Binghamton University
Department of Computer Science

CS 435/535: Introduction to Data Mining
Data Mining Course Project Report

STUDENT PERFORMANCE PREDICTION AND LEARNING BEHAVIOR ANALYSIS USING MACHINE LEARNING & LEARNING ANALYTICS

Instructor: Yingxue Zhang
Semester: Fall 2025
University: Binghamton University

Submitted By:

- Rishika Shivanna – B01106964
- Sai Tharun Gonuguntla - B01171960
- Ilayaraja Rajmohan - B01123362
- Sanjitha Praveen - B01151243

1. Introduction and Motivation

Over the last decade, online and blended learning have become a central part of higher education. Platforms such as virtual learning environments (VLEs) generate large amounts of data about how students interact with course content, submit assignments, and progress through modules. While this data is routinely collected, it is often underused in terms of supporting at-risk students before they fail or withdraw.

A recurring challenge for universities is **early identification of students who are likely to disengage or perform poorly**. Traditional warning systems often rely on exam marks or instructor observations that arrive too late in the semester, limiting the possibility of meaningful intervention. At the same time, different students display very different learning behaviors: some study steadily, others cram at the last minute, some make a strong recovery after a weak start, while others show early promise and then burn out.

This project aims to address these issues by using data mining and machine learning on the **Open University Learning Analytics Dataset (OULAD)**. The core idea is to combine demographic information, assessment performance, and clickstream activity to:

1. **Predict whether a student will Pass/Distinction or Fail/Withdraw**, and
2. **Analyze learning behavior patterns such as binge vs. steady learning and "comeback" trajectories.**

By building a predictive model and performing behavioral analyses, the project seeks to provide actionable insights that can help educators identify at-risk students earlier and understand which patterns of engagement are associated with success or failure.

2. Problem Description

The project focuses on two closely related tasks:

1. **Predictive Task – Student Outcome Classification**
    - Formulated as a binary classification problem.
    - The original final_result categories (Pass, Distinction, Fail, Withdrawn) are mapped to:
        - 1 (success): Pass or Distinction
        - 0 (failure): Fail or Withdrawn
    - Given a set of features derived from student demographics, assessment marks and VLE activity, the model aims to predict this binary outcome for each student.
2. **Analytical Task – Learning Behavior and Pattern Discovery**
    - Beyond simply predicting outcomes, the project investigates how students learn:
        - Are they binge learners who study in short intense bursts?
        - Are they steady learners who engage consistently over time?
        - Do some students show comeback behavior, starting with low performance but improving later?
        - Are there students who burn out, starting strong and then declining?
    - These patterns are explored using both statistical analysis and frequent pattern mining (Apriori algorithm) applied to derived behavioral indicators.

Together, these tasks provide both a practical prediction tool and a richer understanding of student engagement dynamics.

3. Data Description

The notebook uses four core tables from the OULAD dataset:

1. studentInfo.csv
    - Contains one row per student per module-presentation.
    - Includes **demographic and background attributes**, for example:
        - id_student
        - code_module, code_presentation
        - gender
        - age_band
        - highest_education
        - imd_band (socio-economic band)
        - disability
        - final_result (the final outcome, used as target)
2. studentRegistration.csv
    - Describes **when students registered and if/when they unregistered** from a module.
    - Columns typically include:
        - id_student
        - code_module, code_presentation
        - date_registration
        - date_unregistration (for withdrawals)
3. studentAssessment.csv
    - Records **assessment submissions and scores**:
        - id_student
        - id_assessment

        - date_submitted
        - score
    - This table is crucial for constructing features related to performance (average score, total score, etc.).

4. studentVle.csv
    - Contains the **clickstream activity log**:
        - id_student
        - id_site
        - date
        - sum_click (number of clicks on that day/site)
    - This table allows us to quantify student engagement with the online learning environment.

These datasets are linked via id_student and the module/presentation identifiers. The combination of these tables provides a comprehensive view of each student's background, temporal engagement and performance.

4. Data Exploration

The notebook begins by loading the four CSV files and printing a **dataset summary** showing the number of rows and columns in each. Initial exploratory steps include:

- Displaying the **first few rows** of each dataset to understand the structure.
- Printing **data types** and **missing value counts** for each column.
- Inspecting the distribution of the final_result variable in studentInfo to see how many students pass, fail, withdraw, or achieve distinction.
- Examining basic descriptive statistics for numeric fields like score and sum_click.

This exploration quickly reveals several important points:

- The class distribution is imbalanced, with more students passing than failing or withdrawing. This has implications for model evaluation and the need to consider metrics beyond simple accuracy.
- The clickstream data is highly variable: some students show extremely high numbers of clicks, while others have minimal or no recorded activity.
- Missing values appear in certain columns such as date_unregistration (which is naturally absent for students who do not withdraw) and possibly in demographic fields or assessments if some information was not recorded.
- The different datasets must be carefully **merged and aligned** at the student level to create a single modeling table.

The EDA step does not just characterize the data; it also informs the design of preprocessing, feature engineering and modeling choices.

5. Data Preprocessing

The notebook performs a structured data cleaning pipeline:

1. **Duplicate Removal**
   - For each of the tables (studentInfo, studentRegistration, studentAssessment, studentVle), duplicate rows are dropped.
   - The script logs how many duplicates are removed, ensuring that each record is unique and not double counted.
2. **Handling Dates**
   - Fields such as date_registration and date_unregistration are converted into numeric formats (e.g., days relative to the start of the presentation) using pd.to_numeric with appropriate handling for invalid values.
   - This transformation is essential since many machine learning algorithms cannot work directly with raw date strings.
3. **Missing Value Treatment**
   - Missing values are handled context-dependently:
     - For numerical engagement features (like clicks), missing values often mean "no activity" and are replaced with 0.
     - For derived features such as days_enrolled, missing values may indicate that a student never unregistered; in such cases, the duration is computed assuming enrollment until the end of the course.
   - Before modeling, the feature matrix X is again checked and any remaining missing values are filled with 0 to avoid errors during scaling and training.
4. **Standardizing Categorical Values**
   - Categorical fields such as final_result may appear in different cases (Pass vs pass, etc.). The notebook's mapping logic handles both uppercase and lowercase versions when constructing the target variable.
   - Gender and other categorical attributes are encoded into numeric representations (e.g., gender_val), which are suitable for machine learning models.
5. **Merging Tables**
   - A central master_df is created by merging:
     - studentInfo and studentRegistration on id_student, code_module, and code_presentation.
     - Then merging in assessment features (aggregated per student).
     - Finally, merging in VLE features (also aggregated per student).
   - The script prints shapes and column lists after each merge step, and reports final counts of missing values and target distribution in the merged dataset.

Through these preprocessing steps, the raw datasets are transformed into a clean, consistent, and analysis-ready form.

6. Feature Engineering

Feature engineering is one of the most important parts of the notebook and adds substantial predictive power.

6.1 Assessment-Based Features

Using studentAssessment.csv, the notebook groups by id_student and computes for each student:

- avg_score – average assessment score.
- max_score and min_score – best and worst scores.
- submission_count – number of assessments submitted.
- total_score – sum of all scores across assessments.

These features capture not only the overall performance level but also the consistency and engagement with assessments.

6.2 VLE / Clickstream Features

From studentVle.csv, click events are aggregated per student to produce:

- total_clicks – total number of clicks in the VLE.
- days_active – number of distinct days on which the student generated at least one click.
- avg_clicks_per_day – total_clicks / days_active, providing a sense of intensity.
- max_clicks_single_day – the maximum clicks on any given day, capturing peak activity.
- activity_sessions or similar counts of active periods.
- click_consistency – a derived metric such as total_clicks / (days_active + 1), approximating how evenly activity is distributed.

These features reflect both quantity and pattern of engagement with the online learning platform.

6.3 Registration and Time-Based Features

From studentRegistration.csv, the notebook constructs:

- days_enrolled – approximate duration of enrollment in the module (based on registration and unregistration dates).
- early_registration – a flag or numeric feature indicating whether a student registered early relative to the course start, which may correlate with planning and motivation.
- Other derived metrics based on timeline differences, for example: how late in the course a withdrawal occurred.

6.4 Selected Feature Set for Modeling

A list of potential features is defined in code, including:

- gender_val
- avg_score, max_score, min_score, submission_count
- total_clicks, days_active, avg_clicks_per_day, max_clicks_single_day
- activity_sessions, click_consistency
- date_registration, days_enrolled, early_registration

The script then checks which of these are actually present in master_df, forming the final feature matrix X, and creates the target vector y from final_result. This feature set combines performance, engagement, and temporal behavior into a unified representation.

7. Association Rule Mining with Apriori

Before modeling, the notebook performs a pattern mining step to discover interpretable relationships between behaviors and outcomes.

7.1 Binary Tag Construction

A new DataFrame (apriori_df) is built with Boolean indicator columns such as:

- Outcome Indicators:
  - Result_Fail
  - Result_Pass
  - Result_Distinction
- Engagement Indicators:
  - High_Activity, Low_Activity (based on thresholds on total_clicks)
  - High_Score, Low_Score (based on thresholds on avg_score)
  - Additional tags like High_Assessment_Count or Low_Assessment_Count, depending on the engineered features.

7.2 Frequent Itemsets and Rules

Using the Apriori algorithm from mlxtend.frequent_patterns, the notebook:

1. Finds frequent itemsets that occur in at least a specified support (e.g., 5% of students).
2. Generates association rules using lift as the primary metric, with a minimum threshold to focus on non-trivial patterns.

The notebook then filters and highlights two broad groups of rules:

- Failure-related rules – patterns that end with Result_Fail on the right-hand side, indicating combinations of low activity, low scores, or other risk factors strongly associated with failure or withdrawal.
- Success-related rules – patterns that culminate in Result_Pass or Result_Distinction, highlighting combinations associated with high likelihood of success.

These rules are exported (e.g., to Pattern_Mining_Results.csv) and printed in a human-readable format, giving instructors interpretable explanations rather than just black-box predictions.

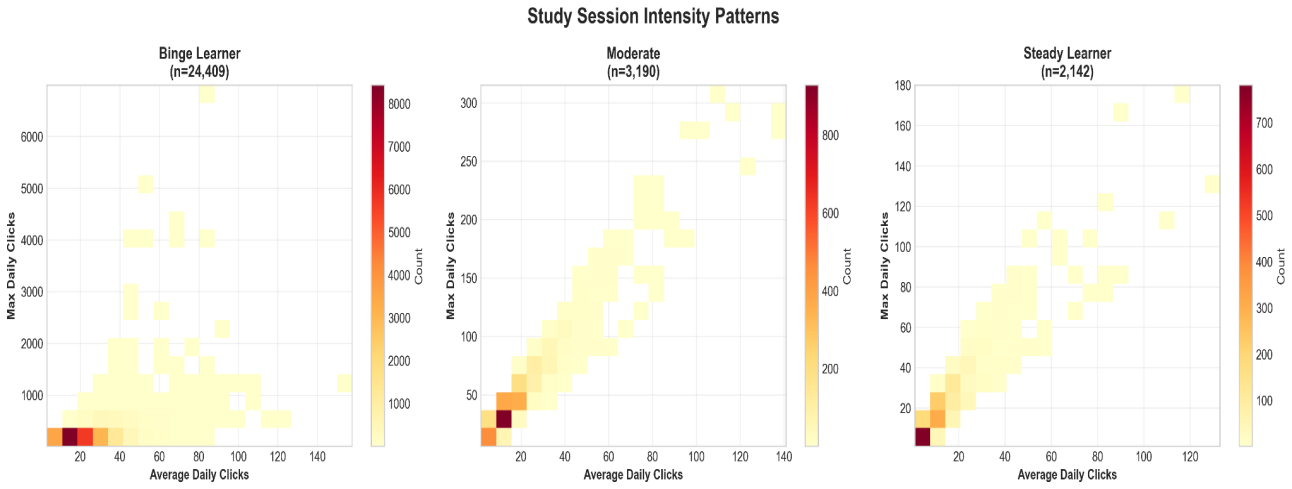
8. Learning Behavior Analysis

Beyond individual features, the notebook performs two major behavior analyses: binge vs. steady learning and comeback/burnout trajectories.

8.1 Binge Learning vs. Steady Learning

Using the detailed studentVle table (reloaded as vle_detailed), the notebook:

1. Computes daily click counts per student over time.
2. Derives summary statistics such as:
    - total_active_days
    - max_daily_clicks
    - measures of variability (e.g., coefficient of variation)
3. Defines rules to classify students into learning styles:
    - Binge Learners – relatively few active days but very high max_daily_clicks and high variability, indicating cramming behavior.
    - Steady Learners – many active days and lower variability, indicating consistent engagement.
    - Moderate Learners – those who fall between the two extremes.



These learning styles are then merged back with the outcome data from master_df to examine:

- How average scores differ by learning style.
- How Pass/Distinction rates vary between binge, steady, and moderate learners.
- Whether binge learning is an effective strategy compared to steady learning.

The notebook also applies a t-test (via scipy.stats) to statistically compare average scores between binge and steady learners, checking whether the observed difference in performance is significant.

8.2 Comeback and Burnout Analysis

To study trajectory over time, the notebook uses studentAssessment combined with an assessments metadata file:

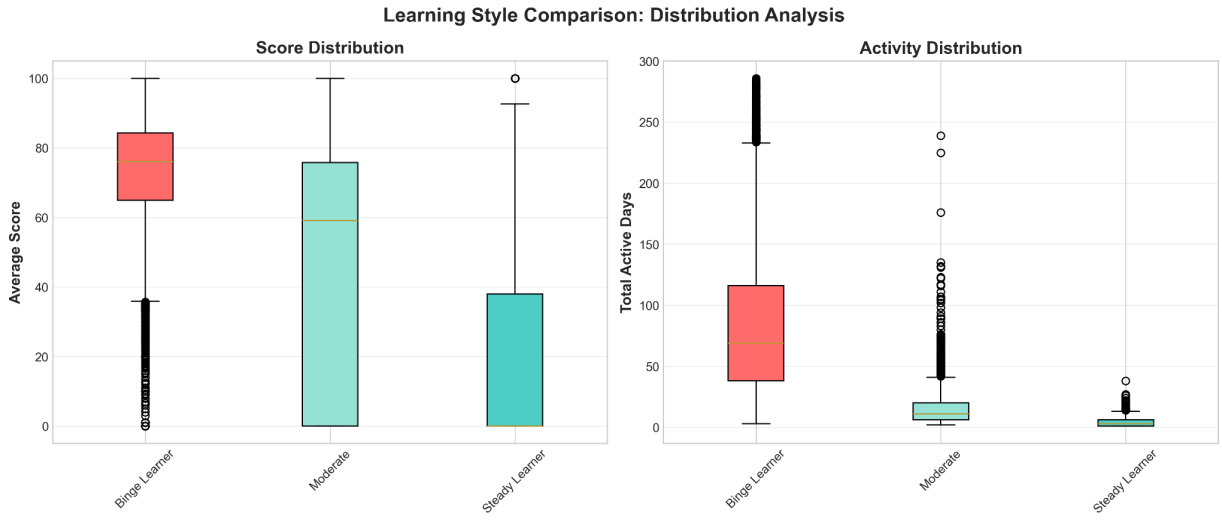
1. Assessments are split into early and late phases based on assessment dates or positions in the course.
2. For each student, the notebook computes:
    - early_score – average score in early assessments.
    - late_score – average score in later assessments.
    - improvement – difference between late_score and early_score.

Using these values, students are categorized into trajectory types:

- Comeback Kids – low early_score but strong positive improvement (e.g., improvement > 20 points).

- Burnout Students – high early_score followed by significant decline.
- Consistent Stars – high performance throughout (both early and late scores are strong).
- Strugglers – consistently low scores with little improvement.

These categories are merged with outcome labels and engagement features, allowing analysis of how trajectory types correlate with final results and activity patterns. The results are saved (e.g., Comeback_Analysis.csv) for further visualization or reporting.



9. Machine Learning Modeling

9.1 Data Preparation for Modeling

From the cleaned and enriched master_df, the notebook constructs:

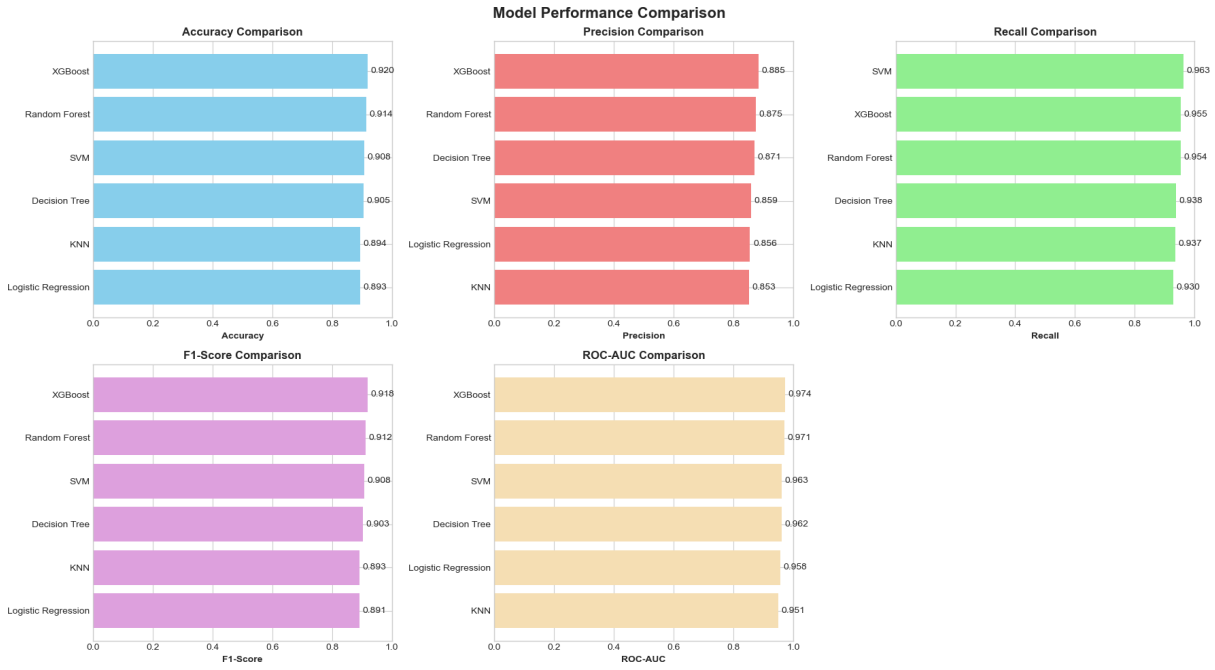
- Features (X) – selected numeric columns from the engineered feature set.
- Target (y) – the binary variable target, where 1 represents Pass/Distinction and 0 represents Fail/Withdrawn.

A stratified train–test split is then performed using train_test_split:

- Training set: 80% of students.
- Test set: 20% of students.
- Stratification ensures that the relative proportion of pass vs. fail classes is maintained in both training and test sets.

Because several models are sensitive to feature scaling, the notebook applies StandardScaler:

- X_train_scaled is fitted and transformed from X_train.
- The same scaler is used to transform X_test into X_test_scaled.



9.2 Models Trained

The notebook defines and trains a set of six widely used classification algorithms:

1. Logistic Regression
2. Decision Tree Classifier
3. Random Forest Classifier
4. Support Vector Machine (SVM)
5. K-Nearest Neighbors (KNN)
6. XGBoost Classifier

Each model is trained using the training data (scaled where appropriate) and then evaluated on the held-out test set. The notebook stores the trained models in a trained_models dictionary for later comparison and saving.

9.3 Evaluation Metrics

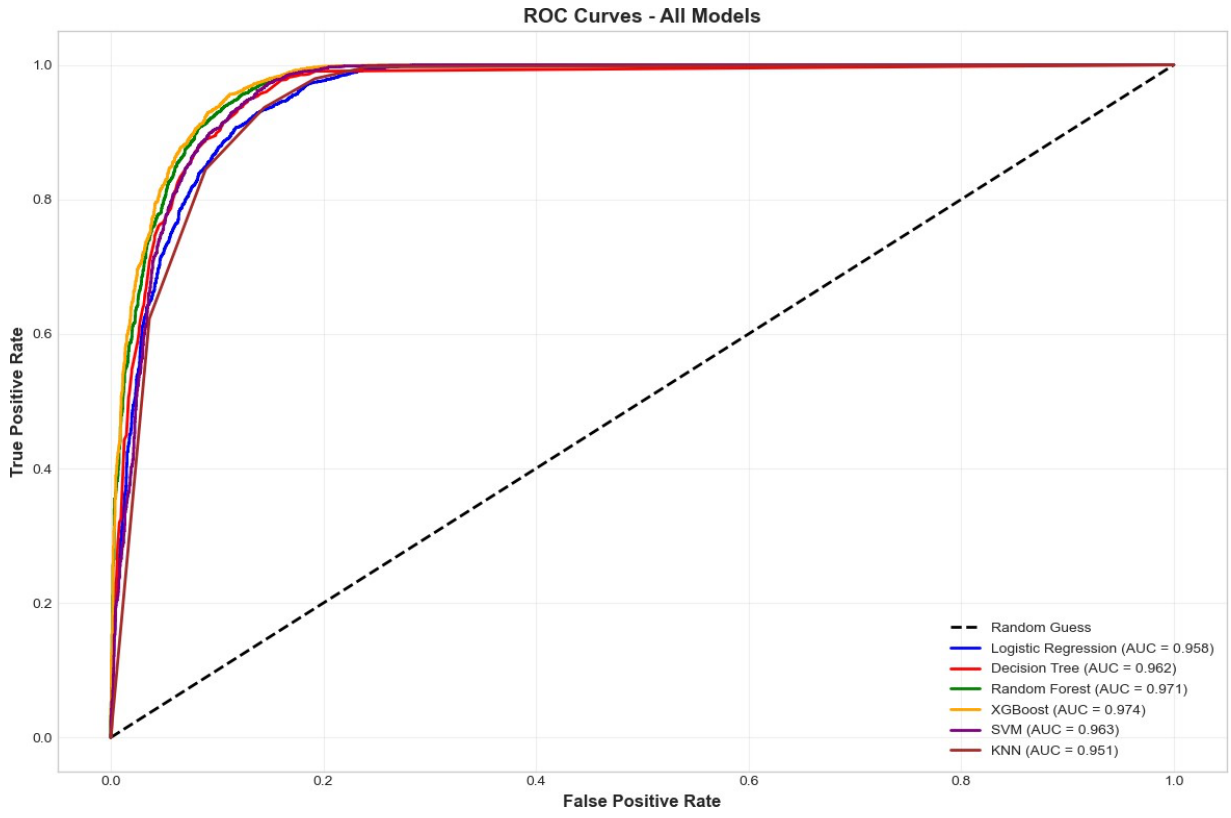
For each model, the notebook computes several performance metrics, including:

- Accuracy – proportion of correctly classified students.
- Precision – particularly important for the "at-risk" class if we consider failing students as the positive class in some analyses.
- Recall – how many of the actual failing students the model successfully identifies.
- F1-score – harmonic mean of precision and recall, balancing false positives and false negatives.
- ROC-AUC – summarizing the trade-off between true positive rate and false positive rate across thresholds.

A classification report (via classification_report) and a confusion matrix (via confusion_matrix) are generated for the best-performing model. These outputs help interpret the types of errors made:

- True Negatives – correctly predicted Fail/Withdrawn.
- False Positives – predicted Pass but actually Fail/Withdrawn (dangerous because at-risk students are missed).
- False Negatives – predicted Fail but actually Pass (unnecessary concern but safer than missing at-risk students).
- True Positives – correctly predicted Pass/Distinction.

The notebook also compares models in a summary table (e.g., model_performance_comparison.csv) and selects the best model based on a chosen metric such as F1-score or ROC-AUC. Ensemble models like Random Forest and XGBoost are typically strong candidates due to their ability to capture non-linear relationships and feature interactions.



10. Effectiveness of the Final Solution

The effectiveness of the solution is evaluated along two dimensions: predictive performance and interpretability/insight.

10.1 Predictive Performance

From the modeling step, the notebook:

- Identifies a best-performing classifier (stored as best_model_name) and saves it along with all trained models and the scaler (e.g., to files like best_model_<name>.pkl, all_trained_models.pkl, scaler.pkl, and feature_names.pkl).
- Uses the best model to generate predictions on the test set and prints a detailed classification report and confusion matrix.
- Summarizes how well the model can distinguish between at-risk and successful students.

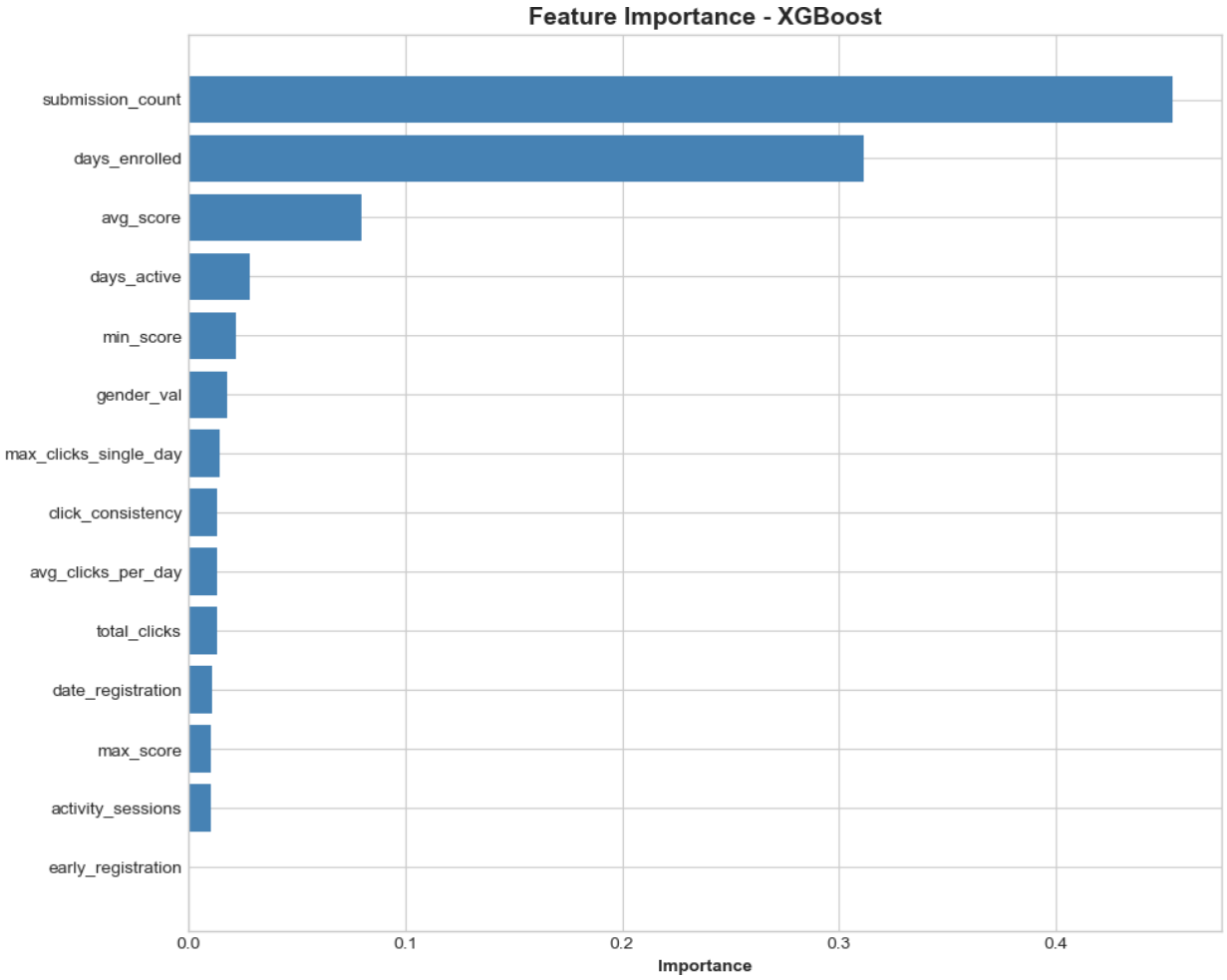
While the exact numerical values depend on the specific training run and data splits, the design of the notebook ensures that the evaluation is comprehensive and based on multiple metrics, not just accuracy. This is important because, in an imbalanced dataset, high accuracy can be misleading if the model simply predicts the majority class.

10.2 Interpretability and Behavioral Insight

Beyond numeric scores, the project provides substantial interpretability:

- Association rules derived from the Apriori algorithm explicitly show combinations of engagement and performance factors associated with success or failure.
- Binge vs. steady learning analysis reveals how different engagement styles relate to average scores and pass rates, supported by statistical tests.
- Comeback and burnout classifications show that students' trajectories over time matter, not just their average scores.

These insights are crucial in an educational context. Even if a model performs well, instructors need to understand why certain students are at risk in order to design appropriate interventions. The combination of predictive modeling and pattern analysis addresses this need.


Feature Importance - XGBoost
submission_count
days_enrolled
avg_score
days_active
min_score
gender_val
max_clicks_single_day
click_consistency
avg_clicks_per_day
total_clicks
date_registration
max_score
activity_sessions
early_registration
0.0
0.1
0.2
0.3
0.4
Importance

11. Limitations and Future Work

Although the project provides a strong foundation, several limitations and potential extensions exist:

1. **Single-Institution Dataset**
    - The OULAD data comes from one institution and may not fully generalize to other universities, subjects, or learning platforms.
2. **Static Predictions**
    - The current approach focuses on end-of-course outcomes. Future work could build time-evolving models that update risk predictions week by week, enabling even earlier interventions.
3. **Feature Scope**
    - While demographics, assessments and clicks are powerful, there may be other useful signals such as forum posts, text sentiment, or peer interactions which are not included.
4. **Handling Class Imbalance**
    - Techniques such as class weighting, SMOTE or customized loss functions could be explored to improve recall on the minority (failing) class.
5. **Interactive Dashboards**
    - The saved models and analysis outputs (CSV files and charts) could be integrated into a dashboard for instructors, where they can monitor risk in real time and drill down into individual students' behavior patterns.

12. Conclusion

This project demonstrates a complete data mining pipeline for student performance prediction and learning behavior analysis using the Open University Learning Analytics Dataset. Starting from raw data dispersed across four tables, the notebook:

- Cleans and merges demographic, registration, assessment and clickstream information into a single student-level dataset.
- Engineers meaningful features that reflect performance, engagement intensity, and temporal behavior.
- Applies association rule mining to discover interpretable patterns that link specific behaviors to outcomes.
- Analyzes binge vs. steady learning and comeback vs. burnout trajectories to highlight how different study strategies relate to success.
- Trains and evaluates multiple machine learning models, selecting and saving the best classifier as a practical tool for early risk prediction.

Overall, the solution is effective not only at predicting which students are likely to pass or fail, but also at explaining why certain students struggle or succeed. This combination of predictive power and interpretability makes the project a strong example of how learning analytics and machine learning can support data-informed decision making in education.