VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

CO3117 - Machine Learning

Assigmment Report

Predict Students' Dropout and Academic Success

Advisor: Msc. Hung Nguyen Thanh

Students: Cuong Doan Phuong Hung ID 2310381
Dang Huynh Tran Hoc ID 2210731
Duc Nguyen Le Anh ID 2210796

HO CHI MINH CITY, 17 OCTOBER 2025

Contents

1 Objectives **1**

2 Dataset Insights **2**
2.1 Data Loading and Initial Inspection 2
2.2 Data Visualization 4
2.2.1 Numerical Feature Distributions 4
2.2.2 Key Categorical Features Distribution 6
2.2.3 Correlation Heatmap 8

3 Preprocessing **9**

4 Training Pipelines **12**
4.1 Model Selection 13
4.2 Training And Evaluating Models 15

5 Enhancing Models **18**
5.1 Feature Engineering 18
5.2 Retrained Models' Evaluation 20
5.3 Hyperparameter Tuning 21
5.3.1 Random Forest 22
5.3.2 Gradient Boosting 23
5.4 Evaluations 24

6 Ensemble Learning **25**

7 Conclusion **26**

List of Figures

2.1 Target Distribution 2
2.2 Numerical Features statistics 3
2.3 Numerical Feature Distributions 4
2.4 Key Categorical Distribution 6
2.5 Correlation Heatmap 8
3.1 Grouped Categorical Features 11
4.1 Pretrain Pipeline 12
4.2 Random Forest Classifier 13
4.3 Gradient Boosting Classifier 14
4.4 Precision - Recall Curve 15
4.5 Some Evaluation Metrics 16
4.6 Top 10 Feature Importances 17
5.1 New Features Distribution 19
5.2 Enhanced Models' Precision - Recall Curve 20
5.3 Enhanced Models' Metrics 21
5.4 Enhanced Models' Top 10 Feature Importances 24
6.1 Voting Pipeline 25

1 Objectives

Aim of the Project

The primary aim of this project is to develop and evaluate a machine learning model for predicting student academic outcomes - specifically, whether a student will graduate, remain enrolled, or drop out - using the *Predict Students' Dropout and Academic Success* dataset from the UCI Machine Learning Repository.

This assignment, part of an introductory Machine Learning course, focuses on applying supervised classification techniques to real-world educational data, emphasizing ethical considerations such as avoiding target leakage and handling class imbalance. By building predictive models, the project seeks to demonstrate how data-driven insights can support early interventions in higher education to improve retention rates and student success.

Specific Objectives

1. **Data Exploration and Preprocessing:** Analyze the dataset's features (demographic, socioeconomic, macroeconomic, academic) to understand distributions, correlations, and potential biases. Perform preprocessing, including feature engineering while ensuring no inclusion of leaky features from post-enrollment periods.

2. **Feature Selection and Leakage Mitigation:** Identify and exclude features that introduce target leakage (e.g., second-semester academic metrics), focusing on enrollment-time and first-semester data for fair, prospective predictions.

3. **Model Development:** Implement baseline and tuned classification models using Random Forest and Gradient Boosting algorithms, incorporating techniques like stratified splitting, robust scaling, one-hot encoding, and class weighting to address multiclass imbalance.

4. **Hyperparameter Tuning and Evaluation:** Use randomized search with cross-validation to optimize model hyperparameters, evaluating performance through metrics such as accuracy, macro F1-score, precision, recall, and ROC-AUC. Assess overfitting and compare results against benchmarks to validate model robustness.

5. **Interpretation and Insights:** Analyze feature importances to uncover key predictors of student outcomes, providing actionable recommendations for educational stakeholders.

Primary Goal

The primary goal is to predict student outcomes as a three-class classification problem: Graduate (completed the degree on time), Enrolled (still ongoing at the end of normal duration), or Dropout (left the institution). This supports early intervention strategies to reduce dropout rates. The dataset has no missing values, and preprocessing was performed to handle anomalies and outliers. Through these objectives, the project not only applies core machine learning concepts such as pipelines, tuning, and evaluation, but also highlights the importance of ethical modeling in sensitive domains like education, where biased or leaky predictions could mislead interventions.

2 Dataset Insights

2.1 Data Loading and Initial Inspection

The dataset was loaded from a CSV file (`data.csv`) into a Pandas DataFrame for analysis. Initial inspection revealed the following key details:

- **Shape:** The dataset consists of **4424** rows (instances) and **37** columns (36 features + 1 target variable). For more details about each features' information please access *UCI Machine Learning Repository*.
- **Target Variable:** `Target` is the multiclass label with three categories:
  - **Graduate**: 2209 instances (∼50%)
  - **Dropout**: 1421 instances (∼32%)
  - **Enrolled**: 794 instances (∼18%)

  This indicates a moderate class imbalance, with Graduates as the majority class.

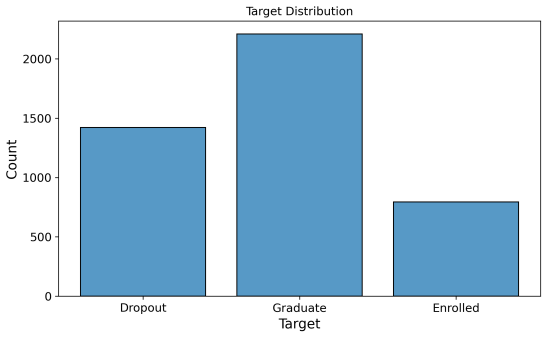

Figure 2.1: Target Distribution

- **Feature Types:**
  - **Numerical (continuous/discrete):** 20 features (e.g., grades, ages, economic indicators)
  - **Categorical (nominal/ordinal/binary):** 16 features (e.g., marital status, course, gender)
- `Missing Values`: *No missing values* were detected across the dataset.

- `Duplicates`: *No duplicate rows* were found.
- `Data Types`: Mostly integers (discrete/binary/ordinal) and floats (grades/rates); the target is categorical (string).

A summary of some basic statistics for numerical features (mean, std, min, max) is provided below (computed via `df.describe()`):

	Pre Qual (grade)	Admission grade	Enroll Age	1st - approved	2nd - approved	2nd - grade	Unemployment rate	Inflation rate	GDP
count	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000	4424.000000
mean	132.613314	126.978119	23.265145	4.706600	4.435805	10.230206	11.566139	1.228029	0.001969
std	13.188332	14.482001	7.587816	3.094238	3.014764	5.210808	2.663850	1.382711	2.269935
min	95.000000	95.000000	17.000000	0.000000	0.000000	0.000000	7.600000	-0.800000	-4.060000
25%	125.000000	117.900000	19.000000	3.000000	2.000000	10.750000	9.400000	0.300000	-1.700000
50%	133.100000	126.100000	20.000000	5.000000	5.000000	12.200000	11.100000	1.400000	0.320000
75%	140.000000	134.800000	25.000000	6.000000	6.000000	13.333333	13.900000	2.600000	1.790000
max	190.000000	190.000000	70.000000	26.000000	20.000000	18.571429	16.200000	3.700000	3.510000

Figure 2.2: Numerical Features statistics

Data Insights and Summary Statistics

- **Age Distribution:** Student ages range from 17 to 70 years, with a mean of approximately 23 years. This reflects a student body that is primarily comprised of traditional undergraduates, but also includes a non-negligible number of older, non-traditional students.
- **Grades:** Key academic grades, including admission and semester averages, fall within a 0-20 scale. The mean values generally range from 10 to 13, indicating that most students perform at or slightly above average, with substantial variability across the cohort.
- **Economic Indicators:** Features such as unemployment rate, inflation, and GDP are included to capture macroeconomic context for each student's enrollment year (2008-2019). Statistical summaries show these indicators vary only moderately across the sample, highlighting changing economic conditions that may indirectly influence student outcomes.
- **Semester Metrics:**
  + *Approvals/Evaluations:* Metrics for the first semester show higher rates of approved units and evaluations when compared to the second semester.
  + *Dropout Effect:* This discrepancy is likely explained by student attrition—some students drop out after the first semester, leading to incomplete or reduced second-semester data.

These insights inform both preprocessing and model development by clarifying typical data ranges, potential sources of variance, and the importance of handling non-standard cases such as older students or those affected by economic conditions.

2.2 Data Visualization

Visualizations were generated using `Matplotlib` and `Seaborn` to explore distributions, relationships, and patterns. All figures were saved to the images directory for inclusion in the report. Key visualizations and insights are described below.

2.2.1 Numerical Feature Distributions

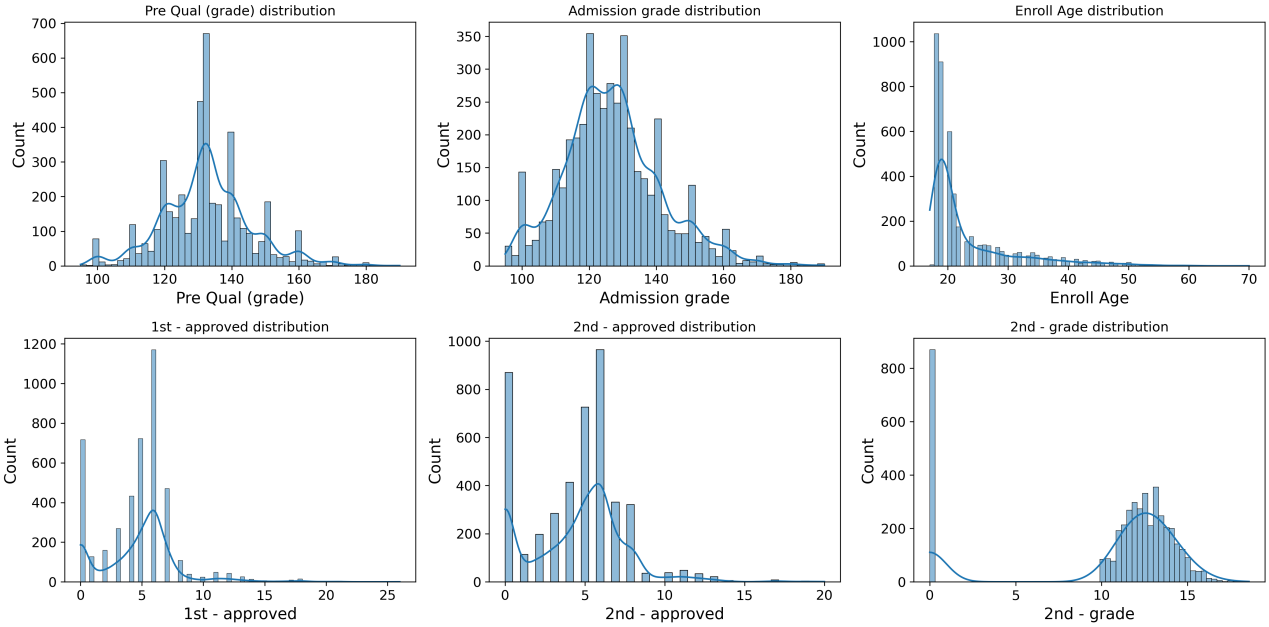

Figure 2.3: Numerical Feature Distributions

Insights The histograms with kernel density estimates (KDE) for selected numerical features highlight distributional characteristics that guide preprocessing steps such as transformation, outlier handling, and scaling to improve model performance and stability:

- **Previous Qualification (grade):** The distribution is roughly symmetric around 130 but shows multimodality with spikes (e.g., at 125, 140), possibly due to grading scales or binning effects. Range 95–190 with std ≈13.2.
- **Admission grade:** Multimodal with peaks around 120–140, similar variance (std ≈14.5) and range (95–190). Indicates clustered admission scores. Robust scaling to mitigate peak influences in distance metrics like KNN.
- **Enrollment Age:** Heavily right-skewed (mean 23.3, median 20) with long tail up to 70, most mass at 18–25. Potential outliers beyond 40. Group ages to reduce parsity

- **1st Semester Approved Units:** Right-skewed with mode at 5–6, many low values (25% at 3, min 0), max 26. Zero-inflated aspect for non-approvals. Treat zeros separately (e.g., binary flag for zero approvals) to handle excess zeros in count-based models.
- **2nd Semester Approved Units:** Similar to 1st semester but slightly shifted (mean 4.4, more spread in low approvals). Right-skewed with potential zeros. Aggregate with 1st semester if correlated; apply square-root transform for variance stabilization in Poisson-like distributions would be a solid option.
- **2nd Semester Grade:** Bimodal with peaks at 0–5 (failures?) and 10–15 (passes), mean 10.2, std 5.2. Heavy mass at low ends, indicating grade inflation or dropout effects. Cap or bin grades into categories if granularity adds noise, especially for tree-based models tolerant to non-normality.

In summary, skewed features like age and approvals benefit from transformations (log/sqrt) to approximate normality for parametric models, while grades may require robust scaling due to multimodality. Outlier clipping and zero-handling are crucial for academic metrics to avoid bias in predictions of student success.

2.2.2 Key Categorical Features Distribution

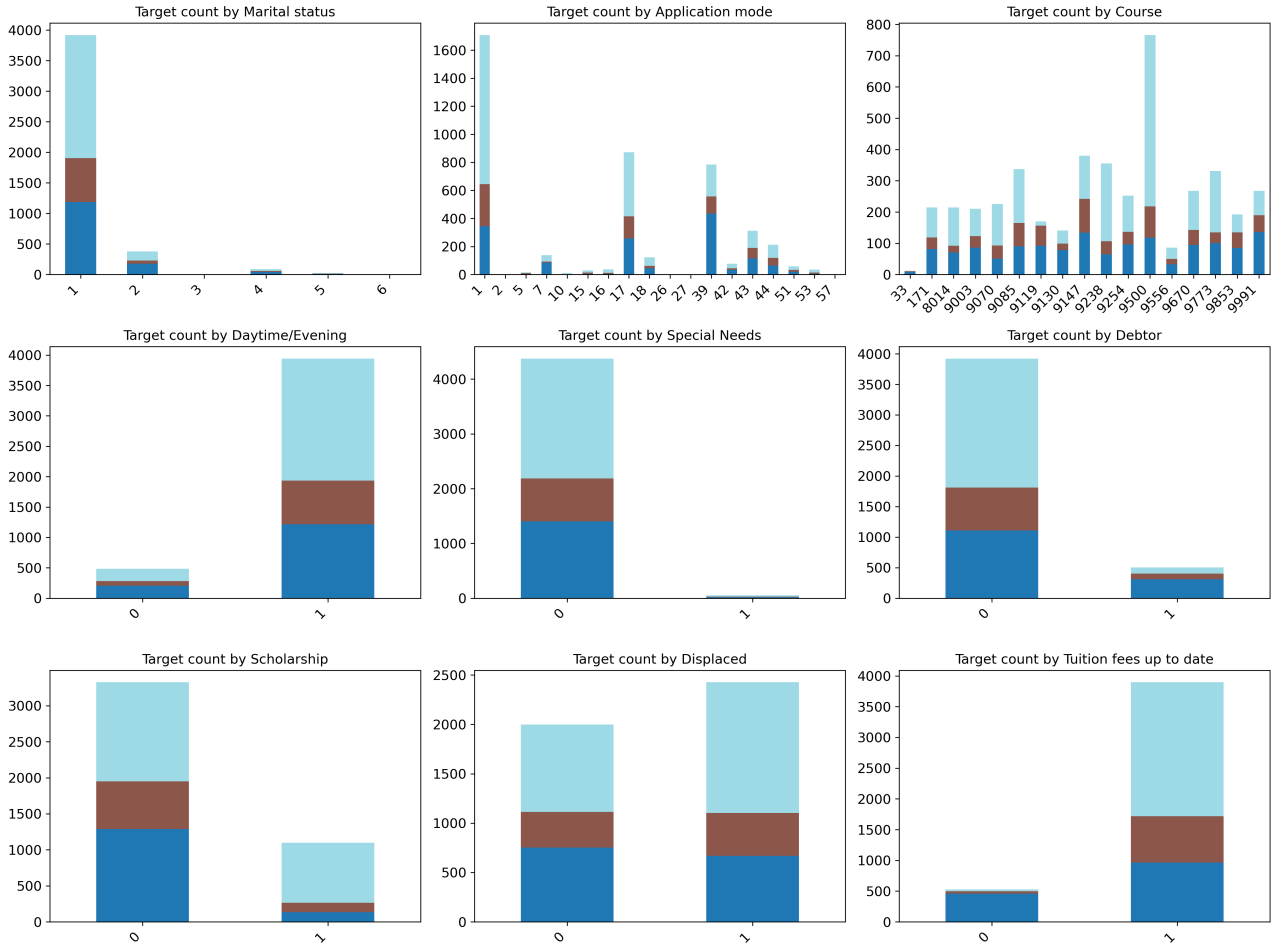

Figure 2.4: Key Categorical Distribution

Insights

- **Marital Status:** Almost all students are single (category 1), with very few in other marital states. This feature exhibits severe imbalance and limited variation, suggesting it may have minimal predictive power or could be grouped into binary categories (single vs. others) to reduce sparsity.
- **Application Mode:** The large concentration in application mode 1 with several smaller categories indicates a high cardinality categorical variable with skewed representation. Rare categories might be grouped or encoded carefully to avoid noise and sparsity in the model.
- **Course:** The counts are spread across many courses, with one course having a notably

high count. This high dimensionality and imbalance may require dimensionality reduction, grouping of less common courses, or using embedding techniques for encoding.

- **Daytime/Evening Attendance:** Shows a clear majority in daytime attendance (category 1), but both categories have sufficient representation. This categorical feature is well balanced for direct encoding.
- **Special Needs:** Overwhelming majority do not have special needs, making this feature highly imbalanced. It may have limited impact unless special needs cases strongly associate with outcomes, warranting inclusion with careful encoding or as a binary flag.
- **Debtor:** Shows class imbalance with most students not being debtors. This binary feature can be used as is, but attention should be paid during model evaluation to account for imbalance effects.
- **Scholarship:** A similar pattern to debtor status, with most students not receiving scholarships. While there is imbalance, the feature may add useful information on socioeconomic status if encoded suitably.
- **Tuition Fees Up to Date:** Nearly all students have tuition fees up to date except for a significant minority. This binary feature appears informative and well suited for direct inclusion.

In summary, categorical features with many rare categories (Application Mode, Course, Marital Status) will benefit from grouping or specialized encoding to reduce sparsity. Features with binary outcomes and moderate imbalance can be used directly with attention to balancing during model training.

2.2.3 Correlation Heatmap

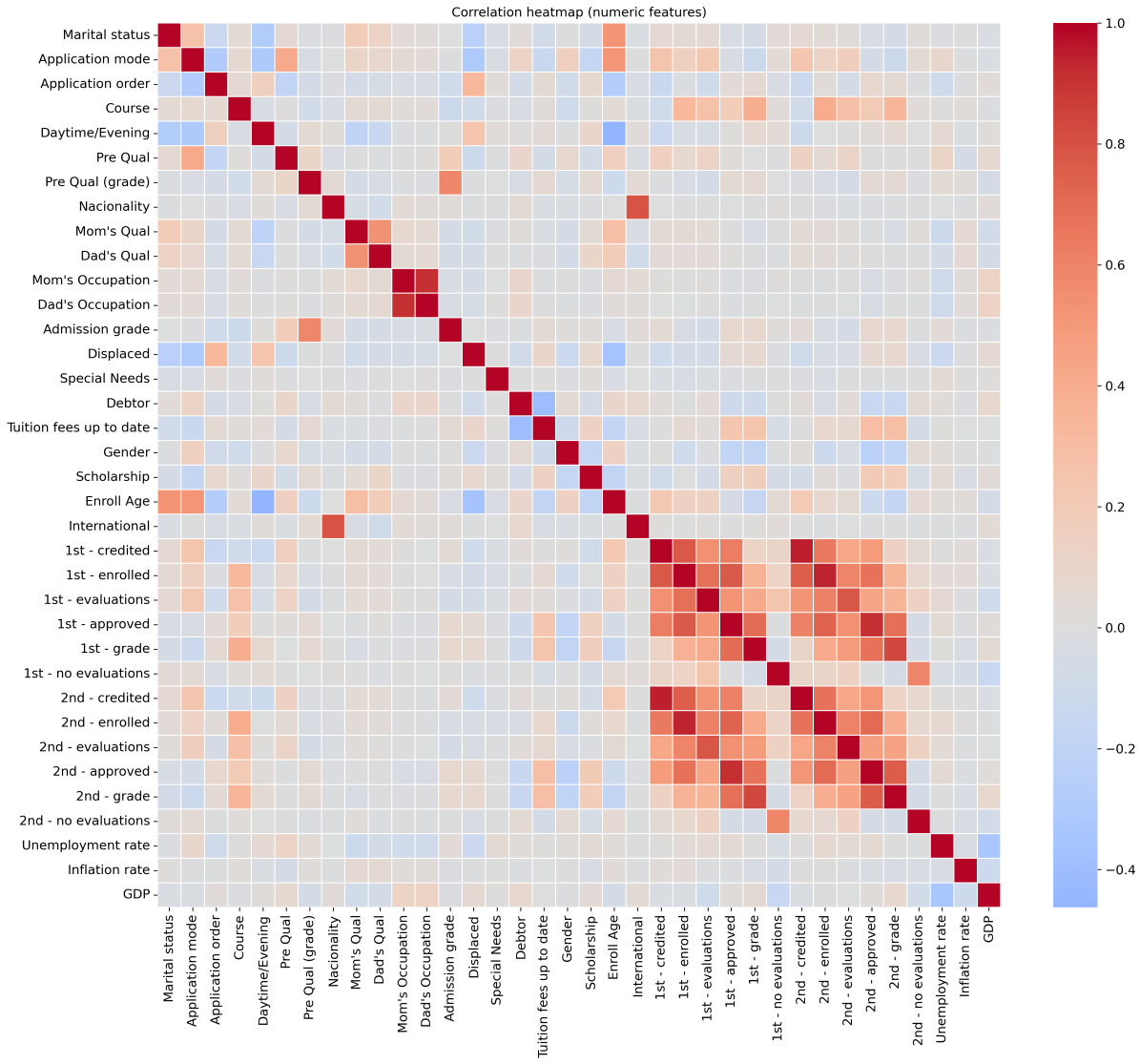

Figure 2.5: Correlation Heatmap

Insights

- **1st and 2nd Semester Academic Metrics:** These show strong positive correlations within and across semesters (often > 0.7, e.g., 1st approved vs. 2nd approved ≈ 0.90). This indicates redundancy; preprocessing should involve aggregating (e.g., total approvals) or applying PCA to reduce dimensions and avoid multicollinearity in linear models.
- **Parental Qualifications and Occupations (Mother's/Father's Qualification, Occupation):** Moderate correlations (0.4–0.7) suggest socioeconomic clustering. Combine

into a parental background score via averaging or factor analysis to simplify without losing information.

- **Macroeconomic Indicators (Unemployment rate, Inflation rate, GDP):** Very weak correlations with all features (< 0.1 absolute), implying low relevance. Drop these during feature selection to minimize noise and model complexity.
- **Enrollment Age:** Low correlations overall (< 0.3) but potential for outliers (range 17–70). Apply winsorization to handle skewness and extreme values that could distort distance-based algorithms.
- **Admission and Previous Grades (Admission grade, Previous qualification grade):** Moderate links to semester grades (0.3–0.5), indicating some predictive value. Standardize scales as they differ from semester grades; retain but monitor for multicollinearity with academic outcomes.
- **International and Nationality:** High correlation (≈ 0.9), as one derives from the other. CLearly we should remove one to eliminate perfect collinearity issues.
- **Gender and Scholarship:** Weak correlations (< 0.2) but binary nature makes them easy to encode. Retain with one-hot or label encoding; address any class imbalance via sampling if linked to target in further analysis.
- **Displaced and Special Needs:** Low inter-correlations (< 0.1); treat as flags. Binary encoding suffices, but check distributions for rarity and potential merging if sparse.

In summary, focus on reducing redundancy in highly correlated academic and family features through aggregation or dimensionality reduction, while dropping weakly correlated macroeconomic variables. Scaling is essential for features with varying ranges (e.g., age, grades) to ensure compatibility in machine learning pipelines.

3 Preprocessing

Anomaly Cleaning

Although the dataset is reported to contain no missing (NaN) values, we nonetheless perform thorough data cleaning as follows to ensure integrity and consistency:

```
# Drop invalid ages (enroll age <17 or >70, or application order <0)
df = df[(df['Enroll Age'] >= 17) & (df['Enroll Age'] <= 70)]
```

```
df = df[df['Application order'] >= 0]
# Drop invalid grades (admission or previous <95 or >190)
df = df[(df['Admission grade'] >= 95) & (df['Admission grade'] <= 190)]
df = df[(df['Pre Qual (grade)'] >= 95) & (df['Pre Qual (grade)'] <= 190)]
# Drop invalid semester metrics (e.g., approved > enrolled, evaluations <0)
for sem in ['1st', '2nd']:
    df = df[df[f'{sem} - approved'] <= df[f'{sem} - enrolled']]
    df = df[df[f'{sem} - evaluations'] >= 0]
    df = df[df[f'{sem} - grade'] >= 0]
# Drop rows with invalid binary/ordinal values (e.g., gender not 0/1)
df = df[df['Gender'].isin([0, 1])]
df = df[df['Scholarship'].isin([0, 1])]
df = df[df['Tuition fees up to date'].isin([0, 1])]
```

Grouping Categorical Features

In the preprocessing pipeline, several categorical features with high cardinality or sparse codes were grouped into broader, more interpretable categories to reduce dimensionality, mitigate overfitting, and enhance model performance.

```
df_prep = df_prep.drop(columns=['Debtor', 'Special Needs', 'Unemployment rate',
 'Inflation rate', 'GDP'])
col_func_map = {
    'Marial Status': lambda x: marial(x),
    'Application mode': lambda x: app_mode(x),
    'Course': lambda x: course(x),
    'Pre Qual': lambda x: pre_qual(x),
    'Nationality': lambda x: nationality(x),
    "Mom's Qual": lambda x: qual(x),
    "Dad's Qual": lambda x: qual(x),
    "Mom's Occupation": lambda x: moms_job(x),
    "Dad's Occupation": lambda x: dads_job(x)
}
for col, func in col_func_map.items():
    df_prep[col] = df_prep[col].apply(func)
```

Grouping decisions were informed by domain knowledge and the dataset documentation, where feature codes correspond to specific educational, occupational, or administrative groups. The motivations for this step include:

- **Dimensionality Reduction:** Categorical features like Application mode or Course would produce sparse and high-dimensional encodings if left ungrouped. Grouping codes with similar meaning reduces this sparsity and enhances the stability of subsequent models.
- **Interpretability:** Aggregating codes along logical groupings (e.g., collapsing educational qualifications into "basic", "secondary", and "higher" categories) results in features that are easier to interpret and reason with, both in exploratory analysis and model output.
- **Handling Imbalance:** Many categorical codes have very few samples merging these rare categories prevents poorly-represented classes from introducing noise or overfitting.
- **Leakage Avoidance:** All groupings were defined based only on code descriptions and situational knowledge, and were applied prior to any exposure to the outcome labels. Thus, target leakage is avoided.

This process ensures the resulting categorical variables are meaningful, compact, and suitable for downstream encoding and analysis (as shown in preprocess.ipynb):

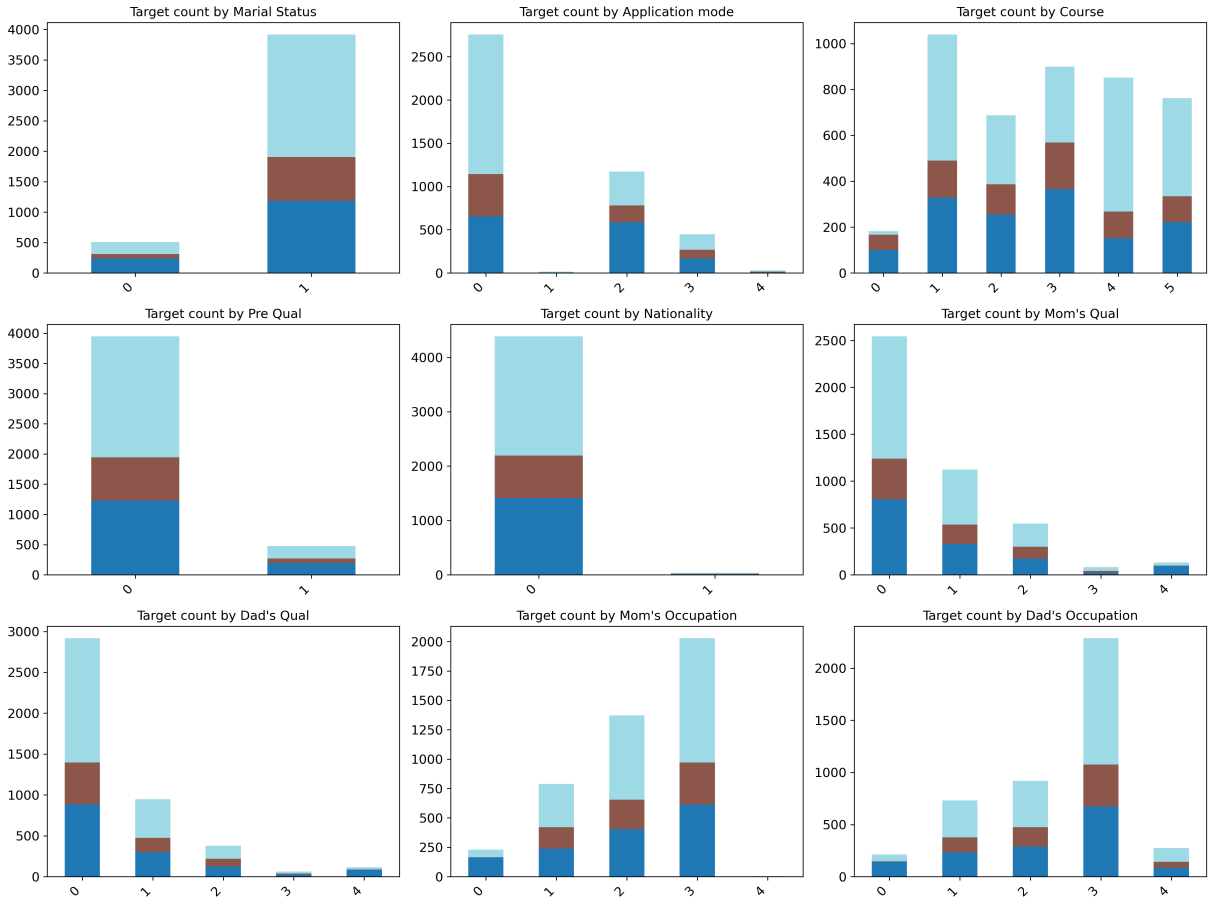

Figure 3.1: Grouped Categorical Features

Post-grouping plots reveals consistent patterns: Socioeconomic advantages (higher parental education/occupation, scholarships) boost graduation, while non-standard admissions or manual backgrounds increase dropout risk. Imbalances persist (e.g., dominant groups skew distributions), necessitating stratified sampling and weighted metrics. Grouping reduced features from high-cardinality to 2-7 levels, improving efficiency (e.g., OHE dimensions from around 100 to around 30). In modeling, these features ranked moderately in importances, interacting with academics. Overall, this step enhanced predictive power and interpretability, enabling targeted educational strategies.

4 Training Pipelines

The training process utilized the preprocessed and engineered dataset to build and evaluate multiclass classification models for predicting student outcomes. The workflow included data splitting, pipeline construction, baseline training, hyperparameter tuning via randomized search, comprehensive evaluation, overfitting checks, feature importance analysis, and visualization of confusion matrices. All steps emphasized imbalance mitigation (stratified splits, class weights) and reproducibility (`random_state` = 42).

```
preprocessor = ColumnTransformer(
  transformers=[
      ('num', RobustScaler(), numeric_cols),
      ('cat', OneHotEncoder(handle_unknown='ignore', sparse_output=False),
   categorical_cols)],
  remainder='drop'
)
```

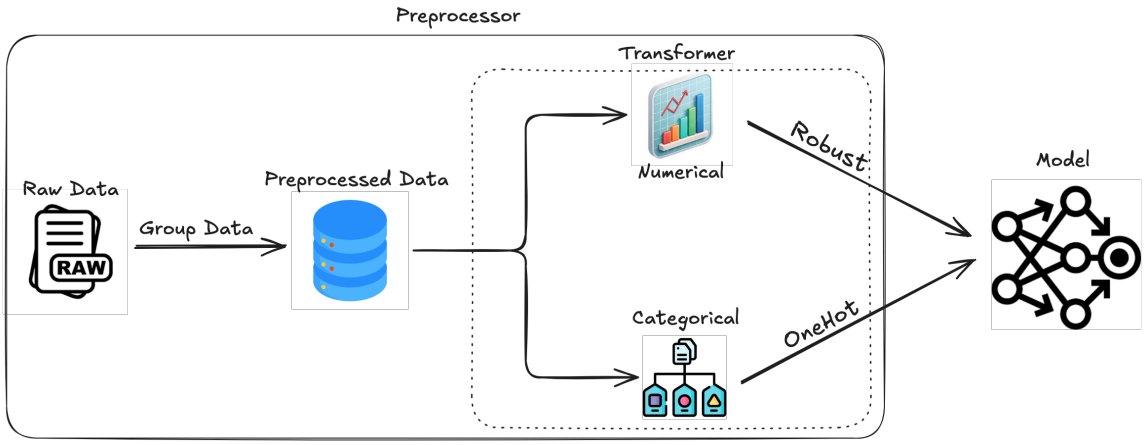

Figure 4.1: Pretrain Pipeline

4.1 Model Selection

The dataset contains 4424 instances with 36 features, including numerical, categorical, and discrete integers, aimed at classifying students into dropout, enrolled, or graduate categories. It has no missing values but features strong class imbalance, making it ideal for educational predictive modeling to identify at-risk students early through machine learning classifiers. Focus was on tree-based ensembles — Random Forest and Gradient Boosting - due to their handling of mixed features and interpretability.

```
models = {
    'RandomForest': RandomForestClassifier(n_estimators=400, max_depth=None,
    random_state=42, class_weight='balanced_subsample', n_jobs=-1),
    'GradientBoosting': GradientBoostingClassifier(random_state=42)
}
```

Random Forest Classifier

Random Forest works by creating an ensemble of multiple decision trees, each trained on a random subset of the data (bootstrapping) and features, then aggregating their predictions through majority voting for classification or averaging for regression. This reduces overfitting, handles variance well, and improves accuracy by leveraging diversity among the trees.

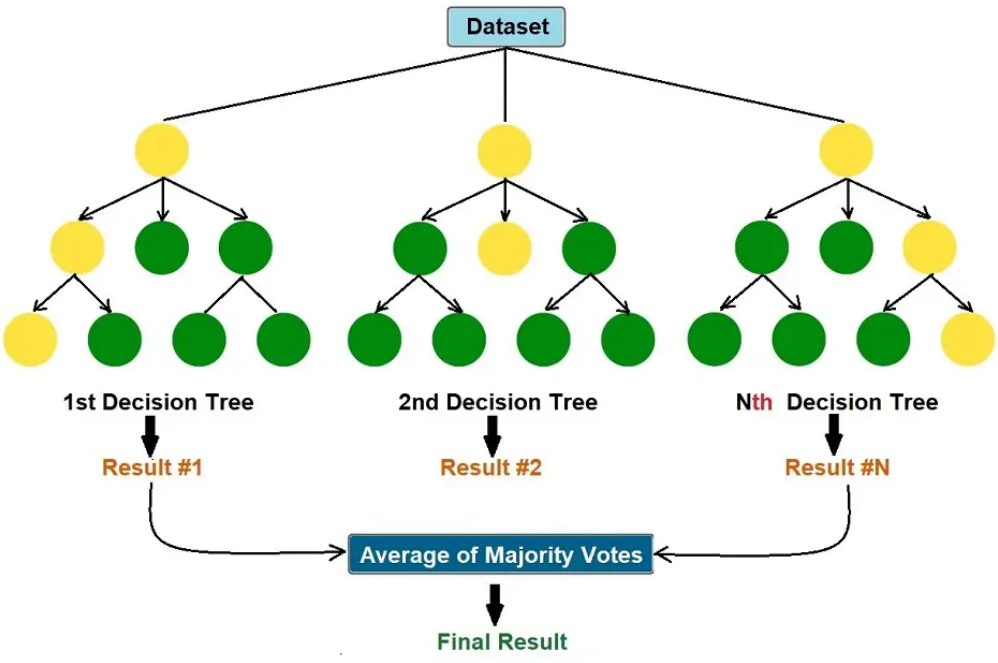

Figure 4.2: Random Forest Classifier

Random Forest (RF) is suitable for this dataset as an ensemble method that builds multiple decision trees via bagging, effectively handling mixed feature types, non-linear relationships, and class imbalance with techniques like class weighting. It reduces overfitting through randomness and provides feature importance scores, offering interpretable insights into factors like socio-economic status or grades that influence dropout, while achieving robust performance on tabular data without extensive preprocessing.

Gradient Boosting Classifier

Gradient Boosting builds trees sequentially, where each new tree corrects the errors of the previous ones by minimizing a loss function using gradient descent. It focuses on hard-to-predict instances, often achieving high performance through additive modeling, with regularization to prevent overfitting.

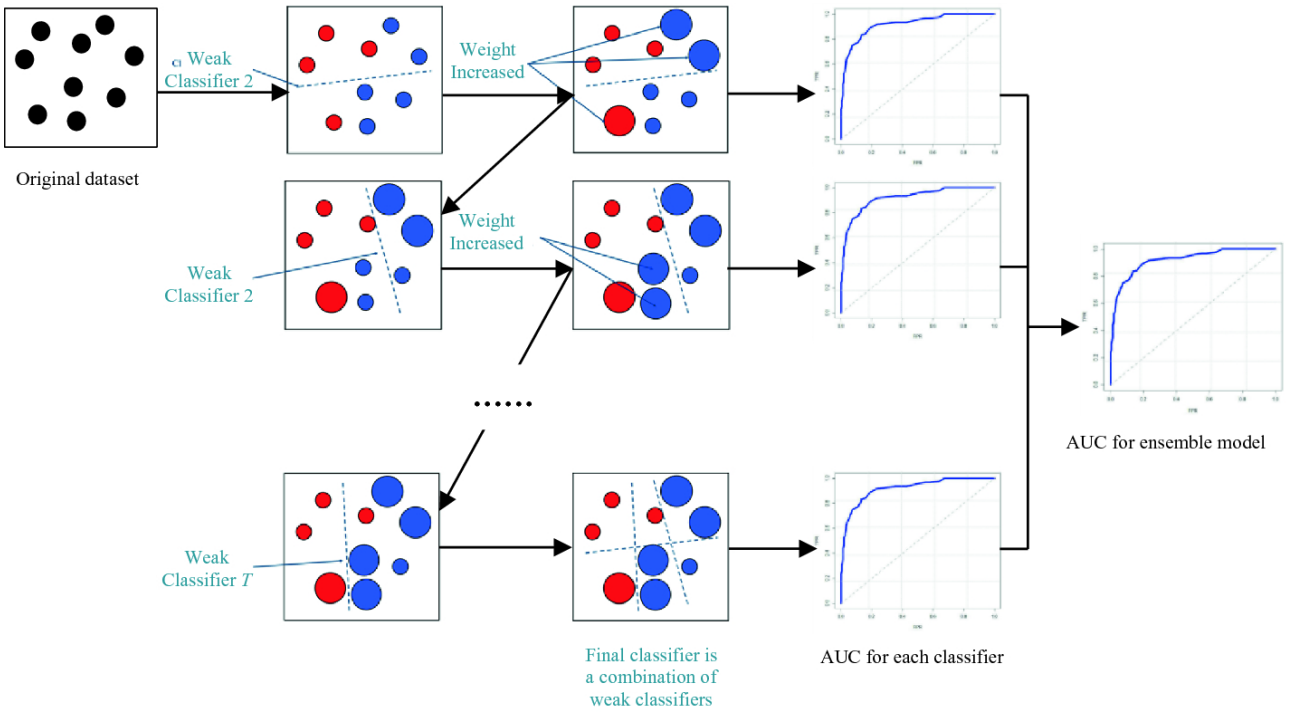

Figure 4.3: Gradient Boosting Classifier

Gradient Boosting excels by sequentially building trees to correct errors, capturing complex interactions and subtle patterns in the data better than single models. It addresses imbalance with built-in weighting and regularization to prevent overfitting, often yielding higher accuracy on imbalanced educational datasets, and includes feature importance for explaining predictions, making it a strong choice for model selection to optimize metrics like F1-score in student success forecasting.

4.2 Training And Evaluating Models

Traning Process

```
for name, model in models.items():
  pipe = Pipeline([
      ('prep', preprocessor),
      ('model', model)
  ])
  pipe.fit(X_train, y_train)
  y_pred = pipe.predict(X_test)
```

Evaluating

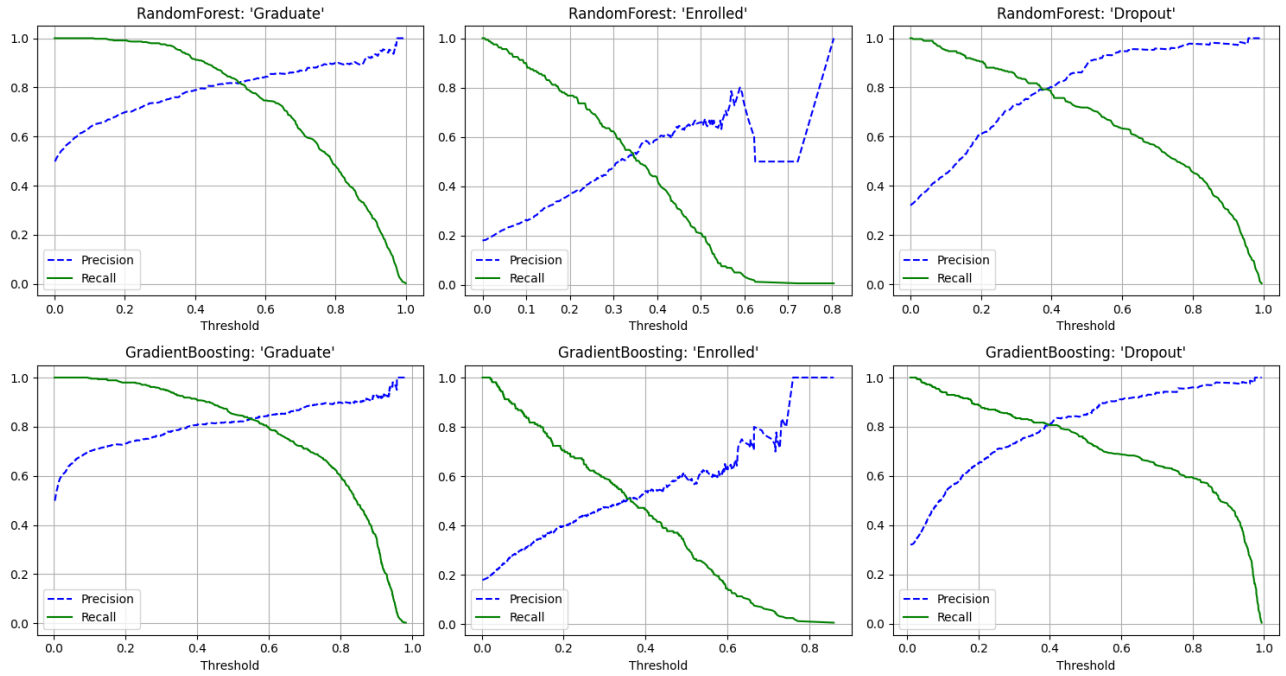

Figure 4.4: Precision - Recall Curve

Overall, both models demonstrate the typical precision-recall tradeoff, with precision increasing and recall decreasing as the decision threshold rises. Gradient Boosting curves are smoother and exhibit steeper transitions, especially for the Enrolled class, indicating better probability calibration and sharper discrimination across thresholds. The Random Forest curves, while broadly following the same trend, show more variability—especially minor fluctuations in recall for the Enrolled group—likely a reflection of the model's inherent randomness due to bagging.

For the majority class (Graduate), both precision and recall remain consistently high and stable, as expected due to class imbalance favoring this group. In contrast, the minority class (Enrolled) presents erratic, lower precision and recall curves, highlighting difficulty in separating this group from others - a common challenge in imbalanced datasets, even when using class weights. Notably, GB attains slightly higher precision at mid-range thresholds, particularly for Enrolled, and this aligns with its marginally superior F1_macro score. This advantage suggests that *GB more effectively balances* precision and recall for underrepresented classes, reaffirming its selection for optimizing imbalanced multi-class problems.

	accuracy	f1_macro	f1_weighted	precision_macro	precision_weighted	recall_macro	recall_weighted	roc_auc_ovr
RandomForest	0.769492	0.696647	0.757046	0.725448	0.757467	0.685157	0.769492	0.878347
GradientBoosting	0.783051	0.718451	0.774273	0.737902	0.772495	0.708721	0.783051	0.882816

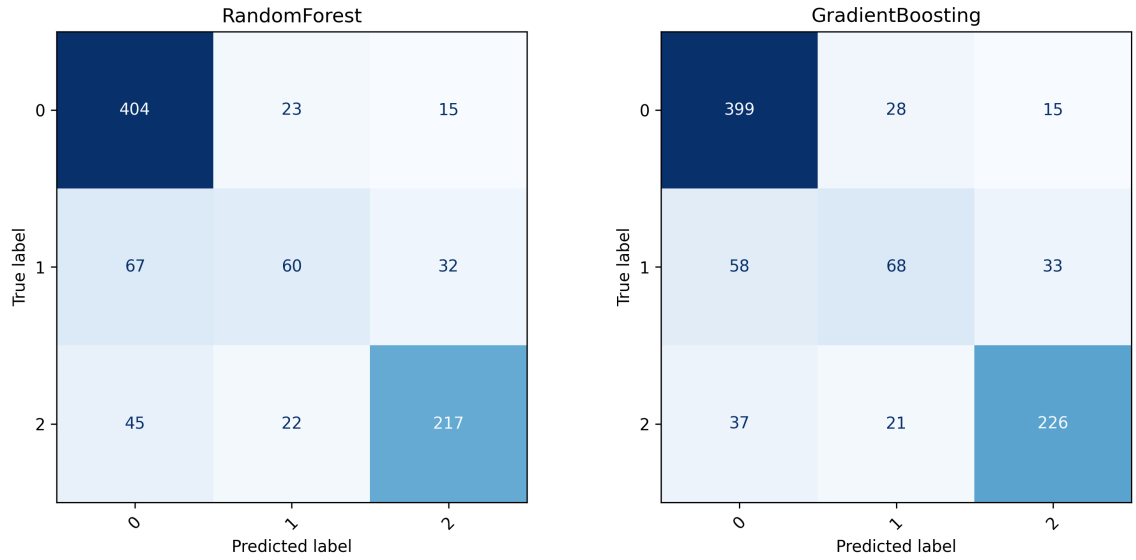

Figure 4.5: Some Evaluation Metrics

Key Insights:

- **Gradient Boosting** excels in probability calibration and precision/recall control, making it especially suitable for imbalanced multiclass problems where identifying and supporting minority classes is vital. Its consistently strong macro-averaged F1-score provides additional support for its selection in such contexts.
- **Random Forest** - while somewhat less precise with calibrated probabilities - provides robustness to noise and outliers, making it valuable when input features are heterogeneous or noisy.

- Both models, particularly when complemented by customized threshold selection and interpretability tools, enable actionable insights for institutions: facilitating risk stratification, optimally directing resources, and improving support for students most at risk of adverse outcomes.

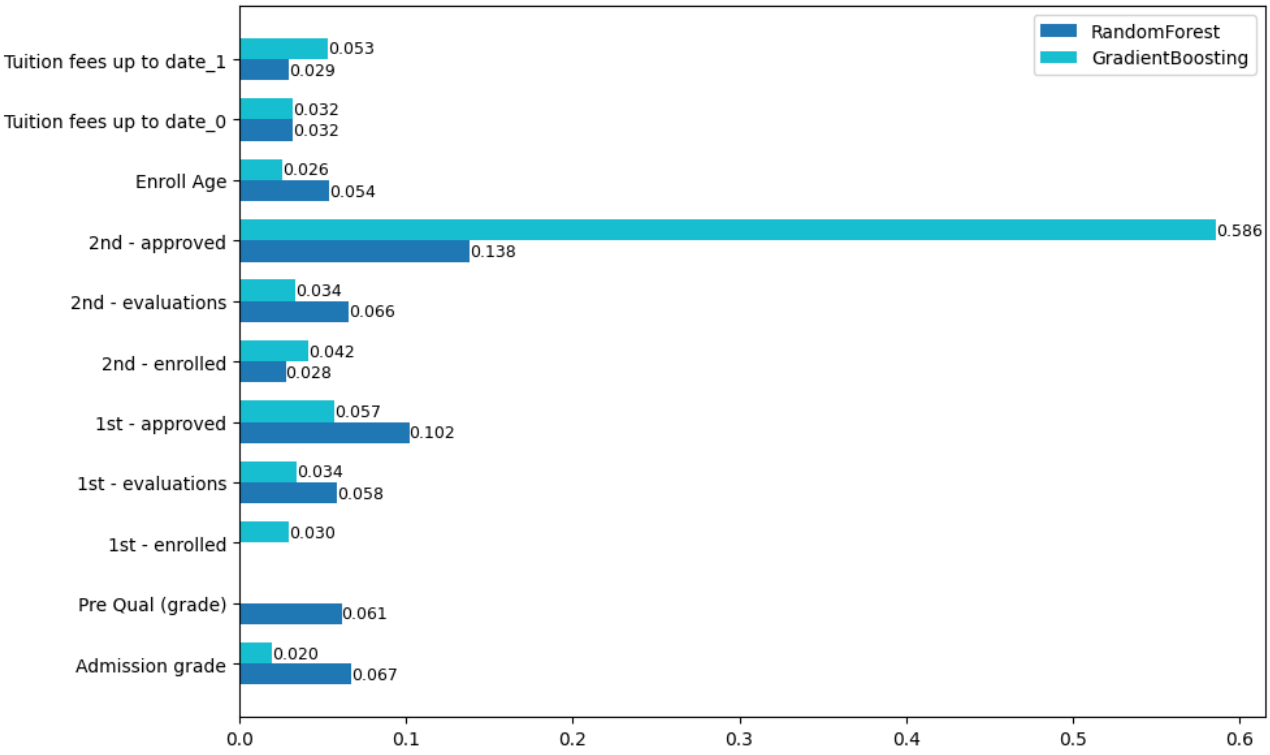

Figure 4.6: Top 10 Feature Importances

Both models highlight academic performance in the early semesters as key predictors of student outcomes, aligning with intuitive expectations: early success (or failure) often signals overall trajectory in higher education. However, the models differ significantly in how they distribute importance, reflecting their algorithmic differences - Random Forest (an ensemble of independent trees) tends to spread importance more evenly, while Gradient Boosting (sequential error-correcting trees) can concentrate on a few dominant features.

- **Gradient Boosting:** High concentration on one feature indicates potential efficiency but risks over-reliance if this feature is noisy or correlated. This makes Gradient Boosting sensitive to second-semester data.
- **Random Forest:** The even distribution reflects it's bagging approach, averaging across diverse trees to reduce variance. This makes it less prone to overemphasizing one feature, potentially leading to more generalizable insights.

5 Enhancing Models

5.1 Feature Engineering

To enhance the predictive power of the dataset, a custom function was applied to derive new variables from raw academic and demographic data. This step focused on creating relative metrics that normalize absolute values, making them more comparable across students and capturing trends in performance. These engineered features proved highly informative for modeling, as indicated by their consistently high rankings in feature importance analyses, outperforming raw counts by emphasizing success efficiency and academic momentum - critical signals for identifying dropout risk.

```
def create_engineered_features(df):
    # Semester-level rates and averages
    for sem in ['1st', '2nd']:
        enrolled = df[f'{sem} - enrolled']
        approved = df[f'{sem} - approved']
        evaluations = df[f'{sem} - evaluations']
        grade = df[f'{sem} - grade']
        # Approval rate: approved / enrolled (zero-denominator safe)
        df[f'{sem}_approval_rate'] = approved.divide(enrolled.replace(0,
    np.nan)).fillna(0)
        # Evaluation rate: evaluations / enrolled
        df[f'{sem}_evaluation_rate'] = evaluations.divide(enrolled.replace(0,
    np.nan)).fillna(0)
        # Average grade: grade / evaluations
        df[f'{sem}_avg_grade'] = grade.divide(evaluations.replace(0,
    np.nan)).fillna(0)
    # Performance improvement deltas (2nd - 1st; for analysis)
    df['delta_approval_rate'] = df['2nd_approval_rate'] - df['1st_approval_rate']
    df['delta_avg_grade'] = df['2nd_avg_grade'] - df['1st_avg_grade']
    # Age binning: four categories (0: <=20, 1: 21-24, 2: 25-30, 3: >30)
    df['AgeGroup'] = pd.cut(
        df['Enroll Age'],
        bins=[-1, 20, 24, 30, np.inf],
        labels=[0, 1, 2, 3]
    ).astype(int)
    return df
```

The main derived features are:

- **Semester-Level Rates:** For each semester, `approval_rate`, `evaluation_rate`, and `avg_grade` were computed with safeguards for division by zero. These normalized ratios deliver robust insights into academic engagement and success, better reflecting student trajectories than absolute counts.
- **Improvement Deltas:** Differences between second and first semester rates were calculated to quantify progress or decline. While these helped analyze performance changes, they were omitted from final models to prevent introduction of future-data leakage.
- **Age Grouping:** The continuous `Enroll Age` variable was binned into four interpretable categories (0: ≤ 20, 1: 21–24, 2: 25–30, 3: > 30) to reduce noise and help models detect age-related risk factors.

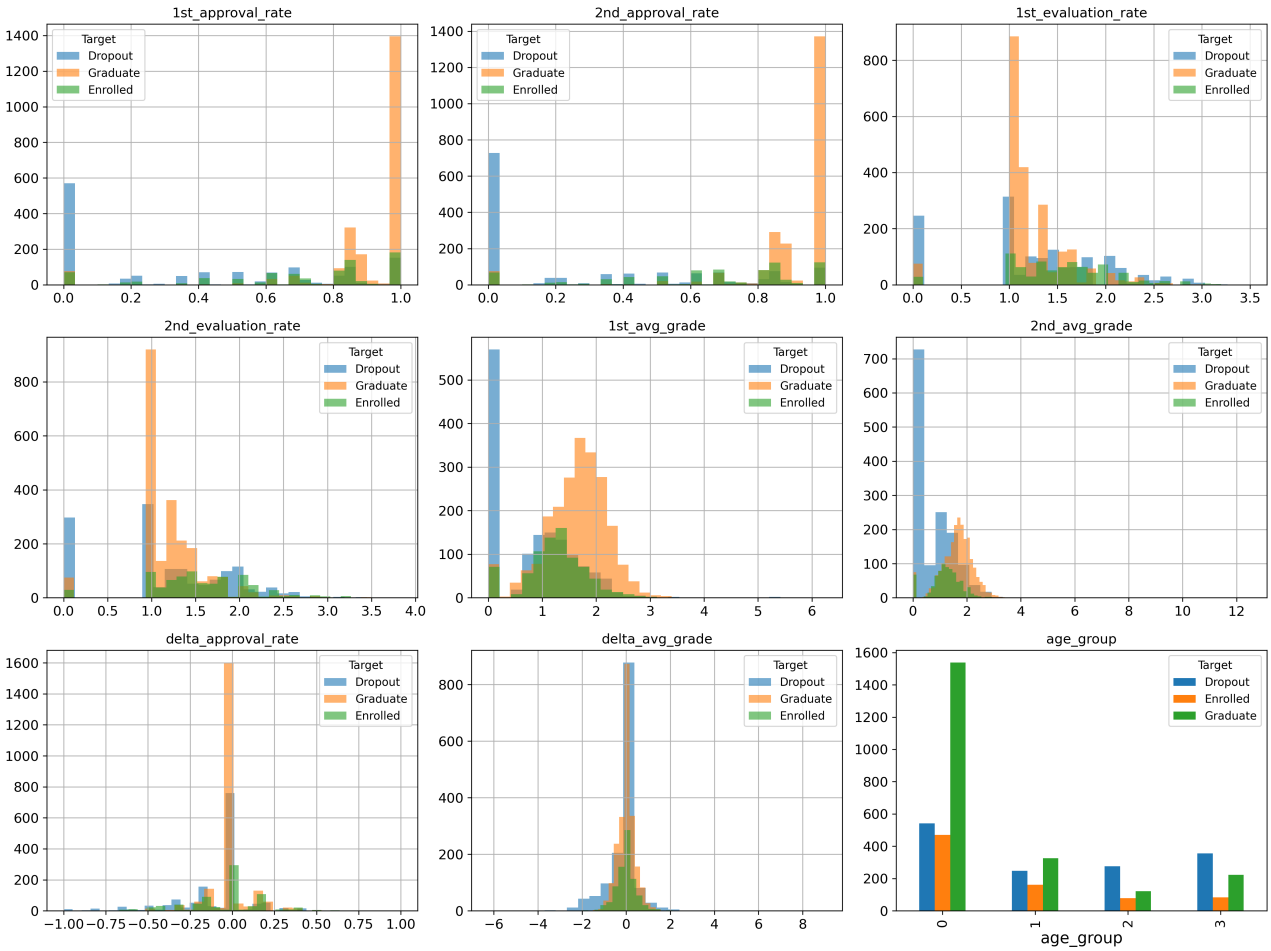

Figure 5.1: New Features Distribution

These visualizations reveal distinct distributional patterns that underscore the features' value in distinguishing outcomes. High approval/grade rates strongly favor Graduates, while negative deltas and older age groups signal dropout risk. These insights validate the engineering choices, as the new features capture normalized performance and trends more effectively than raw metrics, ranking highly in model importances and improving predictive separation without introducing bias. These enhancements improved both model interpretability and predictive accuracy by emphasizing relative academic performance, validated by the dominance of rate-based features in the model's importance rankings.

5.2 Retrained Models' Evaluation

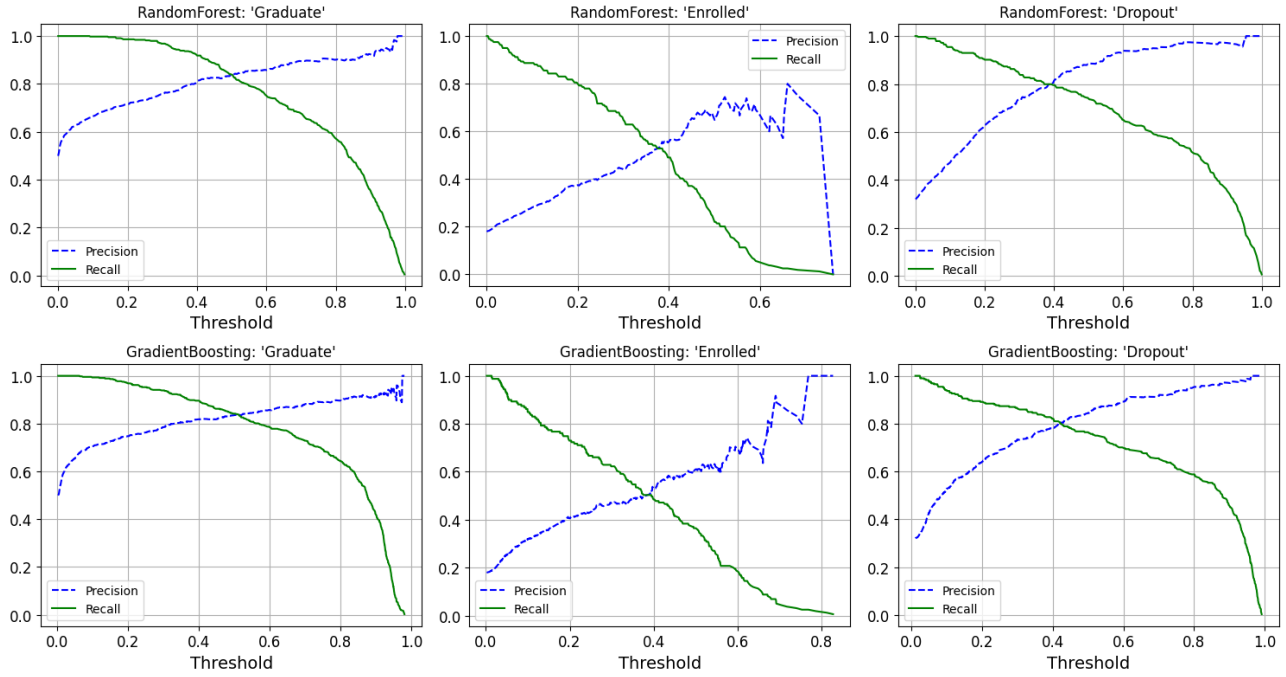

Figure 5.2: Enhanced Models' Precision - Recall Curve

Feature engineering and hyperparameter tuning have clearly enhanced model performance. The post-tuning curves show improved precision, more stable recall, and better overall trade-offs – especially for the previously unstable Enrolled class. These improvements indicate more confident and reliable model predictions across all classes. Comparing models, Gradient Boosting generally shows smoother precision-recall trends than Random Forest, especially for the Graduate and Dropout classes. It also tends to achieve slightly higher precision at higher thresholds. However, both models struggle similarly with the Enrolled class, suggesting the issue lies more with the data than the model itself.

After retraining with the engineered features and tuned hyperparameters, both models showed improved performance, especially in handling the class imbalance:

	accuracy	f1_macro	f1_weighted	precision_macro	precision_weighted	recall_macro	recall_weighted
RandomForest	0.780791	0.715736	0.771916	0.736498	0.770893	0.705115	0.780791
GradientBoosting	0.771751	0.707667	0.764722	0.719111	0.761292	0.701599	0.771751

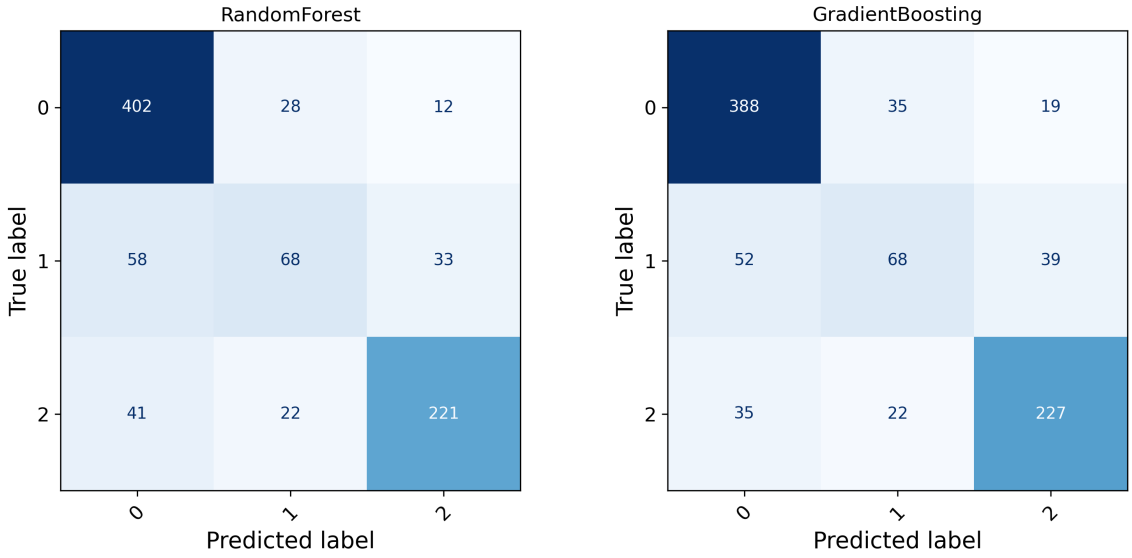

Figure 5.3: Enhanced Models' Metrics

While the accuracy lift was modest, the macro F1-score and ROC-AUC scores improved, indicating a better balance in predicting the minority classes. The new feature importances confirmed the value of feature engineering, with `2nd_approval_rate` becoming the top predictor for the tuned Random Forest model. An overfitting analysis showed the difference between cross-validation and test scores was minimal and acceptable for both models.

5.3 Hyperparameter Tuning

Set up

— A custom scoring function using macro-averaged F1 (`f1_scorer`) is created. This metric gives equal weight to each class, making it well-suited for imbalanced problems where accuracy alone could be misleading.

— Cross-validation is configured using `StratifiedKFold` (`cv`) with 5 splits and shuffling. This ensures the proportion of each class is maintained in all folds, preventing bias during training and model evaluation.

— Class-balanced sample weights (`sample_weights`) are computed for `y_train` using sklearn's 'balanced' strategy. These weights can be supplied to models so that each class has an appropriate influence in the loss, offsetting the effects of class imbalance and helping avoid bias toward the majority class.

```
# Custom scorer for imbalanced data
f1_scorer = make_scorer(f1_score, average='macro')
# Setup cross-validation with proper stratification
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
sample_weights = compute_sample_weight('balanced', y_train);
```

5.3.1 Random Forest

To optimize the Random Forest model, a hyperparameter tuning pipeline was constructed using Scikit-learn's *Pipeline* and *RandomizedSearchCV*. This approach ensures that preprocessing steps are properly applied within each cross-validation fold, preventing data leakage and leading to a more reliable evaluation of the model's performance.

Tuning Pipeline

```
# Parameter grids
rf_param_grid = {
  'n_estimators': [300, 400, 500],
  'max_depth': [20, 25, 30],
  'min_samples_split': [5, 10, 15],
  'min_samples_leaf': [2, 4, 6],
  'max_features': ['sqrt', 'log2'],
  'class_weight': ['balanced_subsample'],
  'criterion': ['gini', 'entropy']
}
# Create pipelines outside of search
def create_rf_pipeline():
    return Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', RandomForestClassifier(random_state=42, n_jobs=-1))
    ])
```

Hyperparameter Tuning Process

```
rf_search = RandomizedSearchCV(
  create_rf_pipeline(),
```

```
  param_distributions={'classifier__' + k: v for k, v in rf_param_grid.items()},
  n_iter=50,
  scoring={'f1_macro': f1_scorer, 'accuracy': 'accuracy'}, refit='accuracy',
  cv=cv,
  random_state=42,
  n_jobs=-1,
  verbose=0,
  return_train_score=True
)

rf_search.fit(X_train, y_train)
```

5.3.2 Gradient Boosting

Tuning Pipeline

```
# Parameter grids
gb_param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.05, 0.1, 0.15],
    'max_depth': [3, 5, 7, 9],
    'min_samples_split': [10, 20, 30],
    'min_samples_leaf': [5, 10, 15],
    'subsample': [0.8, 0.9, 1.0],
}
# Create pipelines outside of search
def create_gb_pipeline():
    return Pipeline([
        ('preprocessor', preprocessor),
        ('classifier', GradientBoostingClassifier(random_state=42))
    ])
```

Hyperparameter Tuning Process

```
gb_search = RandomizedSearchCV(
  create_gb_pipeline(),
  param_distributions={'classifier__' + k: v for k, v in gb_param_grid.items()},
  n_iter=50,
  scoring={'f1_macro': f1_scorer, 'accuracy': 'accuracy'}, refit='accuracy',
  cv=cv,
```

```
  random_state=42,
  n_jobs=-1,
  verbose=0,
  return_train_score=True
)

gb_search.fit(X_train, y_train, classifier__sample_weight=sample_weights)
```

5.4 Evaluations

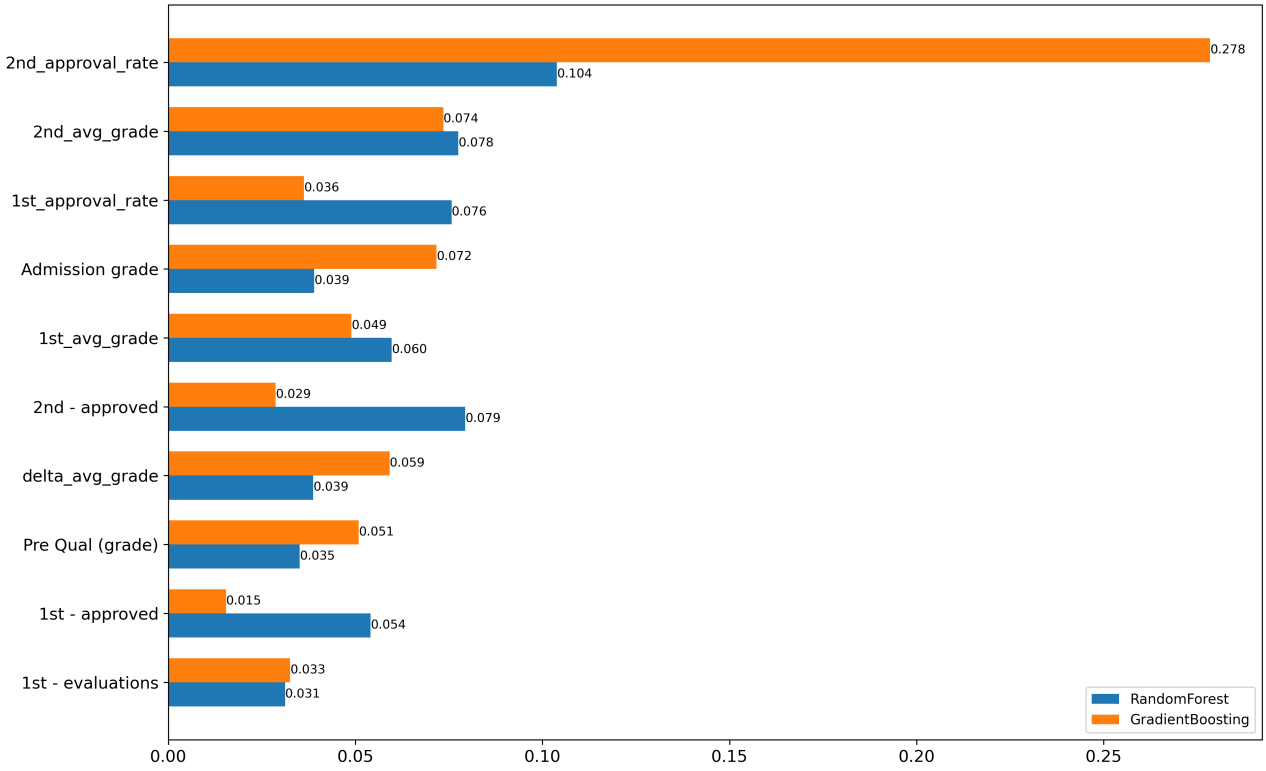

Figure 5.4: Enhanced Models' Top 10 Feature Importances

6 Ensemble Learning

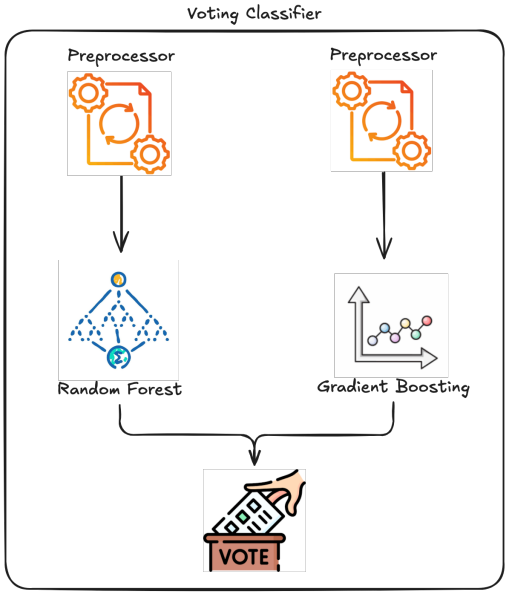

Figure 6.1: Voting Pipeline

7 Conclusion

This machine learning project successfully developed predictive models for student outcome classification, achieving over 77% accuracy in identifying students who will graduate, remain enrolled, or dropout. The comprehensive feature engineering approach, particularly the creation of approval rates and performance delta metrics, significantly contributed to model performance.

The systematic workflow from data exploration through hyperparameter optimization demonstrates best practices in machine learning project development. The results provide a foundation for implementing early warning systems in educational institutions to identify at-risk students and enable timely interventions.