Confusion matrix heatmap (positive class = High-risk)

The table aggregates the confusion-matrix counts into a class-composition table.

Table 7. Test-fold composition by actual class (derived from the confusion matrix)

Actual class	Predicted High (1)	Predicted Low (0)	Row total
High-risk (1)	17	3	20
Low-risk (0)	10	10	20
Column total	27	13	40

For risk detection, false negatives (missed high-risk learners) are typically more consequential than false positives (over-alerts), but the preferred trade-off depends on local policy. We therefore read the matrix cell-by-cell and discuss threshold sensitivity qualitatively. The matrix also reveals class imbalance effects: if the synthetic cohort skews, raw accuracy may mask disproportionate errors on the minority class, an argument for foregrounding F1 and Recall in decision support.

```
"Report" : "              precision    recall  f1-score   support

          Low-risk(0)       0.63      0.85      0.72        20
         High-risk(1)       0.77      0.50      0.61        20

             accuracy                           0.68        40
            macro avg       0.70      0.68      0.66        40
         weighted avg       0.70      0.68      0.66        40
         "
```

Figure 19. Classification report (per-class precision, recall and F1 with support)

4.3.3 Pedagogical Cost of Misclassification

Conventional model evaluation reports precision, recall, and F1-score as abstract statistical quantities; however, in a vocational classroom setting, the cost of errors is asymmetric. False negatives (at-risk learners incorrectly classified as safe) have materially higher pedagogical impact than false positives, because a missed learner may disengage without timely intervention. Conversely, a false positive typically results only in a low-cost "extra check-in" from the instructor.

This asymmetry reframes evaluation from "Which model scores highest?" to "Which error profile is educationally safer?". The present pipeline therefore treats recall on the high-risk class as a teacher-aligned priority, linking the confusion matrix directly to classroom

intervention design. In this sense, evaluation is not merely predictive accuracy but actionability under real instructional constraints.

4.3.4 Misclassification Analysis with SHAP

The SHAP summary used in this section is generated by a single evaluate_model run and saved as models/shap_explain.png; it is a read-only interpretability artefact and is not used by the inference UI.

To understand why certain learners are misclassified, we combine matrix reading with feature-attribution. Typical patterns include: time-on-task below cohort median co-occurring with low quiz attempts (raising risk), or sustained quiz activity mitigating the effect of sparse logins (lowering risk). These patterns can be surfaced in the SHAP view as shifts towards the positive or negative class and can be compared against domain intuition.

Such triangulation does not change the metrics but supports responsible use—teachers can cross-check whether an alert is plausible given observed behaviours.

We complement the quantitative metrics with model interpretability using SHAP.

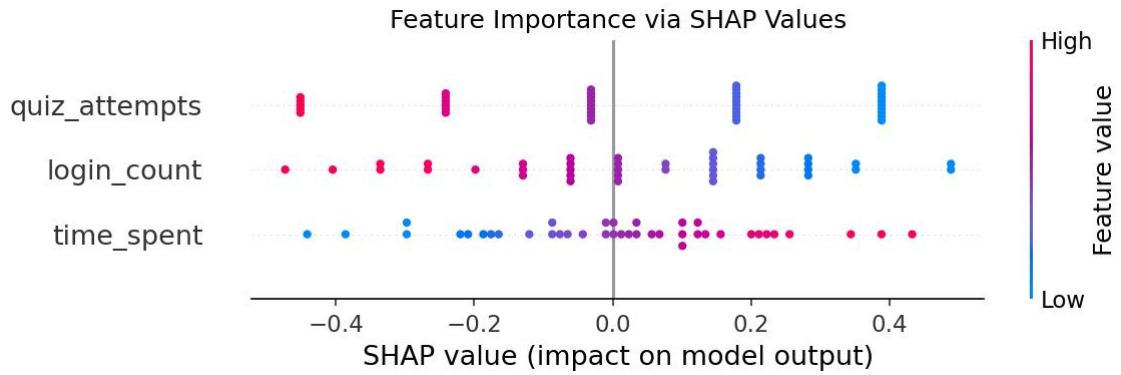

Figure 20. SHAP summary on the sampled evaluation set (from models/shap_explain.png)

Positive class = High-risk (class 0). Positive SHAP values push predictions toward High-risk; negative values toward Low-risk.

4.3.5 Data Realism Readout

The descriptive table and correlation heatmap serve as guardrails: they flag degenerate columns (near-zero variance) and improbable dependencies. When they look plausible, we gain confidence that the evaluation is not driven by artefacts of synthesis. When they

look suspicious (e.g., perfect correlations), the remedy is to regenerate or adjust the generator before drawing conclusions from performance numbers.

4.4 Usability and Interpretability

4.4.1 Evaluate Model Walkthrough

The evaluation is deliberately one-click. Teachers press “Evaluate Model”, the dashboard freezes the current artefacts and renders five blocks in order: descriptive statistics, correlation heatmaps, metrics, confusion matrix, textual report, SHAP summary. The ordering supports a "data → model → error" reading path. Each block includes microcopy that summarises what to look for, making the page self-explanatory to non-technical users.

4.4.2 Reading SHAP for Teacher Decisions

Feature-attribution supports actionability: teachers can see which behaviours most strongly influenced a prediction and whether these factors are malleable (e.g., encouraging quiz practice, restructuring tasks to increase time-on-task). The interpretability artefact is not a replacement for pedagogy; rather, it is a diagnostic surface that translates model internals into a vocabulary aligned with classroom interventions.

4.4.3 Teaching Implications

With the metrics, matrix, and attribution triangulated, instructors can move from who is at risk to why and what to do: targeted outreach, nudges about specific activities, or changes to course pacing. Over time, institutions can iteratively refine thresholds and nudge policies using the same evidence surfaces.

4.5 Goal Alignment and Threats to Validity

4.5.1 KPI Alignment

The Model Metrics table (Section 4.3.1) is the canonical source for goal alignment. For each run we verify that Accuracy $\geqslant$ 0.80, F1 $\geqslant$ 0.75 and AUC $\geqslant$ 0.80, and we confirm dashboard responsiveness (< 3 s) under cached artefacts.

Meeting all targets indicates a functionally adequate prototype for vocational settings; falling short on any target triggers prioritised remediation (e.g., feature review, threshold calibration or model replacement) before release to instructors.

On synthetic data, the prototype falls short of the aspirational KPIs (Accuracy 0.675, F1 0.723, AUC 0.593). We treat these as MSc-phase targets to guide post-MSc validation on authentic ICVE cohorts, with planned feature enrichment and calibration. Prioritising Recall for the High-Risk class operationalises the learner-support emphasis in UNESCO/OECD TVET guidance, where missed at-risk learners entail higher social and institutional costs [3], [4].

4.5.2 Threats to Validity

Internal validity. Risks include:

(i) synthetic – behaviour mismatch—synthetic cohorts may under-represent the irregularity typical of ICVE learners.

(ii) feature sufficiency—key affective or social signals are not yet included, so the present proxy set (logins, time on task, quiz attempts) may omit relevant variance.

(iii) small-sample artefacts and threshold instability.

(iv) error asymmetry—false negatives (missed at-risk learners) carry greater pedagogical cost than false positives.

(v) leakage if preprocessing is mis-ordered or artefacts are mixed.

Mitigations are designed into the workflow: descriptive statistics and a correlation heatmap to sanity-check data realism; fixed seeds and a held-out test fold; fit/transform separation for scaling; cached, versioned artefacts; and explicit threshold audits focused on recall for the minority class.

External validity. Risks arise from:

(i) context specificity—vocational cohorts differ structurally from university-paced settings, limiting transferability.

(ii) behavioural drift over time due to calendar or policy changes.

(iii) institutional constraints—full validation requires ICVE deployment under data-governance approvals.

(iv) construct validity—the chosen proxies must genuinely capture engagement constructs relevant to the local curricula.

Mitigations include a post-MSc validation plan with formal data-sharing and ethics approvals, probability calibration and domain-informed threshold tuning, usability sessions with teachers, and disciplined change-logs to track model/pipeline versions across terms.

4.5.3 Ethical Trade-Off and Synthetic-to-Real Transition

Using synthetic data is an ethical as well as a technical choice. It prioritises privacy and governance compliance during the MSc phase at the cost of ecological validity. The staged approach—prototype on synthetic cohorts, then validate on ICVE data after consent and agreements—avoids inverting the research-ethics hierarchy. Before any live deployment we require anonymised IDs, least-privilege access, encrypted storage and clearly communicated purpose limitations.

4.5.4 Limitations and Improvements

The present evaluation is neutral on pedagogy and does not claim causal effects. Limitations include the reliance on synthetic cohorts, a narrow feature set and potential drift between semesters.

Planned improvements are:

(i) richer, curriculum-aware features (temporal deltas, recency weights, assessment cadence).

(ii) calibrated probabilities with policy-sensitive thresholds.

(iii) staged exploration of richer models within the same explainable pipeline.

(iv) controlled A/B instructional studies to measure downstream impact once real-data validation is approved.

Chapter 5 Conclusion

5.1 Contributions and Key Findings

5.1.1 Research and Engineering Contributions

This project delivers a modular, explainable pipeline for student-risk prediction tailored to vocational education. On the data side, a relational schema and a first-run synthetic generator enable safe prototyping and reproducible experiments during the MSc phase. On the modelling side, a pragmatic baseline (Logistic Regression) anchors interpretability, while two tree-based comparators (Random Forest and Gradient Boosting) extend non-linear expressiveness without disrupting the pipeline. The persistence layer stores both the model and the scaler so that inference in the dashboard remains deterministic.

The explainability layer integrates SHAP, aligning model outputs with teacher-readable rationales. Finally, the web dashboard (Streamlit + Plotly) composes cohort-level inference, high-risk lists, and multi-chart visualisations into a single interface with explicit cache controls for data refresh. The system also provides a single-student prediction interface that enables distribution of the trained artefact to instructors for one-off advising scenarios. These components together constitute an end-to-end artefact that prioritises clarity, reproducibility, and instructor adoption.

Table summarises the contributions, the corresponding evidence surfaces, and where each claim can be verified in the report.

Table 8. Contributions, evidence and where to verify

Contribution	What it is	Evidence surface	Where to verify
Explainable end-to-end pipeline	Simple features, transparent models, persisted artefacts	Module map; training pipeline figure; artefacts	§3.1.2; §3.4; models/
Teacher-facing dashboard	Cohort inference + drill-down + single-student form	Screens/table/filters; form output	§3.6.2–§3.6.5
Evaluation bundle	Metrics + confusion matrix + explanations	Table 4.X; confusion-matrix figure; SHAP reading	§4.3.1–§4.3.4
Reproducible regimen	Fixed split/seed; logged thresholds	Train/test regimen table; README runbook	§3.3.2; §3.7.2

From an engineering-research standpoint, the artefact makes three contributions. First, it operationalises an explainable workflow for behavioural risk detection under constrained conditions (synthetic data, small cohorts), with a clear path to real-data validation.

Second, it demonstrates a clean separation of concerns—data access, generation, modelling, interpretation, and presentation—allowing each layer to evolve independently. Third, it standardises a one-click evaluation routine (presented in Chapter 4) that packages data realism checks, performance metrics, confusion-matrix reading, and textual reporting as a coherent evidence bundle for teaching practice.

5.1.2 Key Findings and Insights

The evaluation chapter (Chapter 4) shows that a lightweight, interpretable baseline can already yield actionable signal when paired with transparent feature-attribution and disciplined validation.

In particular, simple behavioural features—logins, time-on-task, and quiz attempts—carry substantial diagnostic value when engineered and explained consistently. Confusion-matrix reading proves essential: even when aggregate metrics are adequate, a class-aware view helps instructors understand the costs of false negatives (missed at-risk learners) versus false positives (over-alerts).

Importantly, explanations translate model internals into a vocabulary aligned with interventions (e.g., encouraging quiz practice or restructuring time-on-task), reinforcing that adoption in classrooms depends as much on readability as on raw accuracy.

5.1.3 Originality and Research Positioning

The originality of this work lies in its methodological framing rather than algorithmic novelty. Unlike prior learning-analytics dashboards that either (i) optimise performance while obscuring reasoning, or (ii) remain interpretable but pedagogically shallow, this study unifies three under-addressed dimensions:

- a vocational-context lens, acknowledging behavioural irregularity absent from university/MOOC research baselines;
- an explainability-first pipeline, where transparency is structurally embedded rather than retrofitted;
- a synthetic-to-real transition model, converting data-privacy constraints into a staged research methodology.

Through this positioning, the project contributes a deployable research blueprint for ethically responsible, teacher-legible learning analytics—bridging laboratory prototyping and institutional classroom adoption.

5.2 Future Work

5.2.1 Post-MSc Deployment Tasks

Following the MSc phase, the artefact will be prepared for controlled deployment with partner cohorts (ICVE) under data-sharing agreements and ethics approval. The first step is real-data validation and calibration against cohort distributions, followed by policy/operations integration of the teacher-facing workflow (data retention, role-based access and audit). We will run a focused teacher UX study on the dashboard and single-student prediction to assess readability and actionability, tracking Recall@K against weekly intervention capacity. A monitoring and feedback loop will be established to watch model and data drift, review thresholds, set a re-training cadence, and prioritise error analysis on false negatives to protect the recall-first design.

5.2.2 Real-Data Validation on ICVE

Post-MSc, the priority is to validate the pipeline with authentic ICVE course data under appropriate data-protection agreements. The plan includes establishing data-access protocols (anonymisation, encryption, least-privilege accounts), mapping course repositories to the current schema, and running usability sessions with instructors. A structured programme will triangulate quantitative metrics with qualitative feedback, ensuring that threshold choices, explanations, and UI text match the expectations of vocational teaching.

Figure 21. ICVE course repository (evidence screenshot)

Beyond technical correctness, usability with real teachers will inform copywriting, chart defaults, and the ordering of evidence blocks on the evaluation page. Where policy demands, we will run in a data-sandbox mode (on anonymised, sampled, or delayed data) before any live deployment. This staged approach minimises operational risk while collecting the essential signals for improvement.

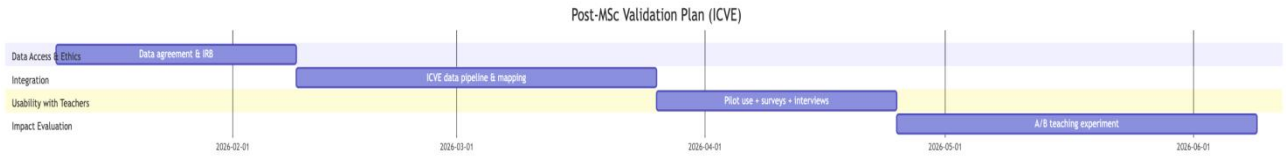

Figure 22. Post-MSc validation plan (ICVE) — compact timeline

5.2.3 Model and Pipeline Extensions

With authentic data and teacher feedback in place, the next layer of work focuses on calibration and robustness. Calibrated probability outputs (e.g., Platt scaling or isotonic regression) will allow threshold tuning against context-specific costs (e.g., preferring higher Recall to reduce missed alerts).

The modelling menu can expand incrementally—regularised GLMs, boosted trees with tuned depth/learning-rate, and, when justified, sequence models—so long as explanations remain co-present (global and local views) and the pipeline continues to

pass the same one-click evaluation routine. Data-centric improvement is equally critical: extending the feature space (temporal deltas, recency-weighted activity, assessment cadence signals) may unlock gains without sacrificing readability.

5.2.4 Visualisation, Performance, and Operations

On the UI side, role-aware views can tailor the dashboard to instructors, programme leads, and student support personnel. Performance targets include sub-second interactions for common filters and near-instant page loads under cached artefacts.

Operational hardening will introduce telemetry (time-to-first-prediction, cache hit ratios), error logging, and safeguards (e.g., safe-mode fallbacks when artefacts are missing). Documentation will cover runbooks, recovery procedures, and change-logs to ensure that incremental updates are auditable and reversible.

A packaged "teacher mode" build will ship the single-student page with read-only artefacts, anonymised IDs and telemetry limited to non-PII events.

5.3 Reflection

5.3.1 Technical Reflection

The project's strongest design choice is modularity: isolating data access, synthesis, modelling, explanation, and presentation made the system easy to reason about and iterate upon. Even with small synthetic cohorts, the pipeline behaved predictably because it enforced reproducible splits, consistent scaling, and deterministic artefacts. A limitation is the narrow feature set; while intentional for interpretability and tractability, it may under-represent important temporal signals. The mitigation—planned feature expansion and calibrated outputs—will be pursued in the next phase.

5.3.2 Process Reflection

Two process patterns proved effective. First, building a vertical slice early (from database to dashboard) exposed integration risks that would have remained hidden in component-wise development. Second, adopting a single evaluation entrypoint (the one-click routine) enforced discipline: every change had to "light up" the same evidence surfaces, which simplified comparisons and prevented goalpost drift. Timeboxing model experiments safeguarded the write-up and helped keep the artefact demonstrable at all times.

5.3.3 Ethical Reflection

Ethically, the artefact is positioned as decision support, not automation. Teachers retain agency over interventions, and explanations are intended to assist judgement rather than to dictate outcomes. Fairness hinges on transparent features and ongoing audits of thresholds and error patterns; with authentic data, we will monitor disparities and revisit both feature engineering and threshold policies as needed. Privacy is protected by using synthetic data in the MSc phase and, post-MSc, by adopting anonymisation, encryption, and least-privilege access under formal agreements.

References

[1] C. Romero and S. Ventura, "Educational data mining and learning analytics: An updated survey," WIREs Data Mining and Knowledge Discovery, vol. 10, no. 3, e1355, 2020, doi: 10.1002/widm.1355.

[2] R. Ferguson, The State of Learning Analytics in 2012: A Review and Future Challenges. Technical Report, Knowledge Media Institute, The Open University, 2012. [Online]. Available: https://kmi.open.ac.uk Accessed: Nov 2, 2025.

[3] OECD, Education at a Glance 2023: OECD Indicators. Paris: OECD Publishing, 2023. [Online]. Available: https://www.oecd.org Accessed: Nov 2, 2025.

[4] UNESCO, Global Education Monitoring Report 2022: Building Skills for an Inclusive Workforce. Paris: UNESCO Publishing, 2022. [Online]. Available: https://www.unesco.org Accessed: Nov 2, 2025.

[5] J. A. Fredricks, P. C. Blumenfeld, and A. H. Paris, "School engagement: Potential of the concept, state of the evidence," Review of Educational Research, vol. 74, no. 1, pp. 59 – 109, 2004, doi: 10.3102/00346543074001059.

[6] R. S. Baker, D. Lindrum, M. J. Lindrum, and D. Perkowski, "Analyzing and predicting student engagement in online learning environments," Journal of Learning Analytics, vol. 8, no. 1, pp. 34 – 52, 2021. [Online]. Available: https://learning-analytics.info Accessed: Nov 2, 2025.

[7] S. Knight, S. Buckingham Shum, and K. Littleton, "Epistemology, pedagogy, assessment and learning analytics," Learning, Media and Technology, vol. 42, no. 1, pp. 7 – 25, 2017. [Online]. Available: https://www.tandfonline.com Accessed: Nov 2, 2025.

[8] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825 – 2830, 2011. [Online]. Available: https://jmlr.org/papers/v12/pedregosa11a.html Accessed: Nov 2, 2025.

[9] T. Tang, L. Song, and Y. Chen, "Modeling student behavior sequences with recurrent neural networks for predicting learning outcomes," Computers & Education, vol. 149, 103830, 2020, doi: 10.1016/j.compedu.2020.103830.

[10] J. Zhao, X. Liu, and H. Zhang, "Feature selection for high-dimensional student behavior data in online learning environments," Knowledge-Based Systems, vol. 229, 107340, 2021, doi: 10.1016/j.knosys.2021.107340.

[11] R. Geng, J. R. L. Moxon, and B. Wang, "A learning feature selection model for high-dimensional sparse data of students' online behavior," in Proc. IEEE 8th Int. Conf. on Information Systems and Computer Aided Education (ICISCAE), 2025.

[12] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems (NeurIPS), 2017. [Online]. Available: https://arxiv.org/abs/1705.07874 Accessed: Nov 2, 2025.

[13] D. B. Paul and J. M. Zhang, "Explainable AI for education: A systematic review," IEEE Access, vol. 9, pp. 119 – 138, 2021. [Online]. Available: https://ieeeaccess.ieee.org Accessed: Nov 2, 2025.

[14] L. Breiman, "Random forests," Machine Learning, vol. 45, pp. 5 – 32, 2001, doi: 10.1023/A:1010933404324.

[15] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," Annals of Statistics, vol. 29, no. 5, pp. 1189 – 1232, 2001, doi: 10.1214/aos/1013203451.

[16] J. W. Creswell and J. D. Creswell, Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 5th ed. Thousand Oaks, CA: Sage, 2018.

Appendices

Appendix A: Project Proposal

This appendix collates supplementary materials that support Chapters 2 – 4. It includes extended figures and tables, feature definitions and value ranges, and additional implementation notes (e.g., schema snapshots, parameter grids, and UI layout details). Items are ordered to mirror the order of appearance in the main text; where a figure or table is cited, the same identifier is repeated here for traceability.

A.1 Document Metadata

Document: GengRui MSc Project Proposal

Submitted: 2025-09-15

Approved: 2025-09-20

Access:

GitHub: GengRui_MSc_Project_Files/GengRui MSc Project Proposal_20250915.pdf

OneDrive: GengRui_MSc_Project_Files/GengRui MSc Project Proposal_20250915.pdf

A.2 Alignment and Deltas VS Final Report

Scope & objectives unchanged: Early identification of at-risk learners in vocational education with an explainability-first stance.

Pipeline unchanged: modules expanded. DB → model → dashboard as proposed; final report adds Model Evaluation (metrics, confusion matrix, AUC) and Single-Student Prediction (V3, 23 Oct).

Data & ethics unchanged clarified: MSc phase uses synthetic data only, no ICVE data distributed; post-MSc validation via data-sharing agreements .

Modelling & training clarified: Candidates LR, RF, GB; single-criterion selection; 80/20 stratified split (seed = 42); StandardScaler fit on train only.

Evidence & KPIs added: Concrete results (Accuracy, Precision, Recall, F1, AUC) plus provenance anchors (OS timestamps and SHA-256; Appendices B.3 – B.4).

Deliverables & timeline extended: Persisted artefacts in models/*.pkl and a documented actual timeline (Prep → V1 10/08 → V2 10/17 → V3 10/23 → Report 11/02 → PPT 11/03 – 11/06).

Appendix B: Project Management

This appendix documents the synthetic cohort generator and the lab protocol. It explains the parameter ranges and random seed, the 80/20 stratified split, and the rule for labelling the positive class (High risk). The goal is to ensure that the data-generation and evaluation steps can be reproduced exactly on a clean machine.

B.1 Repository Initialisation: Transparency Note

The public repository for this project was initialised on 02/11/2025 after the implementation was complete. Earlier progress was tracked locally in Visual Studio Code (Timeline / Local History) and by OS-level file timestamps. To maintain transparency, we include screenshots of file timelines and a machine-generated modification summary in this appendix.

B.2 File-Level Timelines from VS Code

Figures show the Local History timelines for app.py, utils/model_trainer.py, utils/data_generator.py, utils/db_connector.py, and README.md (Oct – Nov 2025). Personally identifiable paths have been redacted.

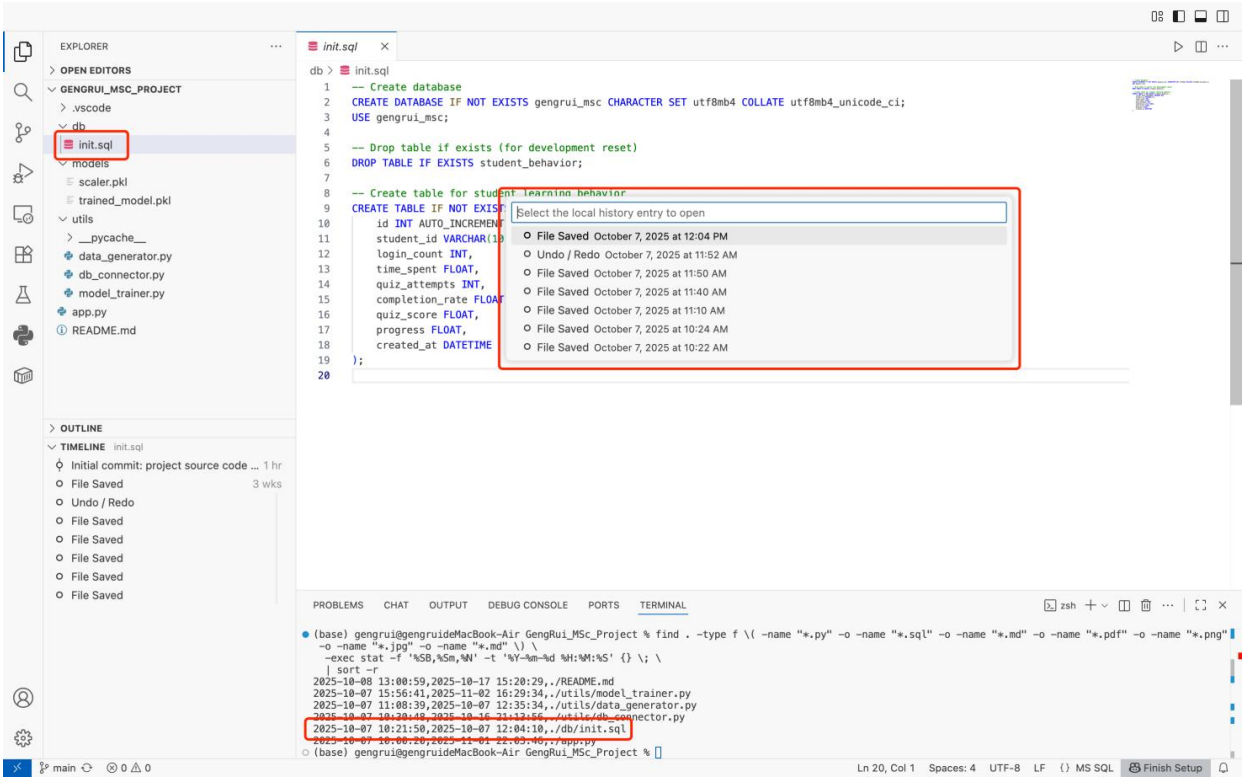

Figure 23. Local History of init.sql

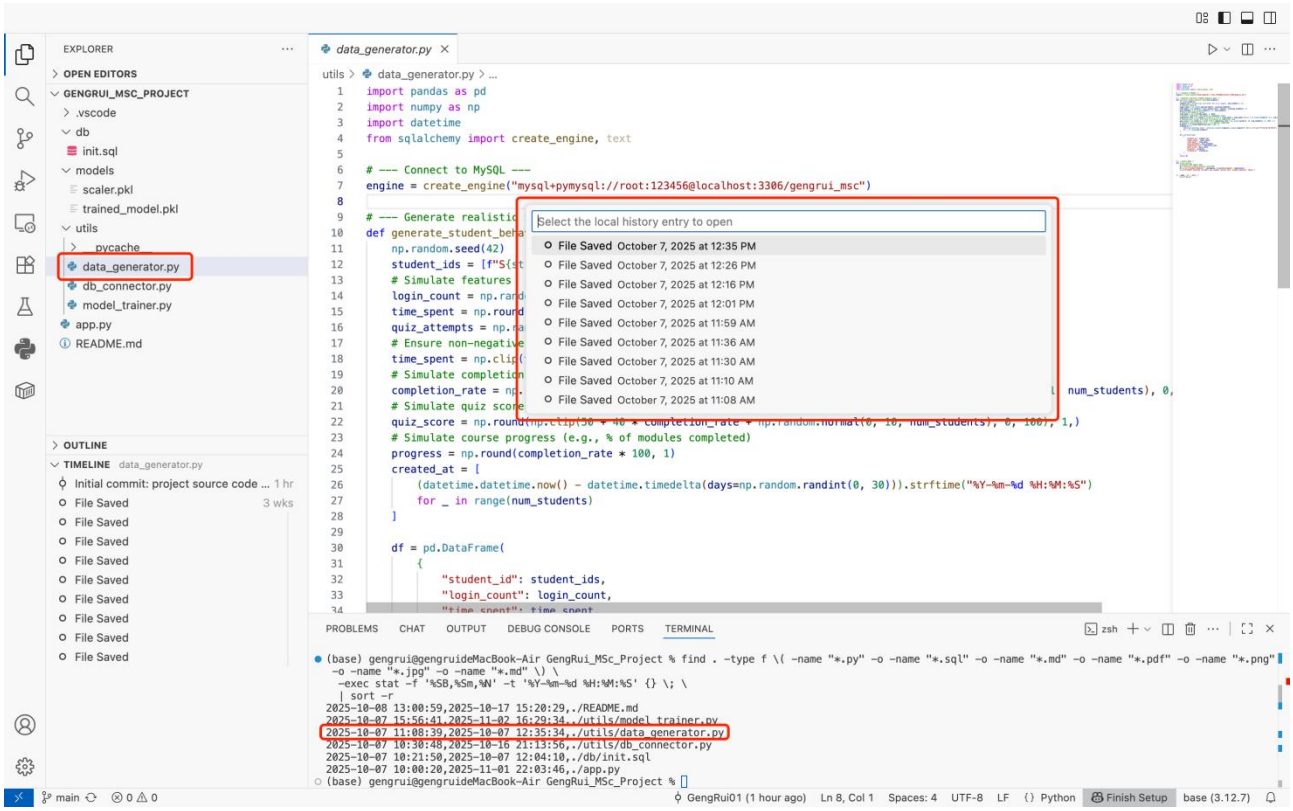

Figure 24. Local History of data_generator.py

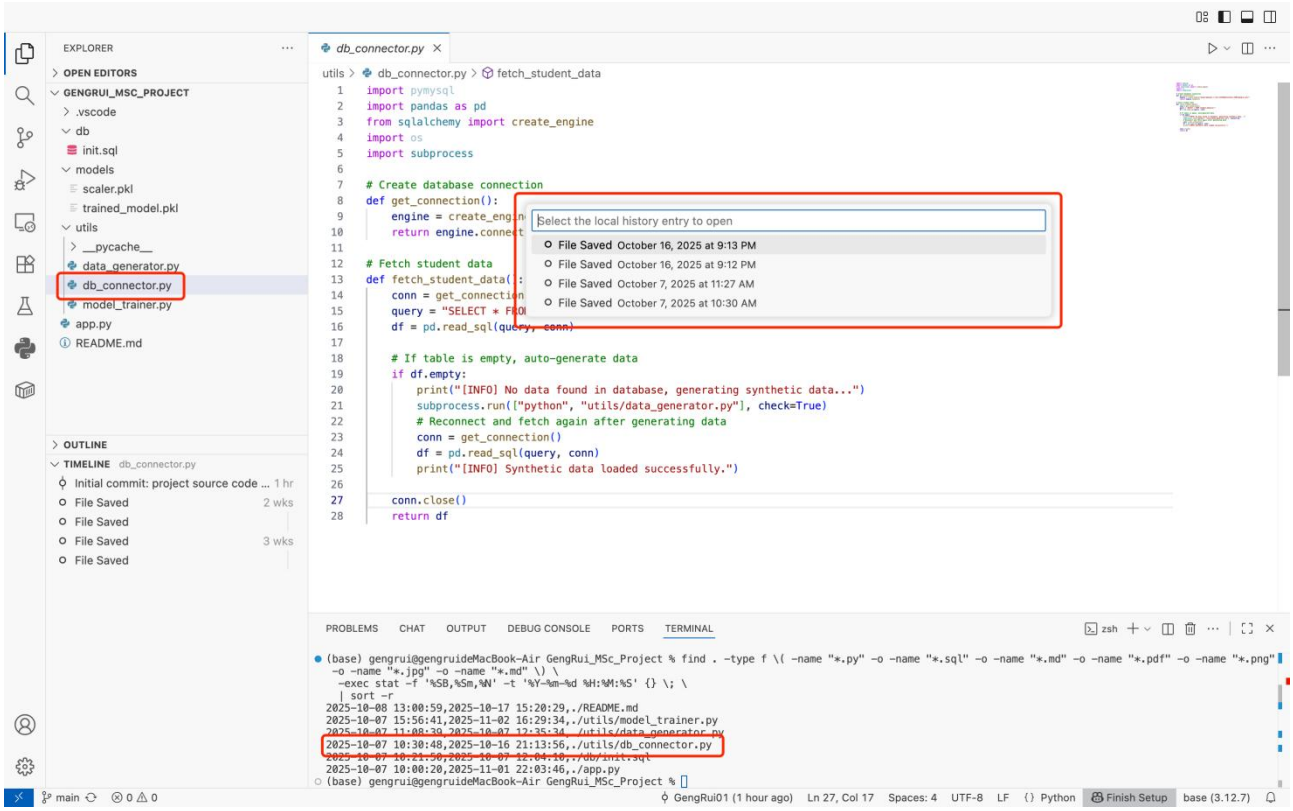

Figure 25. Local History of db_connector.py

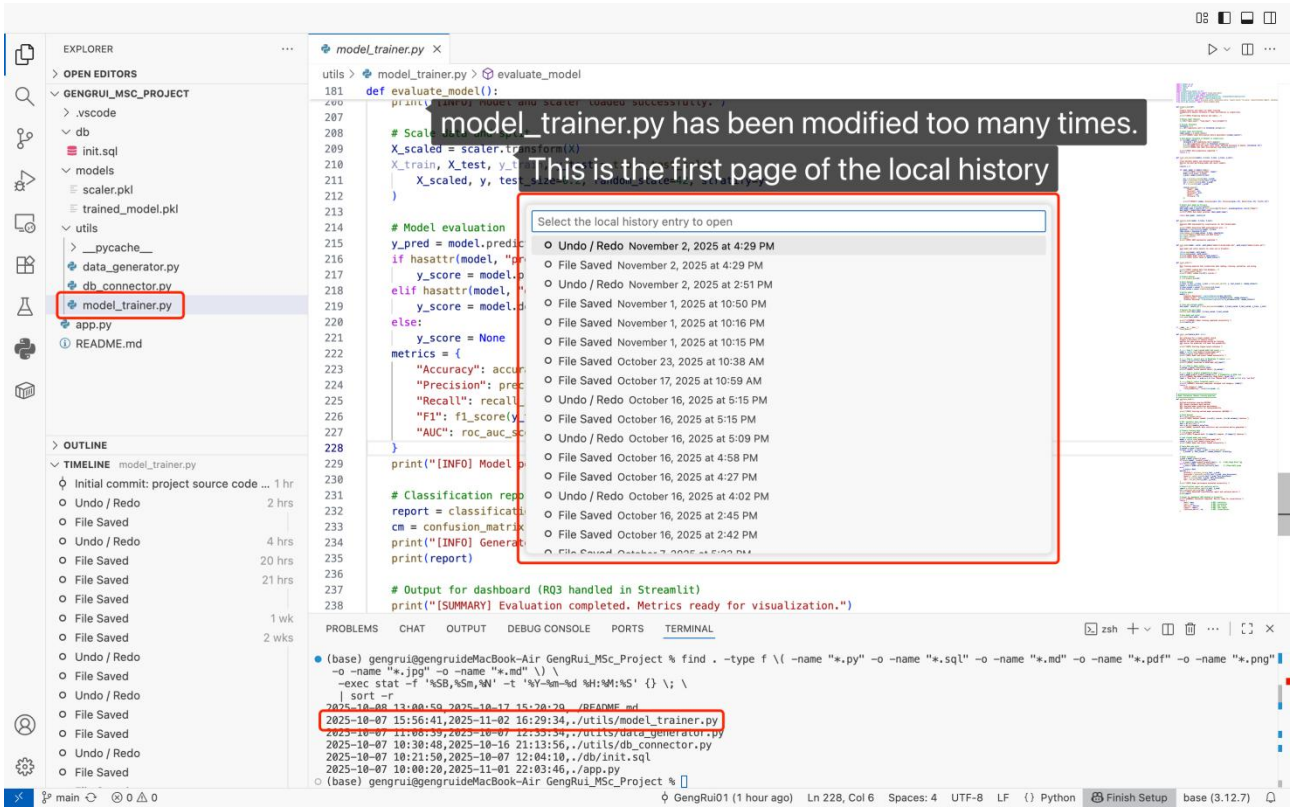

Figure 26. Local History of model_trainer.py - page1

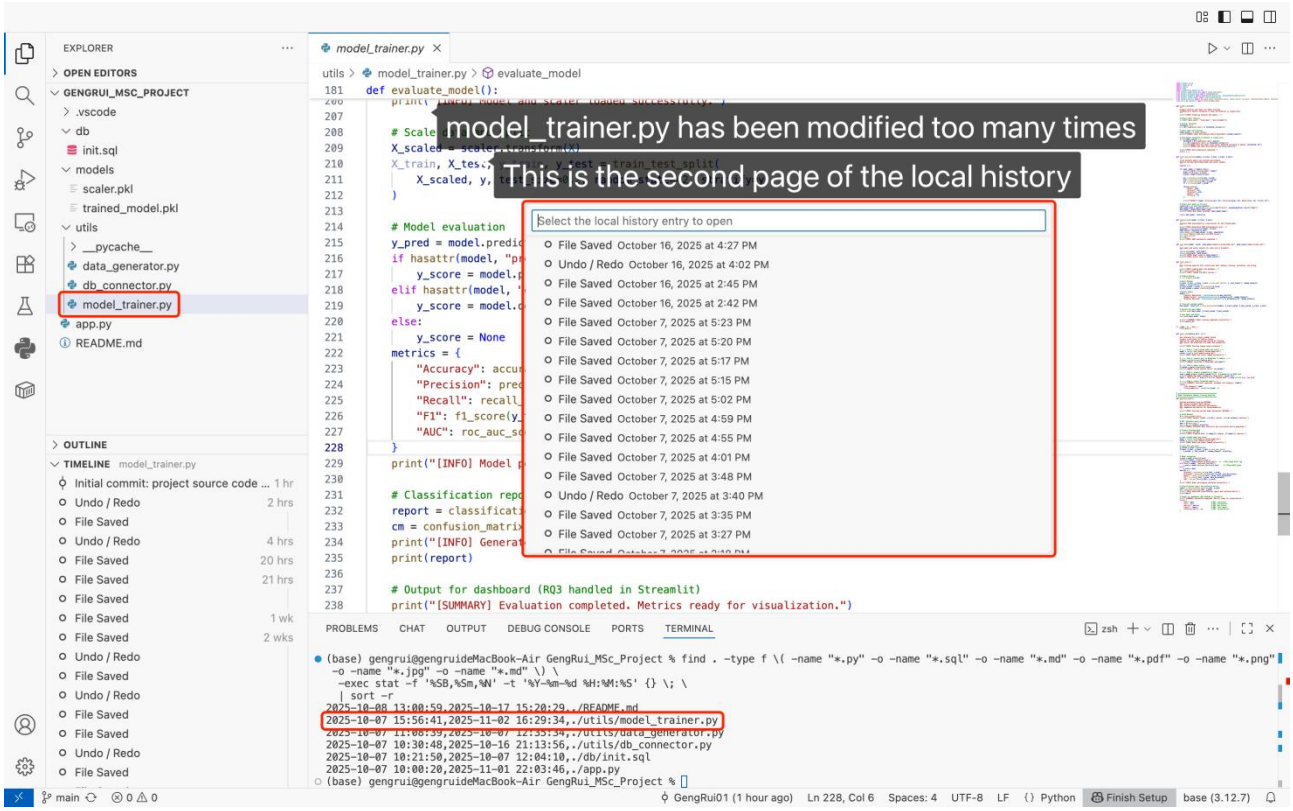

Figure 27. Local History of model_trainer.py - page2

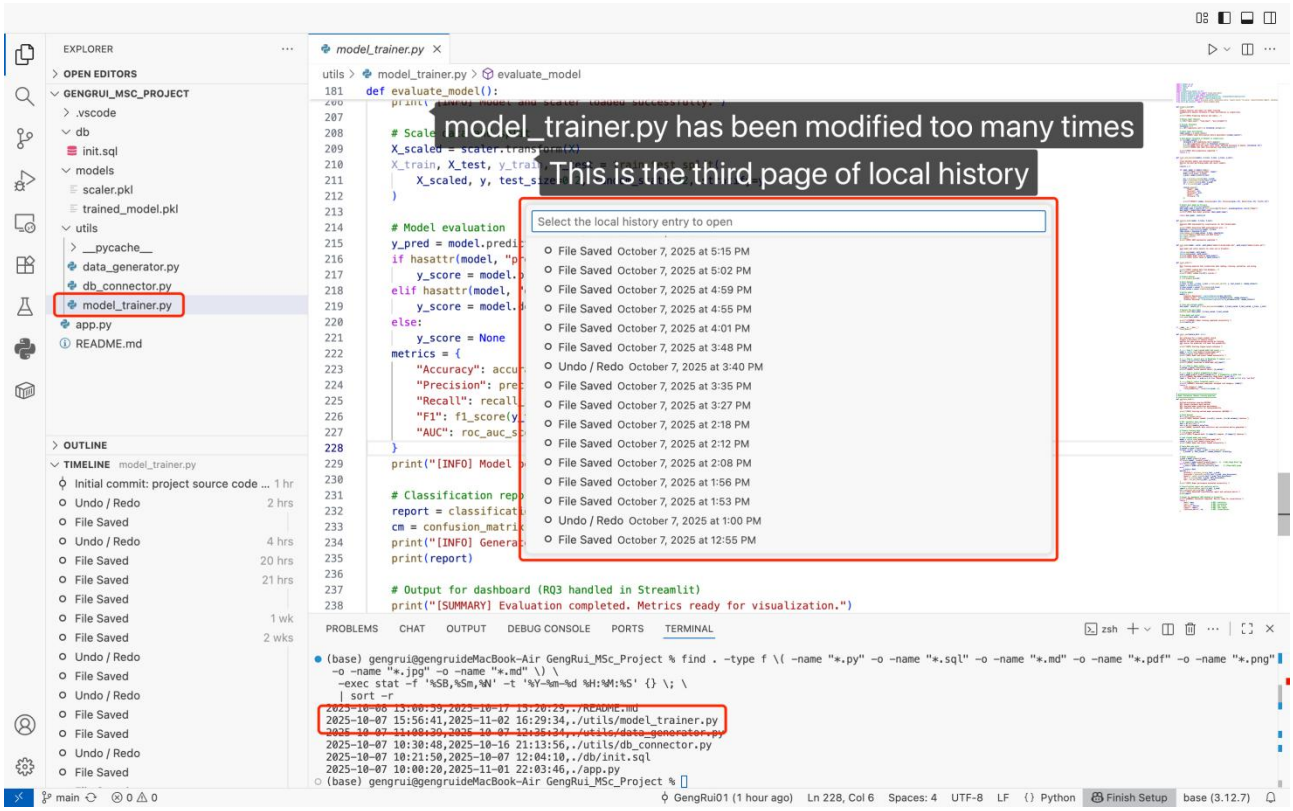

Figure 28. Local History of model_trainer.py - page3

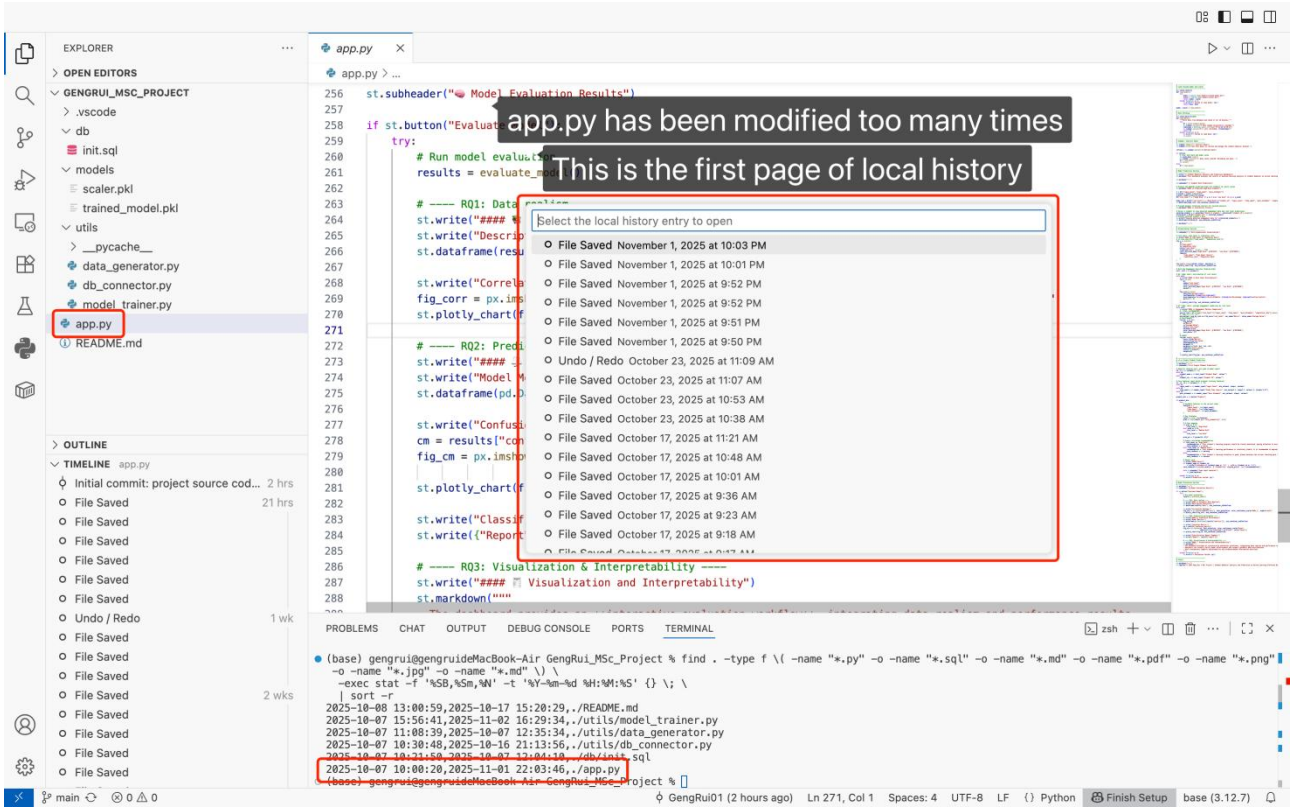

Figure 29. Local History of app.py - page1

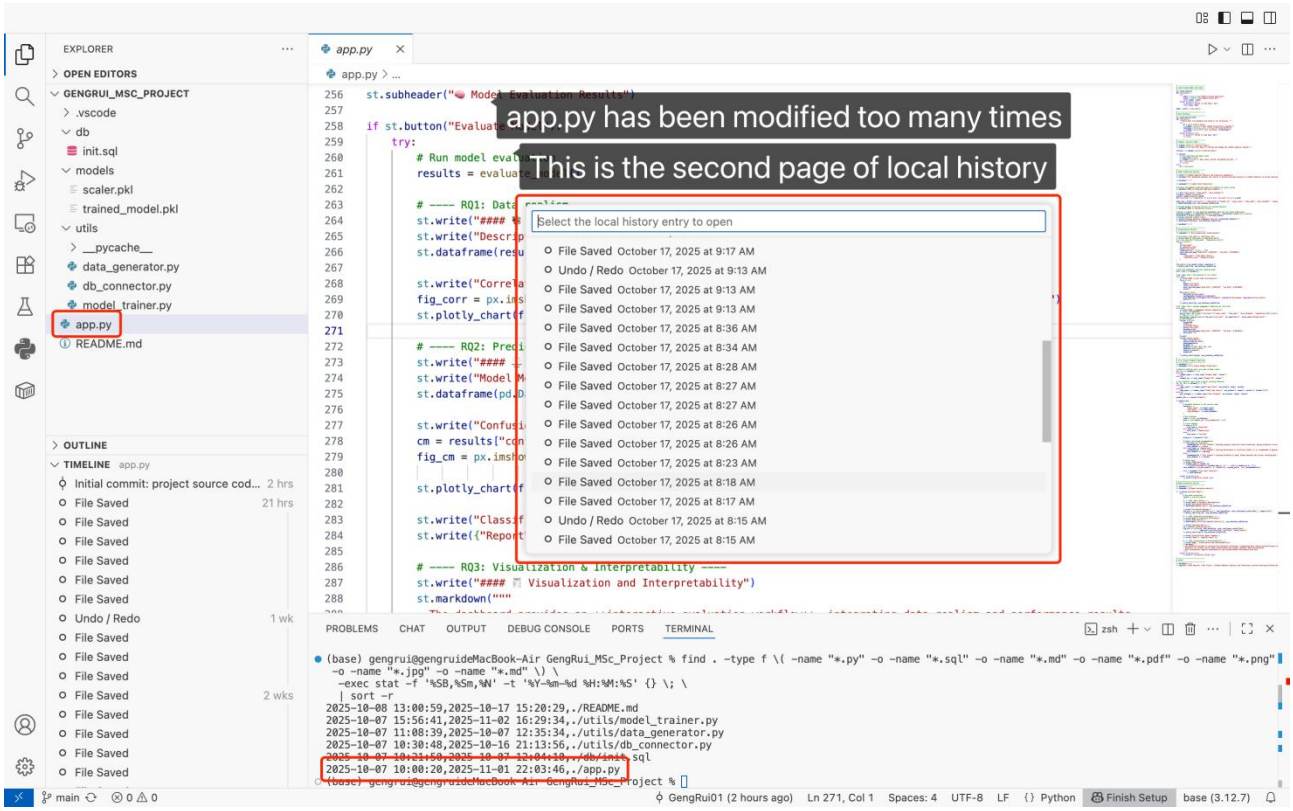

Figure 30. Local History of app.py - page2

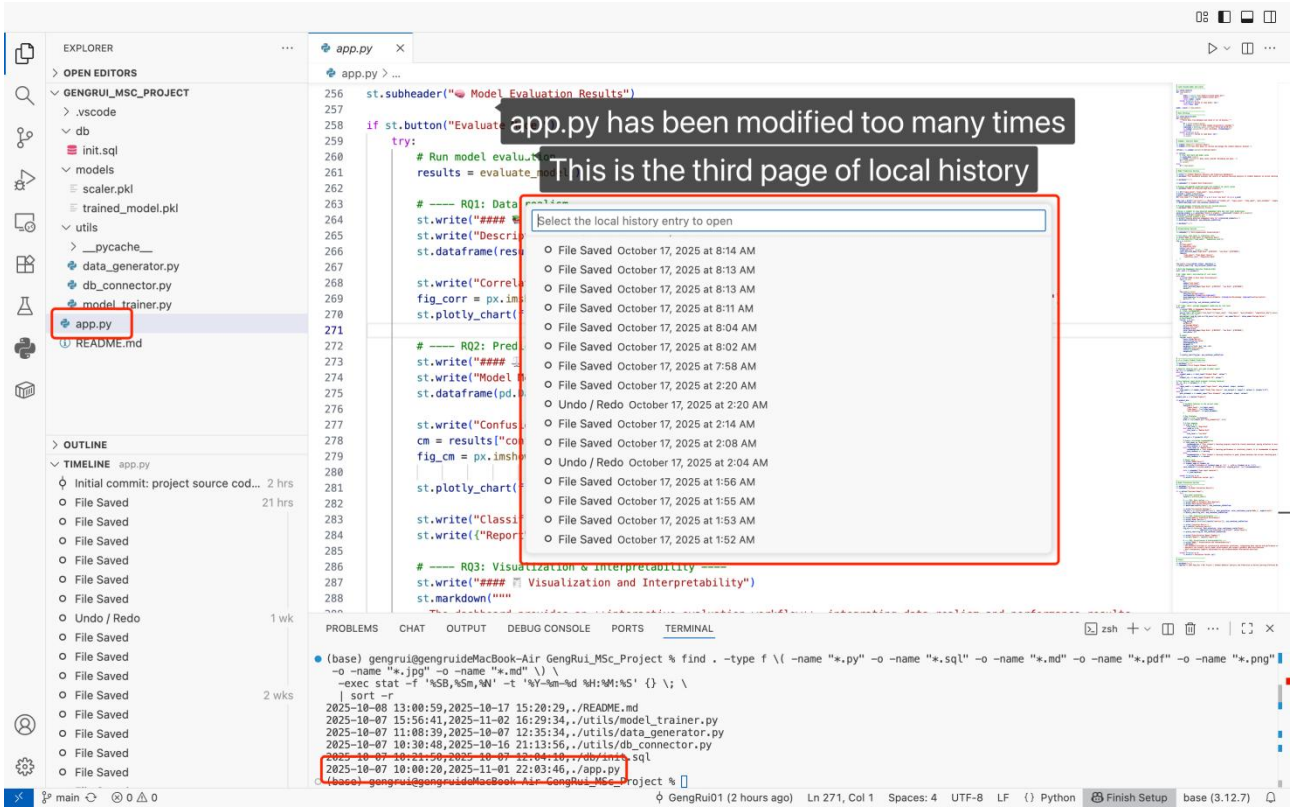

Figure 31. Local History of app.py - page3

B.3 File Timestamps: OS-Level, Collected at 2025-11-02

To provide verifiable absolute timestamps independent of the IDE or VCS, we exported OS-level file metadata on 2025-11-02 from macOS (APFS). The table lists each file's creation/birth time and last-modified time in local timezone, with paths recorded relative to the project root.

Table 9. File creation and modification times (OS-level metadata, collected 2025-11-02)

Path	Created (local time)	Last Modified Until 2025/11/2
./app.py	2025/10/7 10:00	2025/11/1 22:03
./db/init.sql	2025/10/7 10:21	2025/10/7 12:04
./utils/db_connector.py	2025/10/7 10:30	2025/10/16 21:13
./utils/data_generator.py	2025/10/7 11:08	2025/10/7 12:35
./utils/model_trainer.py	2025/10/7 15:56	2025/11/2 16:29
./README.md	2025/10/8 13:00	2025/10/17 15:20

B.4 Runbook Evidence: Timestamps of Model Artefacts

To document the persisted artefacts used by the dashboard and the evaluation pipeline, we captured OS-level timestamps directly from the models/ directory. The terminal capture below lists, for each artefact, its creation (birth) and last-modified times on macOS (APFS). These time anchors independently corroborate the training/evaluation window reported in Chapters 3 – 4.

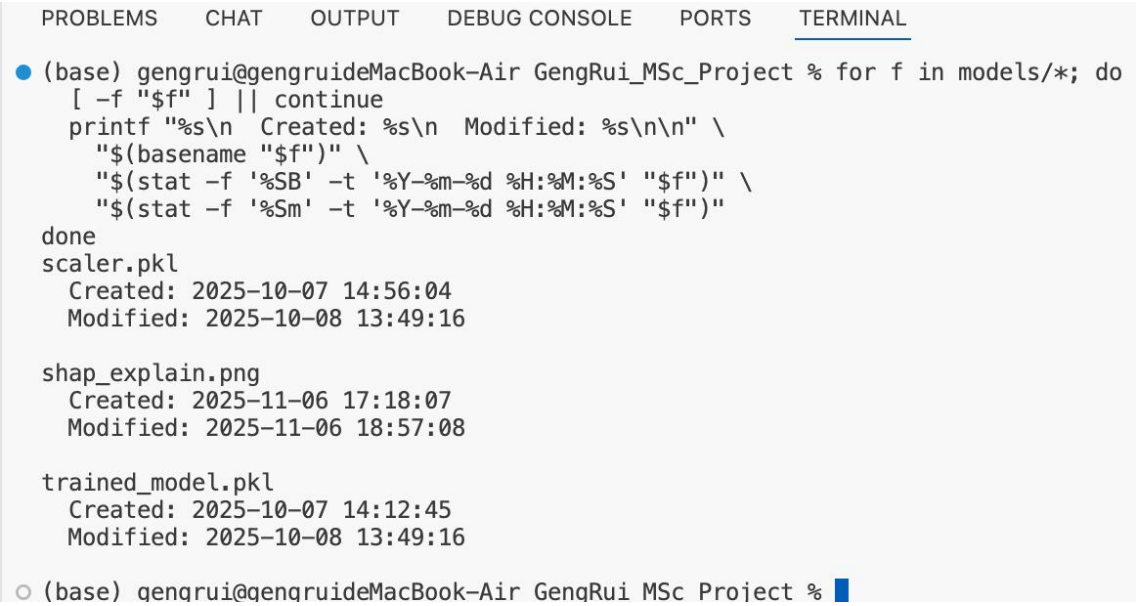

Figure 32. models/ artefacts — OS-level creation and last-modified times

B.5 Actual Timeline

This figure summarises the actual delivery from late September to early November 2025:

pre-implementation preparation (09/21 – 10/06); Iteration 1 (10/07 – 10/08) delivering V1 (delivered the core pipeline + initial UI); Iteration 2 (10/09 – 10/17) delivering V2 (revamped the visuals + added the Model Evaluation module); Iteration 3 (10/18 – 10/23) delivering the final V3 (implemented the Single-Student Prediction module); followed by report writing (10/24 – 11/02), slides & talk preparation (11/03 – 11/06), and submission (11/06).

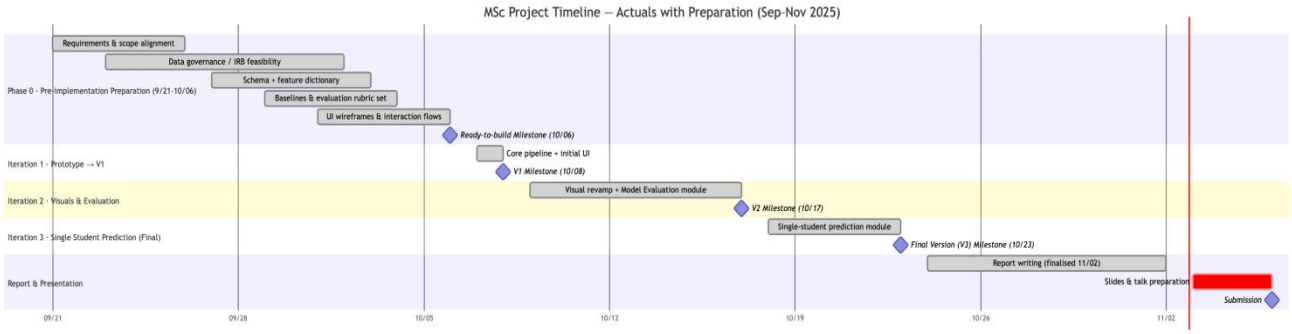

Figure 33. MSc project actual timeline — V1 (10/08), V2 (10/17), final V3 (10/23), report completed (11/02), presentation prep (11/03–11/06)

Appendix C: Artefact and Dataset Overview

This appendix lists the persisted artefacts used by the application: the scaler.pkl, the trained_model.pkl, and the explainability output used during evaluation. It also summarises the data tables and fields consumed by the dashboard. The purpose is to make the runtime dependencies explicit and auditable.

C.1 Repository

GitHub: GengRui01/GengRui_MSc_Project

OneDrive: GengRui01/GengRui_MSc_Project

C.2 Structure

```
GengRui_MSc_Project/
│
├─ db/
│  └─ init.sql                  # Creates the MySQL schema and tables used by this project.
│
├─ models/
│  ├─ scaler.pkl                # A fitted StandardScaler that normalises features exactly as in training.
│  ├─ shap_explain.png          # SHAP beeswarm image saved during evaluation and displayed in Visualisation and
Interpretability.
│  └─ trained_model.pkl         # The trained machine-learning model used by the dashboard.
│
├─ utils/
│  ├─ data_generator.py         # Generates synthetic student-behaviour data and writes it into the database.
│  ├─ db_connector.py           # Creates an SQL connection and fetches student records, auto-seeding the table if it's empty.
│  └─ model_trainer.py          # Holds the logic for model training, evaluation, and single-record inference.
│
├─ .gitignore                   # Ignore rules for caches, editor files, and other local artefacts.
├─ app.py                       # The Streamlit dashboard that brings prediction, visualisation, and evaluation together.
└─ README.md                    # Project documentation.
```

Figure 34. Project Structure

C.3 Environment Setup

To set up the environment, install all required Python packages with a single command:

```
pip install streamlit pandas numpy scikit-learn sqlalchemy pymysql plotly joblib shap
```

Any recent Python 3.x runtime is acceptable for marking. The project is dependency-light and platform-agnostic.

C.4 Data and Schema

Run the SQL script once to create the database and tables:

```
mysql -u root -p < db/init.sql
```

init.sql only creates the database and tables. On the first query, if the table is empty, the app will auto-seed synthetic data via utils/data_generator.py.

C.5 Configure Connection

Open utils/db_connector.py and set MySQL username and password in SQLAlchemy URL, e.g.

```
engine = create_engine(
  "mysql+pymysql://<USER>:<PASSWORD>@localhost:3306/gengrui_msc"
)
```

Data policy: During MSc, the dashboard runs on synthetic behavioural data generated locally (utils/data_generator.py); no real ICVE data are distributed. Post-MSc real-data validation will follow institutional agreements.

C.6 Reproducible Training and Artefacts

The model is trained with an 80/20 stratified split (random_state=42); a StandardScaler is fitted on the train fold only and then applied to both folds to avoid leakage. Three candidates are tried—Logistic Regression, Random Forest, Gradient Boosting—and the best is selected by a single criterion.

The pre-trained model (trained_model.pkl) and scaler (scaler.pkl) and shap (shap_explain.png) are already included under the models/ directory.

This figure shows the SHA-256 checksums for the released artefacts under models/. These hashes uniquely identify the files used in Chapters 3 – 4.

Figure 35. SHA-256 checksums for models/trained_model.pkl, models/scaler.pkl, and models/shap_explain.png

If you wish to retrain the model, you can manually run:

```
python utils/model_trainer.py
```

This will update the model files under the models/ directory.

C.7 Run the Dashboard

Start the Streamlit application:

```
streamlit run app.py
```

Then open your browser at http://localhost:8501.

On load you'll see Student Risk Prediction and Multi-dimensional Visualisation; scroll down to find Single Student Prediction and Model Evaluation Results.

Single Student Prediction: Enter Login Count, Study Time (hours), and Quiz Attempts (optional Student Name/ID) → click 'Predict' → the risk label and probability will appear in Results. 'High Risk' denotes students likely needing additional attention or intervention.

Model Evaluation Results: Click 'Evaluate Model' → the Descriptive Statistics, Correlation Heatmap, metrics table (Accuracy, Precision, Recall, F1, AUC), Confusion Matrix, and the Classification Report, and the SHAP beeswarm are displayed below, with High Risk treated as the positive class, meaning the metrics evaluate how well the model identifies students who need attention.

Appendix D: Screencast

This appendix provides a short, narrated walk-through of the teacher-facing workflow. It shows how the risk table is reviewed, how a single-student prediction is made, and how the evaluation page presents metrics and interpretability. The screencast complements the static figures in Chapter 3.

OneDrive: GengRui MSc Project Screencast_20251111.mp4

Date: 2025-11-11 Duration: 15 min