DEPARTMENT OF ENGINEERING MATHEMATICS

Comparative Machine Learning Analysis for Student Dropout Prediction in a Virtual Learning Environment

Incorporating Student Engagement and Socio-Economic Features

Carlos Duran Calle

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Science in the Faculty of Engineering.

Sunday 24th August, 2025

Supervisor: Dr. Felipe Campelo

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MSc in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Carlos Duran Calle, Sunday 24th August, 2025

Contents

List of Figures

List of Tables

List of Algorithms

‘

Abstract

A compulsory section, of at most 1 **page**

This section should summarise the project context, aims and objectives, and main contributions (e.g., deliverables) and achievements. The goal is to ensure that the reader is clear about what the topic is, what you have done within this topic, *and* **what your view of the outcome is.**

Essentially this section is a (very) short version of what is typically covered in more depth in the first chapter. If appropriate, you should include here a clear statement of your research hypothesis. This will obviously differ significantly for each project, but an example might be as follows:

> My research hypothesis is that a suitable genetic algorithm will yield more accurate results (when applied to the standard ACME data set) than the algorithm proposed by Jones and Smith, while also executing in less time.

The latter aspects should (ideally) be presented as a concise, factual list of the main points of achievement. Again the points will differ for each project, but an might be as follows:

- I spent 120 hours collecting material on and learning about the Java garbage-collection sub-system.
- I wrote a total of 5000 lines of *Python* source code, and associated orchestration scripts.
- I designed a new algorithm for computing the non-linear mapping from A-space to B-space using a genetic algorithm.
- I implemented a version of the algorithm proposed by Jones and Smith (2010), corrected a mistake in it, and compared the results with several alternatives.

Supporting Technologies

A compulsory section, of at most 1 page

This section should present a detailed summary, in bullet point form, of any third-party resources (e.g., hardware and software components) used during the project. Use of such resources is always perfectly acceptable: the goal of this section is simply to be clear about how and where they are used, so that a clear assessment of your work can result. The content can focus on the project topic itself (rather, for example, than including "I used LaTeX to prepare my dissertation"); an example is as follows:

- I used the *Pandas* and *Seaborn* public-domian Python Libraries.
- I used a parts of the OpenCV computer vision library to capture images from a camera, and for various standard operations (e.g., threshold, edge detection).
- I used Amazon Web Services for remote storage and processing of data. Specifically, I used:
  - Simple Storage Service (S3) for data storage
  - Elastic Compute Cloud (EC2) for provision of virtual machines
  - Elastic Beanstalk for scaling and load management
  - Sagemaker for all the machine learning components of my project.
- I used LaTeX to format my thesis, via the desktop service *TeXstudio*.

Notation and Acronyms

SDP	:	Student Dropout Predictor
MOOCs	:	Massive Open Online Courses
VLE	:	Virtual Learning Environments
OULAD	:	Open University Learning Analytics Dataset
OU	:	Open University
ML	:	Machine Learning
RF	:	Random Forest
LG	:	Logistic Regression
LightGBM	:	Light Gradient-Boosting Machine
NN	:	Neural Networks
SE	:	Student Engagement
IMD	:	Index of Multiple Deprivation
TMA	:	Tutor Marked Assessment
KNN	:	K-Nearest Neighbors
KNN	:	K-Nearest Neighbors
KNN	:	K-Nearest Neighbors
KNN	:	K-Nearest Neighbors
x_i	:	the i-th bit of some binary sequence x, st. $x_i \in \{0, 1\}$

Acknowledgements

An optional section, of at most 1 **page**

It is common practice (although totally optional) to acknowledge any third-party advice, contribution or influence you have found useful during your work. Examples include support from friends or family, the input of your Supervisor and/or Advisor, external organisations or persons who have supplied resources of some kind (e.g., funding, advice or time), and so on.

Dave Cliff writes here to say huge thanks to his colleague Dr Dan Page for sharing this LaTeX thesis template, which was originally written by Dan, for Computer Science dissertations. Dave edited Dan's original to better suit the needs of the Data Science MSc: please don't hassle Dan about any of this, but do feel free to contact Dave if you have any questions or comments on it.

Chapter 1

Introduction

Education is now more widely accessible thanks to online learning. However, high attrition has continued. The median completion rate for massive open online courses (MOOCs) is close to 12.6%, which has frequently remained low [1]. Large-scale attrition has also been documented in open-university environments. A study from the Open University (UK) indicates that course-level dropout rates can reach as high as 78% [2]. These numbers have inspired timely assistance and predictive systems.

The Open University (OU) released the widely used Open University Learning Analytics Dataset (OULAD), a public dataset for learning analytics research. OULAD contains information from 22 module presentations and 32,593 registered students. Demographics, tests, and daily click-log summaries from the Virtual Learning Environment (VLE) are among these data [3]. It is currently regarded as the de facto standard for feature engineering and early-warning models in distance learning because of its size and documentation [3]. For example, Hussain et al. [4] discovered that engagement signals predict outcomes consistently across platforms. The current study builds on this evidence by predicting engagement-aware early intervention in the OU context using OULAD.

One possible remedy for this issue is machine learning (ML). Large volumes of student data can be analysed using ML algorithms to find patterns that human observers might miss [5]. These algorithms can identify which students are most likely to leave the VLE early by analysing data on their click behaviour, assessment submission, and resource access [6]. Dropout prevention becomes predictive rather than reactive as a result of this predictive ability.

1.1 Problem Statement

The prediction task is structured as a three-class classification: Pass, Fail, and Withdrawn. This framing supports targeted responses (e.g., academic support for likely fail; re-engagement for likely withdrawn) and reflects OULAD's *final_result* taxonomy. This outcome schema and its connection to registration and assessment tables are described in detail in the OULAD documentation [3].

There is a noticeable disparity in class. The Withdrawn class made up 19.1% of the records in the project's processed dataset, making it the minority class for the three-way target. The other classes had 55.5% Pass and 25.4% Fail. In line with earlier findings that OULAD results are unequally distributed across Pass/Fail/Withdrawn categories, this imbalance makes model training and evaluation more difficult, particularly for recall on the minority class [7].

Early detection is also necessary. Delivering interventions before disengagement solidifies has a greater impact. The importance of early-phase risk detection in reducing withdrawal has been highlighted by previous work in open online courses [8]. As a result, the objective is to provide actionable, early warning in addition to final outcome prediction.

1.2 Research Objectives

According to earlier studies, student engagement is a significant predictor of both academic success and dropout, and engagement proxies that are derived from VLE activity traces have shown especially good

results [4]. Similarly, when examined alongside engagement patterns, sociodemographic factors like age, region, disability, and prior education have been demonstrated to correlate with withdrawal [3].

The specific objectives of this project are:

- To incorporate the student engagement feature as a meaningful predictor for the ML models.
- To determine which socio-demographic factors are significantly associated with student dropout..
- To choose a predictive model that can recognise students who are likely to drop out of a course early on.

The accomplishment is meaningful beyond the technical achievement. Accurate identification of dropout can also help institutions to determine the best investment of their support resources, develop more effective interventions and enhance the retention of students [9]. For students, that support can be the difference between meeting educational goals and falling among the dropout numbers.

1.3 ML Approach

This project uses and compares six widely used ML models with complementary advantages for dropout prediction: Random Forest (RF), an ensemble method known to be robust to high-dimensional data [10]; Logistic Regression (LR), to interpret the results for risk factor analysis [11]; K-Nearest Neighbors (KNN), to capture local data patterns based on instance-based learning [12]; LightGBM, optimized gradient boosting to handle large datasets efficiently [13]; Support Vector Machine (SVM), capable of dealing with high-dimensional spaces via kernel-based transformations [14]; and Neural Networks (NN), to model complex non-linear data relationships [15].

Every model will be tuned with respect to hyperparameters by GridSearchCV with 5-fold cross-validation. The main evaluation metric is recall of the "Withdrawn" class, as we attempt to identify all at risk students. This emphasis on minority class recall is paramount as missing out a student that requires assistance has higher stakes than sometimes offering help even if it may not be necessary [16].

1.4 Contributions

The present study builds upon existing research with three important advances. First, new engagement features are developed to model students' behaviour before their first attempt at the assessment. The *StudentEngagement* (SE) variable is precipitated with academic status along with VLE logs, similar to the approach of Hussain et al. [4]. It is demonstrated in this study that this composite measure allows better capturing student engagement than single traditional measures.

Second, socio-economic variables, in particular the Index of Multiple Deprivation (IMD), are added to explore to what extent dropout risk is influenced by exogenous factors. As compared with the other OULAD studies of Tomasevic et al. [17] and Hussain et al. [4] indicated demography, these relations were not further investigated, in contrast socio-economical aspects came out as some of the strongest drop-out predictors.

Third, it was developed a thorough model comparison framework with a unified evaluation schema. All models are trained on the same data split with data-specific class-weighting (Withdrawn: 2.51, Fail: 1.57, Pass: 0.57) to account for class imbalance, therefore fairly comparable and giving insights into a best approach for deployment.

1.5 Challenges and Scope

This research is faced with many challenges. Attention to class imbalance is required so that models do not ignore the minority dropout class. These temporal characteristics require careful feature engineering to generate predictions early enough to allow for interventions. In addition, stakeholders need to be able to understand why students are being flagged as at-risk, which necessitates a balance between model complexity and interpretability.

The complete methodology is presented in this dissertation, covering the process from data preparation through model deployment. Chapter 2 provides technical background on educational data mining and ML algorithms. Chapter 3 outlines the implementation pipeline and feature engineering approach. Chapter 4

reports the results and model comparisons in detail. Chapter 5 offers a critical evaluation of the findings, and Chapter 6 concludes with practical recommendations and directions for future research.

Through systematic comparison and optimization, Logistic Regression is shown to achieve a 66.84% dropout recall, crossing the 60% threshold while keeping interpretability. Such result provides the educational institution with a useful instrument to identify and support at-risk students, thus contributing to the general objective of successful online education.

In summary, this project focuses on performing and comparing six ML models to identify the best-performing model for detecting students who have withdrawn from online courses. By emphasizing recall for the 'Withdrawn' class and incorporating engagement and socio-economic factors, the study aims to provide actionable insights that support timely interventions and improve student retention in virtual learning environments.

Chapter 2

Technical Background

This chapter offers a theory based on student dropout prediction in VLE. Key concepts that comprise educational data mining, multi-class classification problems, and ML algorithms are discussed. The discussion is the technical basis for a comparison among dropout prediction models.

2.1 Educational Data Mining in VLE

Educational Data Mining in VLE explores the wealth of behavioural data produced by students (ie, clickstream patterns and resource access statistics) for the purpose of gaining knowledge on how they learn [18]. Browsing and resource navigation behaviour have been shown to be strong predictors of academic performance, such as course completion [19]. Combining this behavioural data with a learners' demographics and assessment results substantially enhances prediction accuracy of dropout models [20].

The temporal nature of this data is important. Early warning signs of at-risk students can be well-predicted from early-stage behaviour patterns [21]. The high-frequency of VLE data streams on the other hand requires advanced preprocessing to deal with missing values, irregular sampling intervals, and varying engagement patterns across different student populations [22]. In addition, integrating VLE interaction data with socio-economic features improves model stability and classification performance, and paves the way for a more systematic feature engineering in educational prediction systems [23].

2.2 The OULAD Dataset Architecture

OULAD is released as a set of CSV tables that can be joined through surrogate keys, enabling a student-centric relational view across demographics, registrations, assessments, learning materials, and VLE interactions [3], the database schema can be appreciated in the Figure 2.1 [24]. The public release contains 22 module-presentations, $32,593$ students, and $10,655,280$ daily click summaries, supporting at-scale analyses of behaviour and outcomes. All dates are stored as offsets relative to the module-presentation start, which simplifies time-window filters such as "before the first assessment".

2.2.1 Table Descriptions

Summaries below follow the official data descriptor for OULAD [3].

- Demographics and final result per student-module presentation (e.g., gender, age band, prior education, credits, disability, and $final\_result$).
- Registration and unregistration days for each student in each presentation; empty unregistration implies completion.
- Submission day and score (0–100) for each assessment attempted.
- Per-assessment metadata: type (TMA/CMA/Exam), cut-off day, and weight; non-exam assessments sum to 100 and exams are treated separately.

- Daily counts of interactions (*sum_click*) by student with specific learning materials (*id_site*).
- The catalog of VLE materials with activity type and planned availability windows (*week_from*, *week_to*).
- Module and presentation identifiers with presentation length in days.

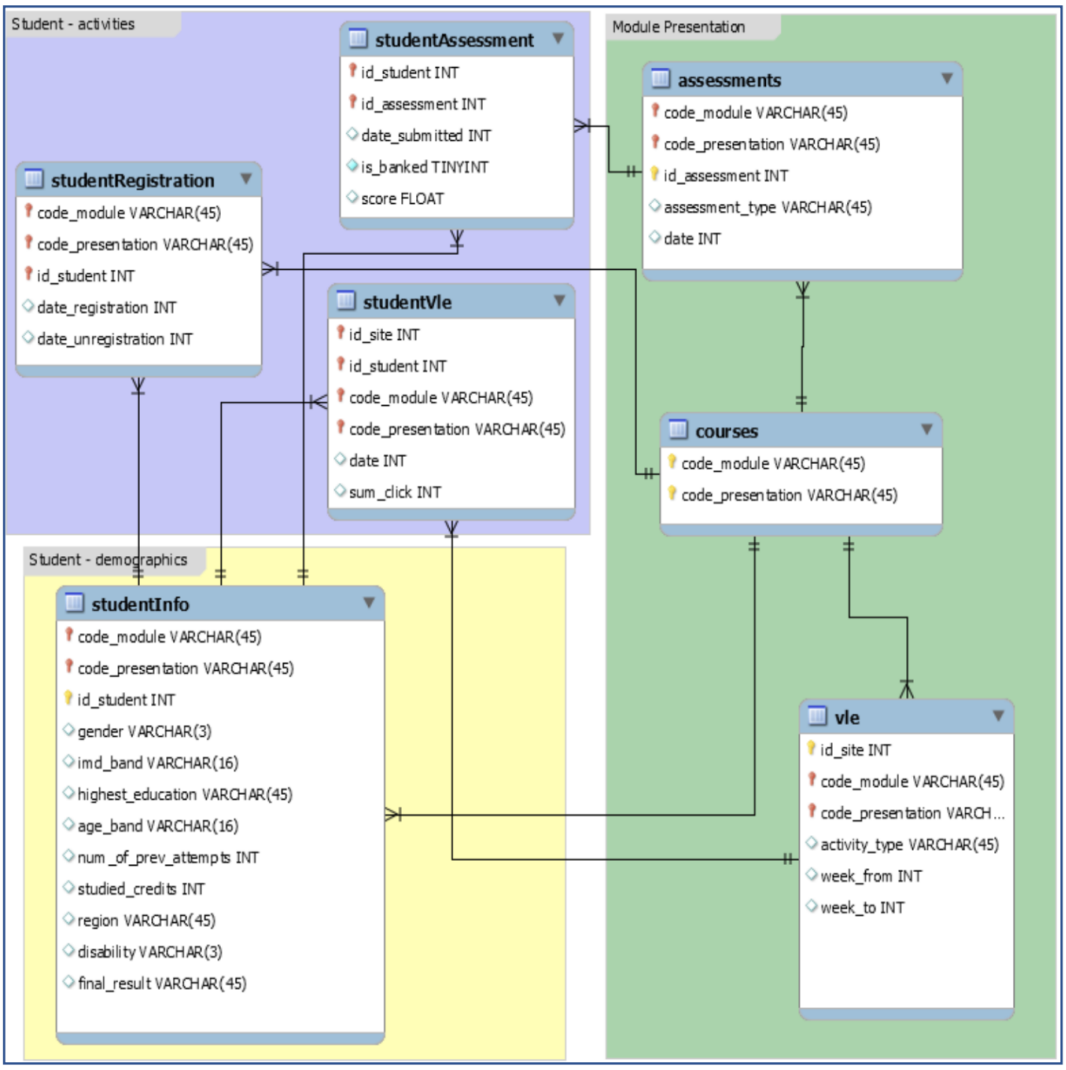

Figure 2.1: Database schema of OULAD dataset (reproduced from [24]).

2.3 Student Engagement Indicators

Simple, interpretable flags were engineered to capture early excellence and activity in ways that align with common early-warning practices in learning analytics [25]. Mathematical notations are added no understand following equations.

- Let $\mathbf{1}[\cdot]$ be the indicator function (1 if the condition is true, else 0).
- Let the first graded assessment in a presentation be indexed by a_1
- Let score a_1 be the student's first-assessment score (0–100).
- Let $date(a_1)$ be its cut-off day.
- Let C_{pre} be the student's total *sum_click* with $studentVle.date < date(a_1)$.

- Let μ_{pre} be the mean of C_{pre} within the same (*code_module*, *code_presentation*).
- These columns and date conventions are defined in OULAD.

2.3.1 Excellent Score Indicator

A binary indicator for early academic excellence was defined as:

$$\text{excellent\_Score} = \mathbf{1}[\,\text{score}(a_1) \geq 70\,] \tag{2.1}$$

According to the Scholaro database, The Open University awards a "Merit" scale when is higher than 70% [26]. This threshold marks, where is categorised as distinction mark for OU, clearly high performance while keeping the rule easy to interpret; early assessment performance has been shown to be informative for final outcomes and targeted support.

2.3.2 Active in VLE Indicator

A binary indicator for above-average pre-assessment activity was defined as:

$$\text{active\_in\_VLE} = \mathbf{1}[\,C_{\text{pre}} > \mu_{\text{pre}}\,] \tag{2.2}$$

Clicks prior to the first assessment are counted and compared with the cohort's mean for the same module-presentation; VLE click behaviours have repeatedly shown predictive value for course performance, including in OULAD-based studies [27].

2.3.3 Student Engagement Indicator

A composite engagement flag was defined with a logical OR:

$$\text{student\_engagement} = \mathbf{1}[\,\text{excellent\_Score} = 1 \,\vee\, \text{active\_in\_VLE} = 1\,] \tag{2.3}$$

This rule fires if either early excellence or above-average activity is observed, a common, conservative design for early-warning heuristics that favours recall of potentially successful or engaged students [25].

Why these signals The pair (grade-based, behaviour-based) captures complementary facets of engagement and aligns with evidence that clickstream engagement and formative performance jointly inform later achievement [28].

2.4 Stratification and Encoding

2.4.1 Cohort-Outcome Stratification

A stratified hold-out split was performed per cohort so each module–presentation kept its original class mix, and the target values (withdrawn/fail/pass) was included in the strata to preserve outcome prevalence within every cohort. Class-imbalance concerns motivated stratification to reduce variance and avoid misleading metrics on under-represented outcomes [29]. The split was executed before any encoding to prevent information leakage from validation/test back into training [30]. The implementation relied on scikit-learn splitters with a fixed random state for reproducibility [31]. In the project, three alternatives were examined and Strategy 2 (cohort + outcome) was retained due to its better distribution in comparison with the Strategy 1 (cohort); a triple-key variant Strategy 3 what include "courses per term" was discarded due to tiny cells that break stratified sampling.

2.4.2 Feature Encoding

Encoding was intentionally minimal and applied only to a selected set of categorical variables that it was found relevant for the SDP: *region*, *highest_education*, *imd_band*, *age_band*, and *disability*; numerical and binary features were left unchanged to keep the signal simple. Nominal fields were one-hot encoded to avoid imposing order, while ordered bands (*imd_band*, *age_band*) were ordinal-encoded to preserve ranks in a compact form [32]. Encoders were fit only on the training split and then applied to test dataset, with unseen categories mapped safely to an "Unknown" bucket to avoid runtime errors and leakage [32]. The steps were orchestrated with *scikit-learn* library and the encoded matrices for the train and test dataset, and labels were persisted for downstream modelling.

2.5 Multi-Class Classification

In education, multi-class classification is complicated since the standard binary techniques are not effective in modelling the interactions among various performance levels, which leads to specialized algorithms [33]. Model collection is in turn influencing the ordinal nature of effects (e.g., from "Pass to Fail") might need attentive attention at some stage in model training [34]. Standard decomposition methods may bias such type of ordered data, which validate why it is useful to apply native multi-class algorithms in order to predict student performances accurately [35].

Class imbalance is a critical challenge in educational data, as traditional accuracy metrics can be misleading by failing to reflect poor performance on key minority classes, such as "Withdrawn" students [36]. Proper evaluation of these models requires specialized metrics, such as class-specific recall and macro-averaged F1 scores, to ensure a balanced assessment [37]. Advanced techniques, including strategic oversampling combined with ensemble methods, have been shown to significantly improve the identification of these at-risk students [38].

2.6 Multi-Class Evaluation Metrics

The class-specific performance metrics can offer valuable insights on how a model performs and work well for imbalanced data. Among these, precision and recall are core indicators for assessment of classification effectiveness. For a given class i, precision is defined as the ratio of true positive predictions to all positive predictions, as shown in Equation 2.1. This concept is introduced in [16].

$$\text{Precision}_i = \frac{TP_i}{TP_i + FP_i} \tag{2.1}$$

where TP_i represents true positives and FP_i represents false positives for class i.

On the other hand, recall is defined as the ratio of the actual positive instances that are classified as positive which is given in Equation 2.2 [16]. In educational applications, recall for the "Withdrawn" class is especially important as it measures the model's capacity to detect students who need early support, while having high recall rates ensures that such at-risk students are not missed out even with elevated false positive rates [39].

$$\text{Recall}_i = \frac{TP_i}{TP_i + FN_i} \tag{2.2}$$

where TP_i represents true positives and FN_i denotes false negatives for class i.

Weighted metrics compensate for the class imbalance by incorporating the relative frequency of each class contributing to the overall performance measure and deliver a model effectiveness evaluation that is more representative of how well it generalizes to all categories [40]. Weighted recall is calculated according to the Equation 2.3 [41].

$$\text{Weighted Recall} = \sum_i w_i \times \text{Recall}_i \tag{2.3}$$

where w_i is the weight for class i, usually based on its frequency in the dataset.

Weighted F1-score is the harmonic mean of precision and recall for each class i [41] [42] as in the Equation 2.4. These balanced metrics allow us to measure the performance level for all categories of evaluation indicators while still being sensitive to the practical important of minority class detection in the educational intervention systems [35].

$$\text{Weighted F1} = \sum_i w_i \times F1_i \tag{2.4}$$

where $F1_i = 2 \times \frac{\text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$ and w_i represents the weight assigned to class i.

2.7 Strategies for Imbalanced Data

Balanced class weight is one of the basic approaches to address the imbalance of the dataset where the importance of each class is adjusted automatically and inversely proportional to the frequency of that class and the mathematical foundation is given as it is written in the Equation 2.5 [43].

$$w_i = \frac{n_{\text{samples}}}{n_{\text{classes}} \times n_{\text{samples}_i}} \tag{2.5}$$

where w_i denotes the weight for class i, n_{samples} represents the total number of samples, n_{classes} indicates the number of classes, and n_{samples_i} represents the number of samples in class i.

Customized scoring metrics deepen this approach by considering domain-specific preferences, by dropout-specific optimization forcing the maximization of the recall for withdrawal detection via biasing the learning criterion in favour of at-risk students and focusing less on false negatives [44].

Early warning signs do exist to prevent high-risk youngsters and to intervene early. In the context of dropout prediction, the imbalance class is the minority dropout class, which results in heavy emphasize on recall to detect the dropout students for timely intervention [45]. Unbalanced-learning techniques (e.g., resampling or class weighting) generally improve recall for the rare class (dropout) at the expense of precision [44], which is the same precision-recall trade-off this work aims to find the best level of.

In the area of dropout early-detection, balancing the class distribution among minority and majority instances should be considered desirably by using class weights, which disadvantage the majority and advantage the minority class, e.g., the weight of the minority class being higher than 1 and the weight of the majority class being equal to 1 is common [46]. More recent losses, e.g., the Class-Balanced Loss, have tried to provide a more principled re-weighting beyond naïve inverses only when classes are long-tailed [47]. For the current study it is beneficial as we would like to investigate alternative to ad-hoc multipliers, used for balanced weighting, like x1.2 for the minority class "Withdrawn" and x0.8 for the majority class "Pass". The multipliers are domain specific choices of the experimentations: there is one setting that uses no multipliers (balanced weighting), and one that applies the multipliers. Lastly, note that overall accuracy may be deceptive of imbalanced data, as Precision and Recall can better assess minority-class detection quality [48].

2.8 ML Algorithms for SDP

Ensemble learning exploits multiple models to reduce variance and improve generalization; in RF, bootstrap aggregation (bagging) draws resamples of the training set and averages randomized decision trees, providing robustness and out-of-bag error estimation [49] [10]. Feature importance in RF is commonly derived from impurity decreases or permutation-based accuracy drops, enabling screening of influential predictors in heterogeneous educational data [10]. Gradient-boosting frameworks fit weak learners sequentially under an additive model to minimize a differentiable loss, while LightGBM accelerates this process using histogram-based splits, leaf-wise growth, and gradient-based one-side sampling for scalability on high-dimensional, large cohorts [50] [13]. LR models the log-odds of class membership and estimates coefficients by maximum likelihood, yielding interpretable effects as odds ratios and supporting regularization for stability when features are correlated [51] [52]. SVM maximize the geometric margin via convex quadratic programming and use kernel mappings to induce flexible non-linear decision boundaries while controlling capacity through the margin and kernel parameters [14].

Instance-based learning with KNN classifies by a majority (plurality) vote among the K closest samples, so decision boundaries are local and sensitive to the choice of distance metric and K, with common metrics including Euclidean, Manhattan, and Mahalanobis distances [12] [53]. Multi-layer perceptrons (a type of NN) compose affine transformations with non-linear activations and are trained end-to-end by backpropagation to minimize a supervised loss, with universal approximation guaranteeing expressive capacity under sufficient width or depth [54] [55]. These foundations motivate comparing RF, LightGBM, LR, SVM, KNN, and NN for dropout prediction, balancing interpretability, non-linear modelling power, and computational efficiency under class imbalance typical of at-risk cohorts [35].

2.9 Model Evaluation and Comparison Methodologies

Performance was estimated with stratified 5-fold cross-validation [56] on the training split to preserve class proportions and reduce variance in imbalanced data, with a fixed stratified train–test split reused for all models to ensure comparability [37]. Hyperparameter search was executed within cross-validation (GridSearchCV) and final metrics were computed once on the untouched test split to limit selection bias [57]. All models were evaluated on identical folds and scoring rules to enable fair, paired comparisons across the pipeline [58]. Hyperparameters were optimized via exhaustive GridSearchCV with five folds, parallel execution, dropout-recall as the objective, and class-weighting to emphasize minority cases [56] [35].

Regularisation and early stopping are treated as crucial mechanisms to curb over-fitting and stabilise learning dynamics [59]. In LR, the strength of regularisation is controlled by the inverse-penalty parameter

C, with $L1/L2$ penalties shaping sparsity and shrinkage behaviour [60]. For SVM, the soft-margin constant C and kernel scale gamma are tuned to balance margin violations and function smoothness [14]. In decision trees and ensemble methods, maximum depth and the number of estimators are adjusted to limit variance and improve robustness [10]. In gradient boosting, the learning rate and number of boosting rounds jointly govern additive model capacity and error reduction [50]. For KNN, the neighbourhood size k and the choice of distance metric determine locality and boundary smoothness [12]. In multilayer perceptrons, hidden-layer size and weight decay are set under backpropagation to control capacity and enhance generalisation [61].

Chapter 3

Methodology and Implementation

This chapter describes the methodical process used to create and assess ML models for student dropout prediction, covering everything from data preprocessing to model deployment and validation.

3.1 Overview and Research Design

With an emphasis on early intervention capabilities, this study applies a ML pipeline intended to forecast the likelihood of student dropout in online learning environments. By creating predictive models that can identify at-risk students before they drop out of their courses, the study tackles the crucial problem of student retention in higher education.

3.1.1 Research Methodology Framework

The approach uses supervised learning and compares six different machine ML in a methodical manner: RF, Multinomial LR, KNN, LightGBM Gradient Boosting, SVM, and NN. Robust evaluation across various learning paradigms, ranging from distance-based and deep learning approaches to ensemble methods and linear models, is ensured by this thorough algorithmic comparison.

From data intake to model winning selection, the research framework uses a structured pipeline (Figure 3.1), integrating modular Python architecture for scalability and reproducibility. Each algorithm undergoes rigorous hyperparameter optimisation using GridSearchCV with 5-fold cross-validation to guarantee fair comparison and optimal performance extraction.

3.1.2 Dropout Optimization

To enable successful early intervention systems, the main goal is to maximise dropout recall $\geq 60\%$. Since false negatives, or missing at-risk students, have more serious repercussions than false positives, or failing students who might succeed, in educational intervention contexts, this metric gives priority to identifying students who will drop out. In order to balance practical resource constraints with effective early intervention capabilities, the 60% threshold was set especially for this project. This ensures that roughly two-thirds of at-risk students are identified while maintaining manageable caseloads for institutional support services.

3.1.3 Class Imbalance Challenge

With 19.1% dropout students (minority class), 25.4% failing students, and 55.5% passing students (majority class), the dataset represents a serious class imbalance issue. Specific optimisation techniques, such as unique class weighting schemes and dropout-focused evaluation metrics, are needed to address this imbalance. By using class weights of roughly 2.51x for dropouts, 1.57x for failures, and 0.57x for passes across various algorithms, the research employs customised strategies to make sure minority class detection is not overpowered by majority class dominance.

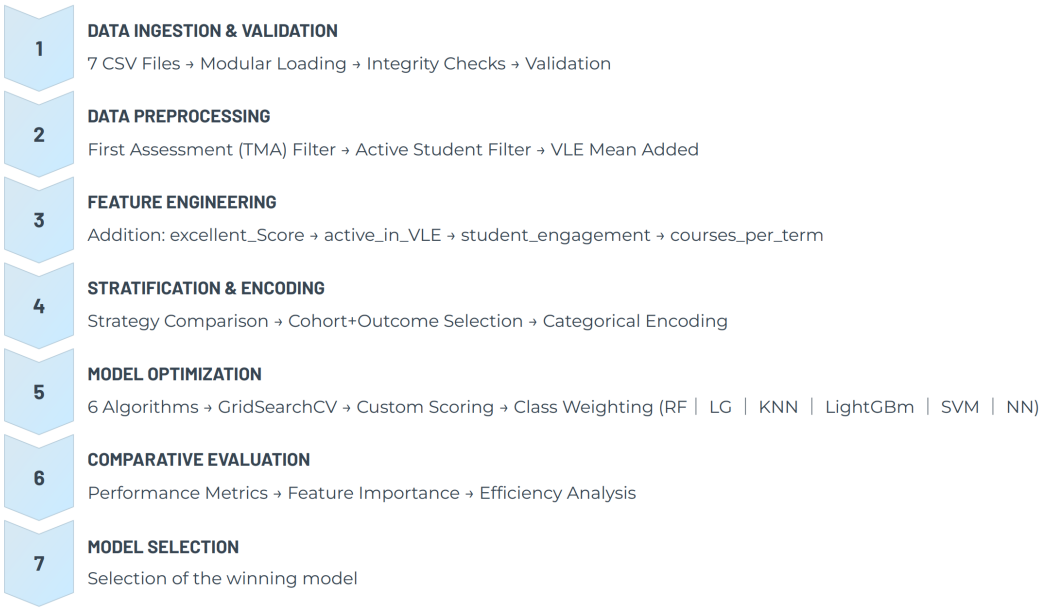

Figure 3.1: The Seven-Phase Project Pipeline

3.2 Data Architecture and Modular Design

As shown in Figure 3.1, the implementation uses a methodical seven-phase pipeline architecture intended for comprehensive SDP. From the intake of raw data to the deployment of models, this modular design guarantees repeatable processes while preserving data integrity across the ML pipeline.

3.2.1 Phase 1: Data Ingestion and Validation

Seven CSV files containing learning analytics data are automatically loaded at the start of the pipeline. Validation protocols check column structures and data integrity.

3.2.2 Phase 2: Data Preprocessing

Predictive modelling is based on core data transformations. By determining the earliest TMA (Tutor Marked Assessment) dates for each course-presentation combination, assessment filtering establishes standardised prediction windows for early intervention. Consistency across diverse data sources is ensured by data type conversions and missing value handling. Only participants who are actively participating during assessment periods are kept by student filtering protocols, which eliminate inactive participants.

3.2.3 Phase 3: Feature Engineering

From raw learning analytics, sophisticated feature creation algorithms produce predictive indicators. Before assessment deadlines, click-stream data is compiled by VLE engagement metrics to identify patterns in behaviour. Academic performance indicators establish engagement thresholds and binary flags for excellence (scores ≥ 70). The *courses_per_term* feature provides context for student capacity analysis by quantifying the distribution of academic load.

3.2.4 Phase 4: Stratification and Encoding

Advanced stratification techniques preserve demographic representation while maintaining class proportions across cohort boundaries. With 19.1% dropout representation, the chosen cohort+outcome strategy guarantees balanced train-test splits. The specific transformations used in categorical encoding are binary encoding for dichotomous variables, ordinal encoding for hierarchical relationships, and one-hot encoding for nominal variables.

3.2.5 Phase 5: Model Optimization

Using thorough hyperparameter optimisation, a systematic algorithm comparison assesses six different ML techniques. 5-fold cross-validation in GridSearchCV guarantees equitable comparison across various learning paradigms. Custom scoring metrics use specific weighting strategies to manage class imbalance while giving priority to dropout recall optimisation.

3.2.6 Phase 6: Comparative Evaluation

A thorough model evaluation analyses performance across six algorithms using a variety of visualisation techniques. Direct performance comparison is made possible by horizontal bar charts that compare key metrics (dropout recall, dropout precision, at-risk recall, weighted F1). Multi-metric performance profiles for each model are displayed using five-dimensional radar plots, which also highlight the models' advantages and disadvantages. Runtime efficiency scatter plots determine the best algorithms for various deployment scenarios by analysing trade-offs between computational cost and performance. With comprehensive confusion matrices for every algorithm, three-panel performance dashboards categorise models into three performance tiers: Excellent $\geq 60\%$, Good $40 - 60\%$, and Needs Work $< 40\%$.

3.2.7 Phase 7: Model Selection

The comparative analysis in Phase 6 is the source of the systematic evaluation criteria used in evidence-based model selection. Dropout recall performance ($\geq$ 60% threshold), computational efficiency, and deployment feasibility are given top priority during the selection process. With a 66.84% dropout recall, superior efficiency (4-minute training time), and interpretability advantages, LR is the best option. Runtime efficiency analysis, practical deployment considerations, and performance tier classification (Excellent, Good, Needs Work) are among the selection criteria. While underperforming algorithms (NN at 18.67% recall) are not taken into consideration for production, runner-up models (SVM with 63.09% recall) offer backup options. The final choice strikes a balance between operational needs for actual educational intervention systems and predictive performance.

3.3 Data Preprocessing and Feature Engineering

A systematic six-stage pipeline is implemented by the data preprocessing and feature engineering phases (Phase 2 and Phase 3 in Figure 3.1), which convert raw learning analytics into predictive features for early intervention systems (Figure 3.2). Prior to determining academic outcomes, this creates behavioural indicators and temporal prediction windows that capture patterns of student engagement.

3.3.1 Stage 1: Integrating Data Sources

The pipeline starts with the thorough integration of five main data sources: student registrations, VLE interaction logs, assessment structure and student assessment metadata, and demographic data. The original dataset includes over 10 million VLE interactions and 32,593 student registrations across 22 courses.

3.3.2 Stage 2: Defining the Initial Assessment Period

For every course-presentation combination, this stage identifies the first TMA dates. In order to determine the earliest assessment dates for 22 courses, this processes 106 TMA assessments. By eliminating participants who withdrawn prior to their initial assessment opportunity, active student filtering guarantees that attention is directed towards students who might profit from intervention. Consequently, 27,725 students in total match these filters.

3.3.3 Stage 3: Extracting Behavioural Features

Up until the initial evaluation, this stage shows how three fundamental behavioural dimensions (VLE interaction, academic performance, and course load), were extracted. In order to determine the total and average clicks per student in relation to a course-presentation baseline, VLE interaction analysis analyses 2,587,468 clicks that took place prior to the first assessment deadlines. Academic performance extraction uses a merit threshold analysis (threshold 70) and records the initial assessment results. Students taking

multiple concurrent courses are identified through course load quantification, which also provides context for their academic capacity. For this data, the maximum number of concurrent courses is 2; it was not found students with three or more courses at the same presentation time.

3.3.4 Stage 4: Creating Indicator

The fourth step filters the $27,725$ students and converts continuous behavioural measures into interpretable binary indicators. Merit-level performance (≥ 70 points) is attained by $16,687$ students (60.2% of the total) according to the academic excellence classification ($excellent\_Score$). $9,870$ students (35.6%) whose platform engagement surpasses course-specific averages are flagged by the VLE activity classification ($active\_in\_VLE$). Additionally, a course load analysis ($courses\_per\_term$) was added to quantify the academic workload of $1,270$ students (4.6%) who are managing multiple concurrent courses.

3.3.5 Stage 5: Aggregating an Engagement Indicator

In stage five, various pathways to academic success are captured by combining individual indicators into composite measures. By using logical OR operations to combine academic excellence and VLE activity, the unified engagement feature ($student\_engagement$) finds $19,932$ students (71.9%) who are either performing well or showing strong platform engagement. This composite approach acknowledges that various engagement patterns may lead to students' success.

3.3.6 Stage 6: Engineering the Final Training Dataset

The last step of data preparation creates a validated, ML-ready dataset of $27,725 students$ and 24 features by combining preprocessed data with engineered behavioural and demographic features. The distribution of classes in this dataset is 55.5% Pass, 25.4% Fail, and 19.1% Withdrawn. With the withdrawn students representing a crucial minority class, it draws attention to the problem of class imbalance. Now that the dataset has been cleaned and optimised, it is ready for the next round of ML pipelines, where specific techniques will be used to predict outcomes for this important minority group.

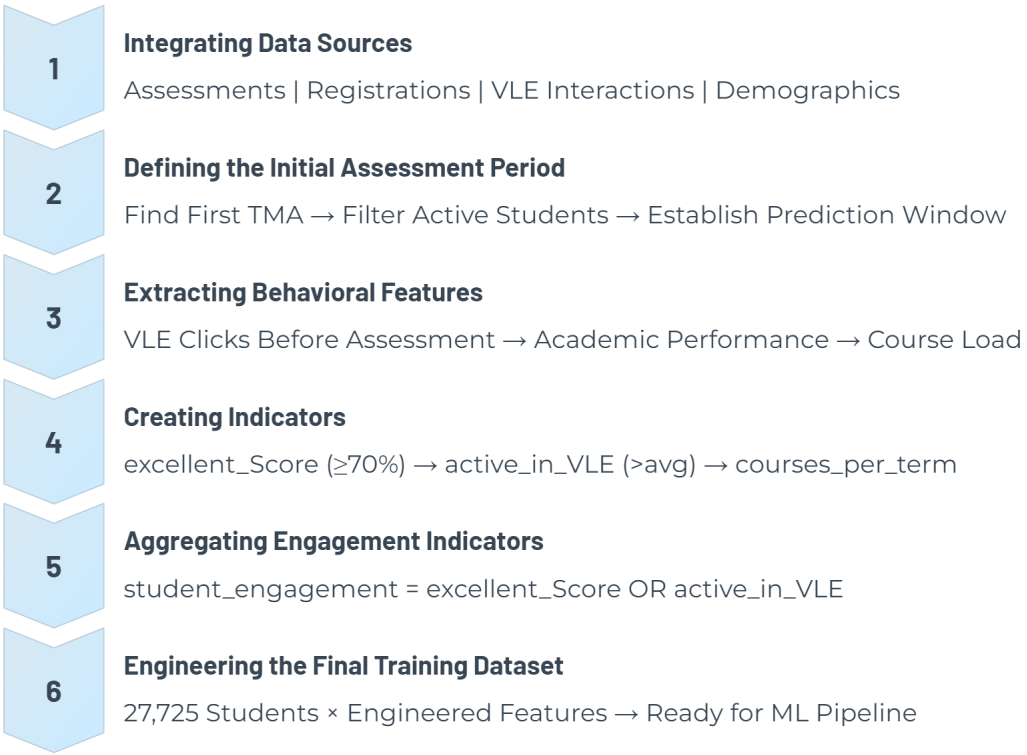

Figure 3.2: The Six-Stage Engineering Feature Pipeline

3.4 Data Stratification and Encoding Strategy

3.4.1 Stratification Analysis and Strategy Selection

Three stratification techniques are evaluated methodically to maximise the preservation of class distribution. Using course-presentation combinations, Strategy 1 uses basic cohort-based stratification. Using a combination of cohort and outcome stratification, Strategy 2 maintains the distributions of academic outcomes and demographics. Triple stratification using cohort+outcome+courseload variables is attempted in Strategy 3.

Strategy Selection: Comparative analysis shows that Strategy 2 (Cohort + Outcome) is the best course of action. Since many cohort-outcome-courseload combinations contain only one student, triple stratification (Strategy 3) fails because combined groupings have insufficient sample sizes. As shown in Table 3.1, Strategy 2 effectively preserves the crucial 19.1% dropout representation across train-test splits while maintaining class proportions.

Stratified Sampling Implementation: The chosen strategy guarantees equitable representation for both academic results and course cohorts. The train-test split allocation maintains proportional class representation by using a $80\% - 20\%$ distribution. This approach ensures sufficient samples for minority class optimisation in later model training stages while maintaining demographic diversity.

Strategy	**Withdrawn (Class 0)**	**Fail (Class 1)**	**Pass (Class 2)**	**Status**
Original Dataset	19.1%	25.4%	55.5%	Baseline
Strategy 1: Cohort Only	19.3%	25.0%	55.7%	Acceptable
Strategy 2: Cohort + Outcome	19.1%	25.4%	55.5%	**Selected**
Strategy 3: Cohort + Outcome + Courseload	-	-	-	Failed

Table 3.1: Class Distribution Comparison Across Stratification Strategies

3.4.2 Categorical Encoding Methodology

Using the *encoding_utils* module, tailored encoding techniques handle various categorical variable types. Region variables are treated as nominal categories with no intrinsic ordering through one-hot encoding. Ordinal encoding is used to preserve hierarchical relationships across socioeconomic and demographic dimensions while maintaining ranked sequences for the variables of education, age, and IMD band. For dichotomous classification, disability status uses binary encoding. Maintaining feature alignment and preventing data leakage are achieved by encoding consistency across train-test splits. Table 3.2 provides an overview of this.

Dataset Export Systematic model comparison is made possible by encoded datasets exporting as standardised CSV files (*X_train_encoded.csv*, *X_test_encoded.csv*, *y_train.csv*, *y_test.csv*). The encoding procedure preserves the temporal validity set in earlier preprocessing stages while generating consistent feature matrices appropriate for a range of algorithm requirements.

Variable Type	**Variables**	**Encoding Method**
Nominal	Region	One-Hot Encoding
Ordinal	Education, Age, IMD Band	Ordinal Encoding
Binary	Disability	Binary Encoding

Table 3.2: Encoding Strategy Matrix

3.5 Class Weighting Strategy

Class imbalance in educational datasets, where withdrawn students make up only 19.1% of the dataset population, requires strategic weighting methods that are implemented through a two-stage calculation

process that combines custom multipliers with sklearn's balanced weighting. First, balanced weights are calculated using sklearn's *compute_class_weight*('*balanced*') function to produce baseline values proportional to inverse class frequencies. Next, strategic multiplier application is applied to improve dropout detection capabilities. This is demonstrated in Algorithm 3.1, where the multiplier values (1.2x, 1.0x, 0.8x) were chosen to reduce majority class dominance by 20% and increase dropout class sensitivity by 20%. With this setup, the class balance is changed from the initial balanced weights $\{0 : 1.74, 1 : 1.31, 2 : 0.60\}$ to the optimised custom weights $\{0 : 2.09, 1 : 1.31, 2 : 0.48\}$.

Input: *y_train*, *classes*
Output: Custom class weights
balanced_weights ← *compute_class_weight*('*balanced*', *classes*, *y_train*)
custom_weights ← {
    0 : *balanced_weights*[0] × 1.2 `// Dropout class boost`
    1 : *balanced_weights*[1] × 1.0 `// Fail class unchanged`
    2 : *balanced_weights*[2] × 0.8 `// Pass class reduction`
}
Resulting weights: $\{0 : 2.09, 1 : 1.31, 2 : 0.48\}$

Algorithm 3.1: Computation of custom class weights for ML training.

3.5.1 Class Weight implementation

Three different categories based on native support capabilities are revealed by class weight implementation across six chosen algorithms, as shown in Table 3.3. Through built-in *class_weight* parameters that automatically modify loss functions and sample importance during training, RF, LG, LightGBM, and SVM offer direct implementation. Because of its distance-based architecture, KNN has limited implementation capabilities and does not support native class weights. Custom weights can be declared, but they are not successfully integrated during prediction. With no alternative strategies like weighted sampling or custom loss functions implemented in the examined notebooks, NN ($MLPClassifier$) is still unsupported and expressly lacks class weight functionality.

Algorithm	Native Support	Implementation Method	Usage Status
Random Forest	✓	`class_weight` parameter	**Active**
Logistic Regression	✓	`class_weight` parameter	**Active**
LightGBM	✓	`class_weight` parameter	**Active**
Support Vector Machine	✓	`class_weight` parameter	**Active**
K-Nearest Neighbors	×	Use another type of weighting	**Not applied**
Neural Networks	×	Not supported by MLPClassifier	**Not applied**

Table 3.3: Class Weight Support by Algorithm

3.6 ML Model Selection and Optimization

Six ML algorithms were compared methodically and rigorously optimised for student dropout prediction during the model selection phase.

3.6.1 Algorithm Selection and Optimization

The following six algorithms are complementary in their ability to handle the complexities of educational data mining: RF for robustness to feature interactions, Multinomial LR for interpretable probabilistic outputs, KNN for local behavioural patterns, LightGBM for advanced gradient boosting, SVM for non-linear boundary detection, and NN for deep learning capabilities. In order to address the crucial importance of accurately identifying at-risk students, each algorithm underwent systematic hyperparameter optimisation using GridSearchCV with 5-fold cross-validation, prioritising dropout recall performance ($\geq$ 60%) while maintaining balanced accuracy through custom class weights: dropout class weighted 2.5x, fail class 1.5x, and pass class 0.6x. Each ML model's hyperparameter search space is displayed in Table 3.4. The hyperparameter search encompassed 2,032 total parameter combinations using parallel processing with 4 CPU cores.

Algorithm	Key Parameters	Search Range	Search Space Size
RF	n_estimators	100–500	480 combinations
	max_depth	10–25	
	min_samples_split	2–10	
	class_weight	balanced / custom	
LR	C (regularization)	0.01–100	450 combinations
	penalty	L1 / L2 / ElasticNet	
	solver	SAGA / liblinear	
	max_iter	1000–5000	
KNN	n_neighbors	3–21	90 combinations
	metric	euclidean / manhattan / minkowski	
	weights	uniform / distance	
	algorithm	ball_tree / kd_tree / brute	
LightGBM	n_estimators	100–500	512 combinations
	learning_rate	0.01–0.2	
	max_depth	3–8	
	subsample	0.7–1.0	
SVM	C	0.01–100	180 combinations
	kernel	linear / RBF / polynomial	
	gamma	scale / auto / numeric	
	degree	2–4 (poly only)	
NN	hidden_layer_sizes	(50)–(150) + multilayer	320 combinations
	alpha (L2 penalty)	0.0001–1.0	
	learning_rate_init	0.0001–0.1	
	solver	adam / lbfgs	

Table 3.4: Hyperparameter search spaces and search sizes for each algorithm.

Chapter 4

Critical Evaluation

A topic-specific chapter, roughly 30% of the total page-count

This chapter is intended to evaluate what you did. The content is highly topic-specific, but for many projects will have flavours of the following:

1. functional testing, including analysis and explanation of failure cases,
2. behavioural testing, often including analysis of any results that draw some form of conclusion wrt. the aims and objectives, and
3. evaluation of options and decisions within the project, and/or a comparison with alternatives.

This chapter often acts to differentiate project quality: even if the work completed is of a high technical quality, critical yet objective evaluation and comparison of the outcomes is crucial. In essence, the reader wants to learn something, so the worst examples amount to simple statements of fact (e.g., "graph X shows the result is Y"); the best examples are analytical and exploratory (e.g., "graph X shows the result is Y, which means Z; this contradicts [1], which may be because I use a different assumption"). As such, both positive *and* negative outcomes are valid *if* presented in a suitable manner.

Chapter 5

Conclusion

A compulsory chapter, roughly 10% of the total page-count

The concluding chapter(s) of a dissertation are often underutilized because they're too often left too close to the deadline: it is important to allocate enough time and attention to closing off the story, the narrative, of your thesis.

Again, there is no single correct way of closing a thesis.

One good way of doing this is to have a single chapter consisting of three parts:

1. (Re)summarise the main contributions and achievements, in essence summing up the content.
2. Clearly state the current project status (e.g., "X is working, Y is not") and evaluate what has been achieved with respect to the initial aims and objectives (e.g., "I completed aim X outlined previously, the evidence for this is within Chapter Y"). There is no problem including aims which were not completed, but it is important to evaluate and/or justify why this is the case.
3. Outline any open problems or future plans. Rather than treat this only as an exercise in what you *could* have done given more time, try to focus on any unexplored options or interesting outcomes (e.g., "my experiment for X gave counter-intuitive results, this could be because Y and would form an interesting area for further study" or "users found feature Z of my software difficult to use, which is obvious in hindsight but not during at design stage; to resolve this, I could clearly apply the technique of Bloggs *et al.*.

Alternatively, you might want to divide this content into two chapters: a penultimate chapter with a title such as "Further Work" and then a final chapter "Conclusions". Again, there is no hard and fast rule, we trust you to make the right decision.

And this, the final paragraph of this thesis template, is just a bunch of citations, added to show how to generate a BibTeX bibliography. Sources that have been randomly chosen to be cited here include:

Bibliography

[1] K. Jordan. Massive open online course completion rates revisited: Assessment, length and attrition. *The International Review of Research in Open and Distributed Learning*, 16(3), 2015.

[2] O. Simpson. '22% -can we do better?'-The CWP Retention Literature Review. *ResearchGate*, 2010.

[3] J. Kuzilek, M. Hlosta, and Z. Zdrahal. Open University Learning Analytics dataset. *Scientific Data*, 4(1):170171, 2017.

[4] M. Hussain, W. Zhu, W. Zhang, and S. M. R. Abidi. Student Engagement Predictions in an e-Learning System and Their Impact on Student Course Assessment Scores. *Computational Intelligence and Neuroscience*, 2018(1):6347186, 2018.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, first edition, 1992.

[6] R. S. Baker and P. S. Inventado. Educational Data Mining and Learning Analytics. In J. A. Larusson and B. White, editors, *Learning Analytics: From Research to Practice*, pages 61–75. Springer, 2014.

[7] T. Le Quy, A. Roy, V. Iosifidis, W. Zhang, and E. Ntoutsi. A survey on datasets for fairness-aware machine learning. *WIREs Data Mining and Knowledge Discovery*, 12(3):e1452, 2022.

[8] J. A. Martínez-Carrascal, M. Hlosta, and T. Sancho-Vinuesa. Using survival analysis to identify populations of learners at risk of withdrawal: Conceptualization and impact of demographics. *The International Review of Research in Open and Distributed Learning*, 24(1):1–21, 2023.

[9] E. R. Kahu. Framing student engagement in higher education. *Studies in Higher Education*, 38(5):758–773, 2013.

[10] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[11] F. E. Harrell. *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis*. Springer International Publishing, second edition, 2015.

[12] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

[13] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[15] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[16] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[17] N. Tomasevic, N. Gvozdenovic, and S. Vranes. An overview and comparison of supervised data mining techniques for student exam performance prediction. *Computers & Education*, 143:103676, 2020.

[18] C. Romero and S. Ventura. Data mining in education. *WIREs Data Mining and Knowledge Discovery*, 3(1):12–27, 2013.

[19] D. Gasevic, S. Dawson, T. Rogers, and D. Gasevic. Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. *The Internet and Higher Education*, 28:68–84, 2016.

[20] O. Viberg, M. Hatakka, O. Bälter, and A. Mavroudi. The current landscape of learning analytics in higher education. *Computers in Human Behavior*, 89:98–110, 2018.

[21] R. Conijn, C. Snijders, A. Kleingeld, and U. Matzat. Predicting student performance from LMS data: A comparison of 17 blended courses using moodle LMS. *IEEE Transactions on Learning Technologies*, 10(1):17–29, 2017.

[22] N. Sclater, A. Peasgood, and J. Mullan. Learning analytics in higher education: A review of UK and international practice, 2016.

[23] M. Hlosta, D. Herrmannova, L. Vachova, J. Kuzilek, Z. Zdrahal, and A. Wolff. Modelling student online behaviour in a virtual learning environment, 2018.

[24] A. A. Nafea, M. Mishlish, A. M. Haban Shaban, M. M. Al-Ani, K. M. A. Alheeti, and H. J. Mohammed. Enhancing student's performance classification using ensemble modeling. *Iraqi Journal for Computer Science and Mathematics*, 4(4), 2023.

[25] L. P. Macfadyen and S. Dawson. Mining lms data to develop an early warning system for educators: A proof of concept. *Computers & Education*, 54(2):588–599, 2010.

[26] The open university grading, n.d. Available at: https://www.scholaro.com/db/Countries/United-Kingdom/Grading-System/The-Open-University-11929.

[27] Y. Liu, S. Fan, S. Xu, A. Sajjanhar, S. Yeom, and Y. Wei. Predicting student performance using clickstream data and machine learning. *Education Sciences*, 13(1):17, 2023.

[28] R. Baker, D. Xu, J. Park, R. Yu, Q. Li, B. Cung, C. Fischer, F. Rodriguez, M. Warschauer, and P. Smyth. The benefits and caveats of using clickstream data to understand student self-regulatory behaviors: Opening the black box of learning processes. *International Journal of Educational Technology in Higher Education*, 17(1):13, 2020.

[29] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009.

[30] S. Kaufman, S. Rosset, C. Perlich, and O. Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data*, 6(4):15:1–15:21, 2012.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[32] Encoding of categorical variables — scikit-learn course, n.d. Available at: https://inria.github.io/scikit-learn-mooc/python_scripts/03_categorical_pipeline.html.

[33] A. Fernandez, S. Garcia, M. Galar, R. C. Prati, B. , and F. Herrera. *Learning from Imbalanced Data Sets*. Springer International Publishing, 2018.

[34] X.-Y. Liu, J. Wu, and Z.-H. Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2009.

[35] B. Krawczyk. Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[36] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing. Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220–239, 2017.

[37] A. Luque, A. Carrasco, A. Martin, and A. de las Heras. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. *Pattern Recognition*, 91:216–231, 2019.

[38] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):463–484, 2012.

[39] M. Grandini, E. Bagli, and G. Visani. Metrics for multi-class classification: An overview, 2020.

[40] L. A. Jeni, J. F. Cohn, and F. De La Torre. Facing imbalanced data: Recommendations for the use of performance metrics. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 245–251, 2013.

[41] T. Tantisripreecha and N. Soonthornphisaj. A novel term weighting scheme for imbalanced text classification. *Informatica*, 46(2), 2022.

[42] J. Opitz. A closer look at classification evaluation metrics and a critical reflection of common evaluation practice. *Transactions of the Association for Computational Linguistics*, 12:820–836, 2024.

[43] P. Akor, G. Enemali, U. Muhammad, R. R. Singh, and H. Larijani. Hierarchical deep learning for comprehensive epileptic seizure analysis: From detection to fine-grained classification. *Information*, 16(7):532, 2025.

[44] M. Orooji and J. Chen. Predicting louisiana public high school dropout through imbalanced learning techniques, 2019.

[45] S. Lee and J. Y. Chung. The machine learning-based dropout early warning system for improving the performance of dropout prediction. *Applied Sciences*, 9(15):3093, 2019.

[46] M. Hlosta, Z. Zdrahal, and J. Zendulka. Ouroboros: Early identification of at-risk students without models based on legacy data. In *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, pages 6–15. ACM, 2017.

[47] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[48] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the roc plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10(3):e0118432, 2015.

[49] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[50] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[51] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 20(2):215–242, 1958.

[52] J. A. Nelder and R. W. M. Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, 135(3):370–384, 1972.

[53] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[55] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[56] R. Kohavi. A study of cross validation and bootstrap for accuracy estimation and model selection. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1137–1145, 1995.

[57] S. Varma and R. Simon. Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, 7(1):91, 2006.

[58] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[59] L. Prechelt. Early stopping—but when? In G. Montavon, G. B. Orr, and K.-R. Muller, editors, *Neural Networks: Tricks of the Trade, Second Edition*, pages 53–67. Springer, 2012.

[60] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, page 78. ACM, 2004.

[61] A. Krogh and J. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4. Morgan-Kaufmann, 1991.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the examiners are not obliged to read such appendices.