# Step 1: Overview and Brief Analysis.

Begin by loading the dataset and reviewing all available features/columns to get a general sense of the data. Perform light exploratory data analysis (EDA) to identify key characteristics, such as feature distributions, potential outliers, and initial patterns across features.

# Step 2: Define the Project Objective and Hypothesis.

Clearly establish the objective of the project. is it a classification or regression problem? For instance, are you trying to predict whether a student will drop out (binary classification: Yes = 1, No = 0), or estimate a probability of dropout?

# Step 3: Data Cleaning and Preprocessing.

Once the problem type is defined, proceed with cleaning and preparing the dataset. This includes handling missing values, removing duplicates, and standardising formats. During this process, explore the data further to understand feature distributions (e.g., gender, age group, final result) and identify class imbalance. This step also helps inform decisions around scaling, encoding, and any adjustments needed to ensure the model isn’t biased by disproportionate feature values.

# Step 4: Feature Engineering.

After cleaning and preprocessing, the next step is feature engineering, which includes merging various features and datasets into a single structured format suitable for modelling.

For example, consider the following time-series dataset related to student assessments:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Course | id\_student | ass\_id | ass\_type | weight | score | date\_due | date\_submitted | final\_result |
| AAA | 0 | 0 | TMA | 20 | 35 | 5 | 6 | Pass |
| AAA | 0 | 1 | TMA | 20 | 60 | 20 | 19 | Pass |
| AAA | 0 | 2 | CMA | 20 | 75 | 50 | 55 | Pass |
| AAA | 0 | 3 | Exam | 40 | 85 | 100 | 100 | Pass |

In this example, final\_result is the target variable, representing whether a student passed, failed, withdrew, or achieved distinction. According to the grading rules:

* A score < 40 is a fail (including when TMA, CMA, and exam scores are combined).
* 40–69 is a pass.
* 70 or more is a distinction.

Because the data is time-dependent, we need to aggregate it to make it usable for modelling. For instance, we can summarise a student’s performance as:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **course** | **id\_student** | **weighted\_score** | **late\_rate** | **fail\_rate** | **final\_result** |
| AAA | 0 | 68 | 0.50 | 0.25 | Pass |

Here's how the features are engineered:

* **weighted\_score** is calculated by multiplying each score with its corresponding weight and dividing by the total weight (100 in this example).
* **late\_rate** is derived by comparing date\_submitted with date\_due. If a submission is late (i.e., date\_submitted > date\_due), it's counted. The late rate is the proportion of late submissions out of total submissions (range 0 to 1).
* **fail\_rate** is computed by counting how many assessments have a score below 40 (or missing scores, treated as non-submissions/fails) and dividing by the total number of assessments.

In the example above, the student failed one assessment (ass\_id = 0 with a score of 35), resulting in a fail rate of 0.25.

## Time-Based Feature Limiting for Early Prediction

In real-world applications, we rarely have access to a student's full course history when trying to predict dropout early. Predicting dropout at the end of a course is often too late to intervene effectively.

Therefore, we introduce the idea of timeline-based feature limitation. This means only including data up to a certain point in time (e.g., mid-module) to simulate early prediction. For example, if the total module duration is 100 days, we can truncate the dataset to only include events up to day 50 (midpoint). The aggregated data then looks like the following table:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **course** | **id\_student** | **weighted\_score** | **late\_rate** | **fail\_rate** | **final\_result** |
| AAA | 0 | 57 | 0.67 | 0.33 | Pass |

Here we can see that compared to the overall student performance (covering all the 100 days of the module). The 50% point of the module shows a higher late and fail rate as it has not covered the final exam yet as it did not happen yet. This is an example of what data would be precessed and what can ML models learn from this for thousands of students and identify trends on what indicators indicate student dropout.

This truncated snapshot shows poorer performance, higher late and fail rates, since it does not yet include the final exam. Such intermediate datasets allow us to train models to identify early warning signs of potential dropout, such as high failure rates, frequent late submissions, or poor engagement early in the course.

By simulating limited-time views for thousands of students, ML models can learn early behavioural patterns that are predictive of eventual dropout.

# Step 5: Processing Dataset

After completing feature engineering and merging the relevant tables, we proceed with final data cleaning. This involves removing irrelevant features, handling missing values, and ensuring the dataset is suitable for predictive modelling.

One such feature, "Date Unregistration", is excluded because it is not useful for early dropout prediction. For instance, when predicting dropout at the halfway point of a module, we wouldn’t yet know the unregistration date for a student who hasn't withdrawn. Including this feature would introduce data leakage, making the model unrealistically optimistic.

## Imputation Strategy for Missing Values

Missing values in various features are handled through context-aware imputation. The following table summarises the imputation methods and the rationale behind them:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Imputation Type** | **Justification** |
| date\_registration | Replace missing values with the median. | The majority of null values correspond to withdrawn students. To impute these missing values, we can use the median date as a substitute for the absent entries. |
| total\_clicks | Replace missing values with 0. | Students who failed or withdrew typically show no interaction with the VLE. Since they have no recorded VLE activity, missing values reflect zero engagement. |
| banked\_rate | Most missing banked rates correspond to failing or withdrawn students. Given the low proportion of known data and its limited predictive power, a default of 0 is a reasonable assumption. |
| weighted\_score | |  | | --- | |  |  |  | | --- | | Students who failed or withdrew often have unsubmitted assessments, and thus no recorded scores. According to the data specification, non-submissions are treated as missing results, and a score of 0 is appropriate. | |
| late\_rate | Students with no submission records are assumed to have submitted all assessments late (or not at all). Hence, a late rate of 1 (i.e., 100%) reflects complete non-engagement. |
| fail rate | Similar to late rate, the absence of submissions is interpreted as failure to complete assessments. A fail rate of 1 reflects full assessment failure. |
| imd\_band | Bayesian Ridge Regression | This socio-demographic feature is important for dropout risk modelling. Instead of using basic imputation methods (mean, median, mode), we apply Bayesian Ridge Regression to predict missing IMD values based on other related features such as age, region, and education. This produces more accurate, context-driven imputations. |

## Excluding Early Withdrawals

We also removed 4,609 students who withdrew before the module started or within the first 19 days. This threshold was selected because most modules begin assessments after day 19, meaning these students typically show no VLE activity or assessment records. Including them would add noise without providing any useful insight. Moreover, early withdrawal data would not be available when predicting dropout during the early or mid-phase of the module, so keeping such records would compromise the realism and applicability of the model.

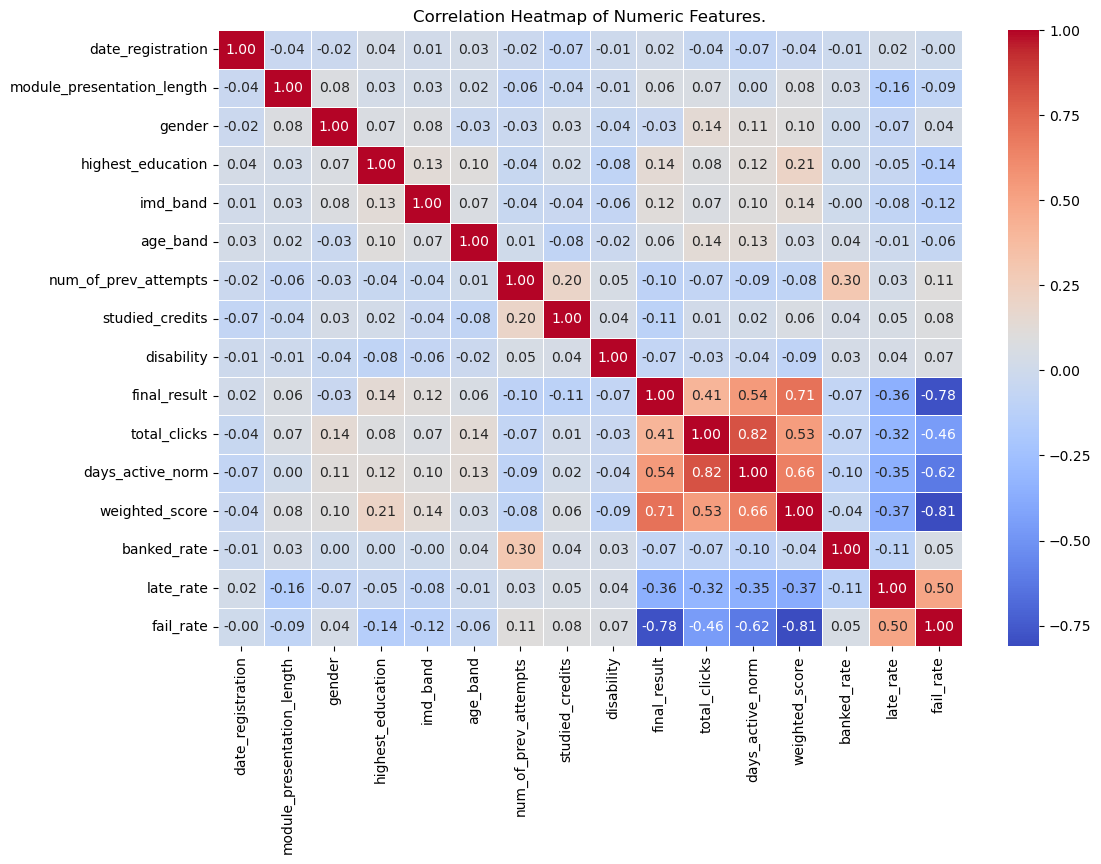
# Step 6: Train/Test Split

Before conducting any detailed EDA, it’s crucial to split the dataset to avoid inadvertently gaining insights from the test data, a form of data leakage. We apply an 80/20 train-test split, ensuring that the training set is used to build the model (also using the training set for final EDA) and the test set remains unseen until final evaluation. To maintain a balanced representation of different modules across both sets, we stratify the split by course module. This ensures that each course is proportionally represented in both the training and testing subsets.

# Step 7: Final EDA

With the dataset properly split, we then perform a more detailed EDA on the training set. This helps uncover insights from the engineered features, explore value distributions, and assess feature relevance. We examine relationships between features, identify outliers, and use tools such as correlation matrices to evaluate how strongly features are associated with each other and the target variable.

This analysis also informs which features may need scaling or encoding to improve model performance. For instance, continuous variables with wide ranges may require standardisation, while categorical variables will need encoding. After EDA, we also drop irrelevant or redundant features, such as “id\_student”, which are not useful for modelling.



# Step 8: Scaling and Encoding Dataset

Using the EDA insights from the previous step we can determine which steps need to be encoded, scaled or transformed.

|  |  |
| --- | --- |
| **Features** | **Scaling/Encoding Type** |
| code\_module | One-Hot Encoder |
| code\_presentation | One-Hot Encoder |
| date\_registration | Standard Scalar |
| module\_presentation\_length | Standard Scalar |
| gender | One-Hot Encoder |
| region | One-Hot Encoder |
| highest\_education | Standard Scalar |
| imd\_band | Standard Scalar |
| age\_band | Standard Encoder and Standard Scalar |
| num\_of\_prev\_attempts | Standard Scaler |
| studied\_credits | Standard Scalar |
| days\_active\_norm | Standard Scalar |
| disability | One-Hot Encoder |
| final\_result (target feature) | Distinction, Pass and Fail = 1  Withdrawn = 0 |
| total\_clicks | Standard Scalar |
| weighted\_score | Standard Scalar |
| banked\_rate | Standard Scalar |
| late Rate | Standard Scalar |
| fail Rate | Standard Scalar |

We apply One-Hot Encoding to categorical variables such as gender and disability, converting each category into a separate binary column. This is necessary because most machine learning models, such as logistic regression and SVM, require numerical inputs and cannot directly process string-based categorical data.

Standard Scaling is applied to continuous numerical features to normalise their values, ensuring they all have a mean of 0 and a standard deviation of 1. This is especially important for models that are sensitive to feature scales, like MLP, LR, SVM, or any algorithm using gradient-based optimisation. For example, “studied\_credits” can range from 30 to 600, while “num\_of\_prev\_attempts” ranges from 0 to 5. Without scaling, the larger-valued features would dominate the learning process.

Some features, such as “age\_band”, are ordinal categorical; they have a meaningful order but are not inherently numeric. These are first encoded using integer labels (e.g., "0-35" → 0, "35-55" → 1, "55<=" → 2), and then scaled using StandardScaler to ensure consistent treatment alongside continuous variables.

Finally, the “final\_result” is transformed into a binary classification label: students who passed, failed, or achieved distinction are assigned a 1 (i.e., they completed the module), while students who withdrew are assigned a 0. This aligns with our classification goal of predicting dropout versus continuation.

# Step 9: Machine Learning Models

As this is a classification task, we employ Logistic Regression (LR), Support Vector Machine (SVM), Multilayer Perceptron Classifier (MLP), and Random Forest (RF) models. These models are trained on the training dataset, then evaluated on the test set using two key metrics: accuracy and macro-averaged F1 score, which balances precision and recall across all classes.

## Model Selection Justification

The models chosen for this task, LR, SVM, MLP Classifier, and Random Forest, were selected to cover a diverse range of classification approaches, each offering distinct advantages relevant to student dropout prediction:

**Logistic Regression:** LR serves as a strong baseline model due to its simplicity, interpretability, and efficiency. It helps establish a benchmark for classification performance and enables us to observe the effect of linear separability on dropout prediction. Its probabilistic output is also useful in threshold-based decision-making for identifying at-risk students.

**SVMs:** SVM is well-suited for high-dimensional datasets and often performs well with imbalanced data, especially when combined with class weights or kernel tricks. It provides a robust non-linear decision boundary (using RBF or polynomial kernels), allowing it to capture more complex relationships in student behaviour patterns.

**MLP Classifier:** MLP, a basic form of a neural network, is included to model non-linear patterns in the data that simpler models like Logistic Regression might miss. It is especially valuable when interactions between features are not explicitly represented. While it requires more tuning, MLP can be powerful for learning latent structures in sequential or categorical student activity data.

**Random Forest:** RF is a robust ensemble method that effectively handles missing values, categorical variables, and non-linearity. It also provides feature importance scores, which can offer valuable insights into which student behaviours or demographics are most indicative of dropout risk. Additionally, it is resilient to overfitting due to averaging across multiple decision trees.

## Why a Diverse Set of Models?

By selecting models from different algorithmic families: linear (LR), kernel-based (SVM), neural (MLP), and ensemble tree-based (RF), we ensure broad coverage of learning mechanisms. This helps:

* Assess how linear vs non-linear models perform on the task,
* Understand model robustness under different data regimes (early vs full),
* Identify trade-offs in performance and interpretability,
* Compare how well models generalise to unseen data.

Given our primary goal of accurately identifying withdrawn students, greater emphasis is placed on the precision and especially the recall of the dropout class. The model must correctly flag as many students at risk of dropping out as possible so that early interventions and support measures can be deployed to help them stay in their courses.

Each model is optimised using GridSearchCV for hyperparameter tuning, coupled with 5-fold cross-validation to ensure more robust and reliable performance estimates.

Moreover, we test each model on four different versions of the dataset: early, midpoint, late, and full, to assess how predictive power varies with data availability over time. While all versions are evaluated, the early and midpoint phases are our primary focus, as they represent the windows where intervention is most impactful. The late and full versions serve as benchmarks for comparison.