Student Dropout Prediction Using Machine Learning

Contents

1. Introduction and Problem Statement

In the landscape of higher education, student dropout rates present a significant challenge for academic institutions globally. The ability to proactively identify students at risk of attrition is crucial for providing timely interventions and fostering a supportive learning environment that promotes academic success. This project addresses this challenge by developing a predictive model to distinguish between students who are likely to drop out and those who will successfully graduate.

The primary objective of this study is to leverage machine learning techniques to build a robust classification model. By analyzing a comprehensive dataset of student information encompassing demographic data, socio-economic background, and academic performance. We aim to uncover the key factors that differentiate these two critical outcomes. The successful implementation of this model can provide immense value, enabling educational institutions to allocate resources more effectively and design targeted support programs for at-risk students. The problem is framed as a binary classification task, where the model will predict one of two outcomes for each student: "Dropout" or "Graduate."

2. Dataset Description

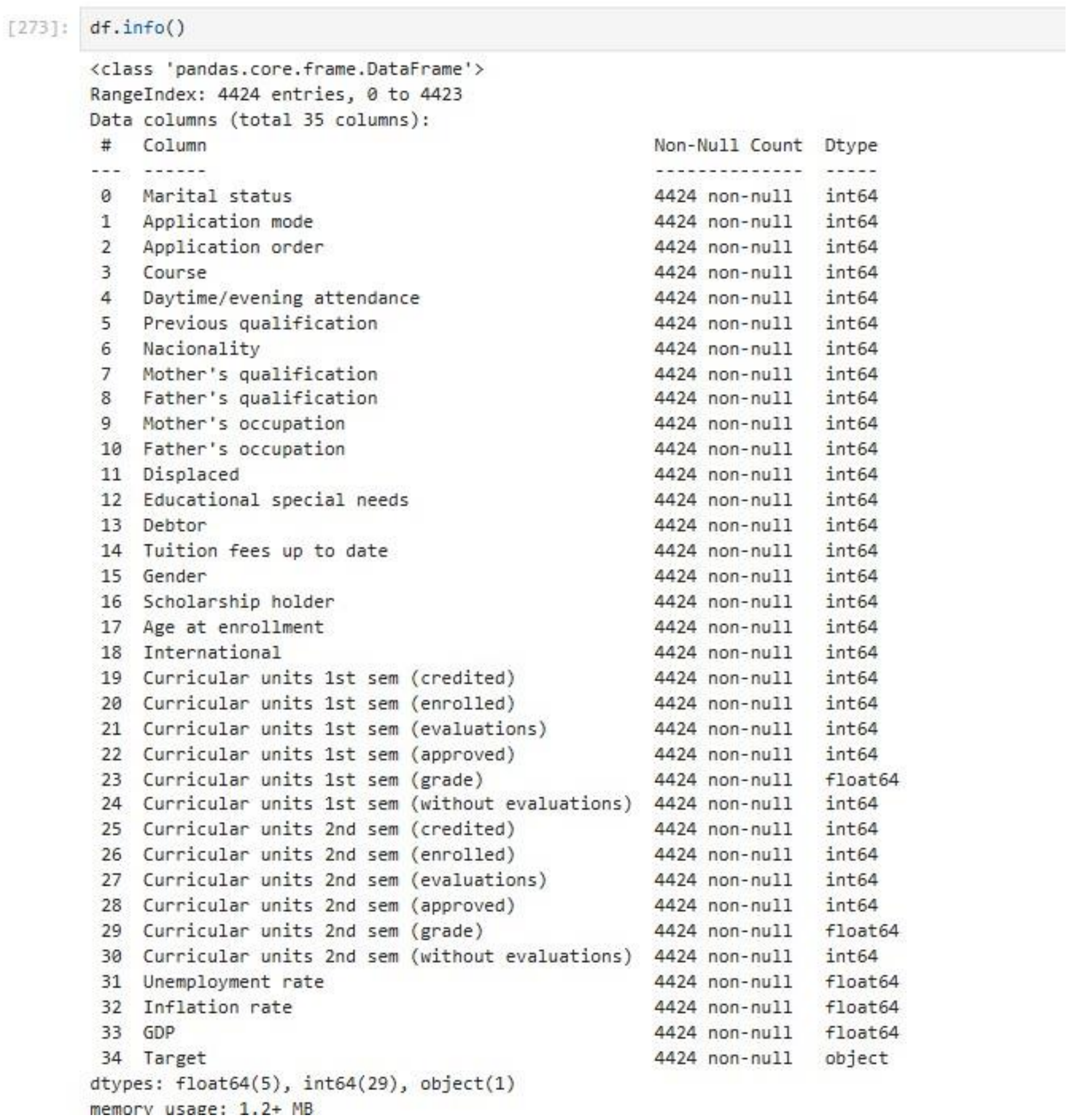
This project utilizes the "Predict students' dropout and academic success (graduate)" dataset, which is publicly available from the UCI Machine Learning Repository. The dataset initially contains 4,424 instances and 37 attributes. The data was collected from a higher education institution in Portugal and includes information about students enrolled in various undergraduate degree programs.

Link : https://www.kaggle.com/api/v1/datasets/download/thedevastator/higher-education-predictors-ofstudent-retention

The features in the dataset can be broadly categorized as follows:

- **Demographic Data:** Includes attributes such as marital status, nationality, gender, and age at enrollment.
- **Socio-Economic Factors:** Captures information like mother's and father's qualifications and occupations, providing insight into the student's background.
- **Academic Information (Pre-Higher Education):** Contains details about previous educational qualifications, admission grades, and performance in secondary school.
- **Academic Information (Higher Education):** Tracks the student's progress within the institution, including credited, enrolled, and approved units for both the first and second semesters, as well as tuition fee status and scholarship details.
- **Target Variable:** For this project, the "Target" attribute was refined to focus on the two definitive outcomes: "Dropout" and "Graduate." Students with an "Enrolled" status were excluded to create a clear binary classification problem.

A potential limitation is that the data is from a single institution in Portugal, which might affect the generalizability of the model. Furthermore, inherent biases could exist in the data, reflecting historical trends in academic success.



```
[273]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4424 entries, 0 to 4423
Data columns (total 35 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  4424 non-null   int64
 1   Application mode                                4424 non-null   int64
 2   Application order                               4424 non-null   int64
 3   Course                                          4424 non-null   int64
 4   Daytime/evening attendance                      4424 non-null   int64
 5   Previous qualification                          4424 non-null   int64
 6   Nacionality                                     4424 non-null   int64
 7   Mother's qualification                          4424 non-null   int64
 8   Father's qualification                          4424 non-null   int64
 9   Mother's occupation                             4424 non-null   int64
 10  Father's occupation                             4424 non-null   int64
 11  Displaced                                       4424 non-null   int64
 12  Educational special needs                       4424 non-null   int64
 13  Debtor                                          4424 non-null   int64
 14  Tuition fees up to date                         4424 non-null   int64
 15  Gender                                          4424 non-null   int64
 16  Scholarship holder                              4424 non-null   int64
 17  Age at enrollment                               4424 non-null   int64
 18  International                                   4424 non-null   int64
 19  Curricular units 1st sem (credited)             4424 non-null   int64
 20  Curricular units 1st sem (enrolled)             4424 non-null   int64
 21  Curricular units 1st sem (evaluations)          4424 non-null   int64
 22  Curricular units 1st sem (approved)             4424 non-null   int64
 23  Curricular units 1st sem (grade)                4424 non-null   float64
 24  Curricular units 1st sem (without evaluations)  4424 non-null   int64
 25  Curricular units 2nd sem (credited)             4424 non-null   int64
 26  Curricular units 2nd sem (enrolled)             4424 non-null   int64
 27  Curricular units 2nd sem (evaluations)          4424 non-null   int64
 28  Curricular units 2nd sem (approved)             4424 non-null   int64
 29  Curricular units 2nd sem (grade)                4424 non-null   float64
 30  Curricular units 2nd sem (without evaluations)  4424 non-null   int64
 31  Unemployment rate                               4424 non-null   float64
 32  Inflation rate                                  4424 non-null   float64
 33  GDP                                             4424 non-null   float64
 34  Target                                          4424 non-null   object
dtypes: float64(5), int64(29), object(1)
memory usage: 1.2+ MB
```

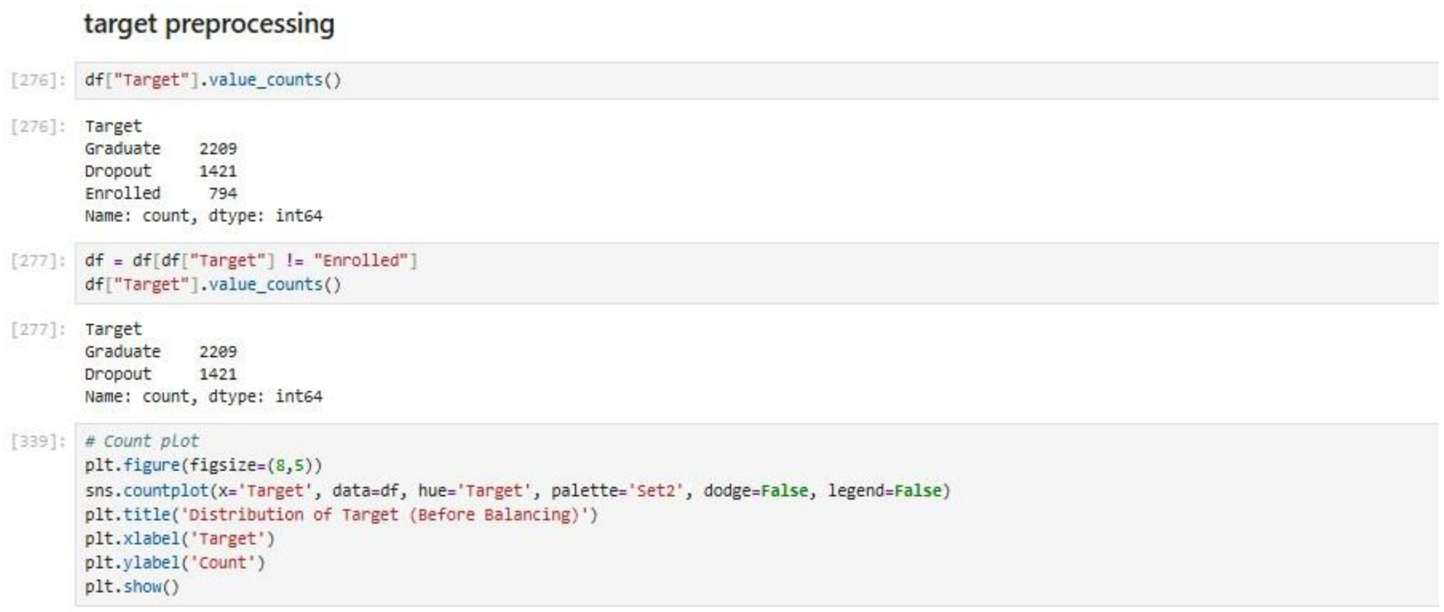
3. Preprocessing & Exploratory Data Analysis (EDA)

Data Preprocessing:

A thorough preprocessing of the raw dataset was conducted to prepare it for modeling. The key steps were as follows:

1. Data Cleaning: The dataset was checked for missing values and duplicates. No null values or duplicate rows were found, indicating a high-quality dataset in terms of completeness.

2. **Dataset Filtering:** The first crucial step was to refine the problem into a binary classification. All instances where the 'Target' was "Enrolled" were removed from the dataset, leaving only students who had either dropped out or graduated.



target preprocessing

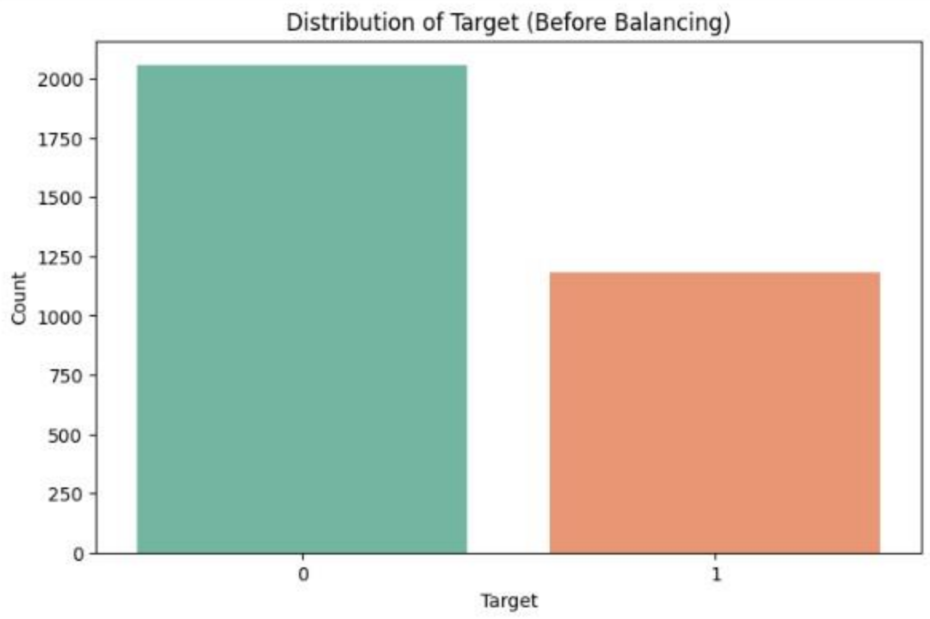
```
[276]: df["Target"].value_counts()
```

```
[276]: Target
       Graduate    2209
       Dropout     1421
       Enrolled     794
       Name: count, dtype: int64
```

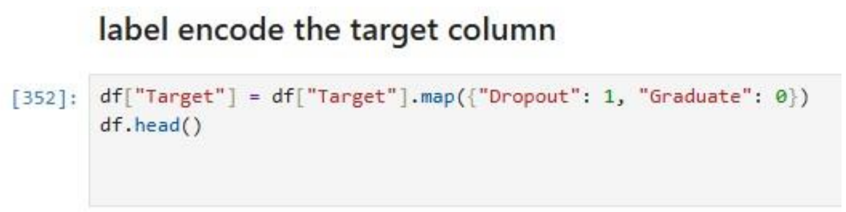
```
[277]: df = df[df["Target"] != "Enrolled"]
       df["Target"].value_counts()
```

```
[277]: Target
       Graduate    2209
       Dropout     1421
       Name: count, dtype: int64
```

```
[339]: # Count plot
       plt.figure(figsize=(8,5))
       sns.countplot(x='Target', data=df, hue='Target', palette='Set2', dodge=False, legend=False)
       plt.title('Distribution of Target (Before Balancing)')
       plt.xlabel('Target')
       plt.ylabel('Count')
       plt.show()
```



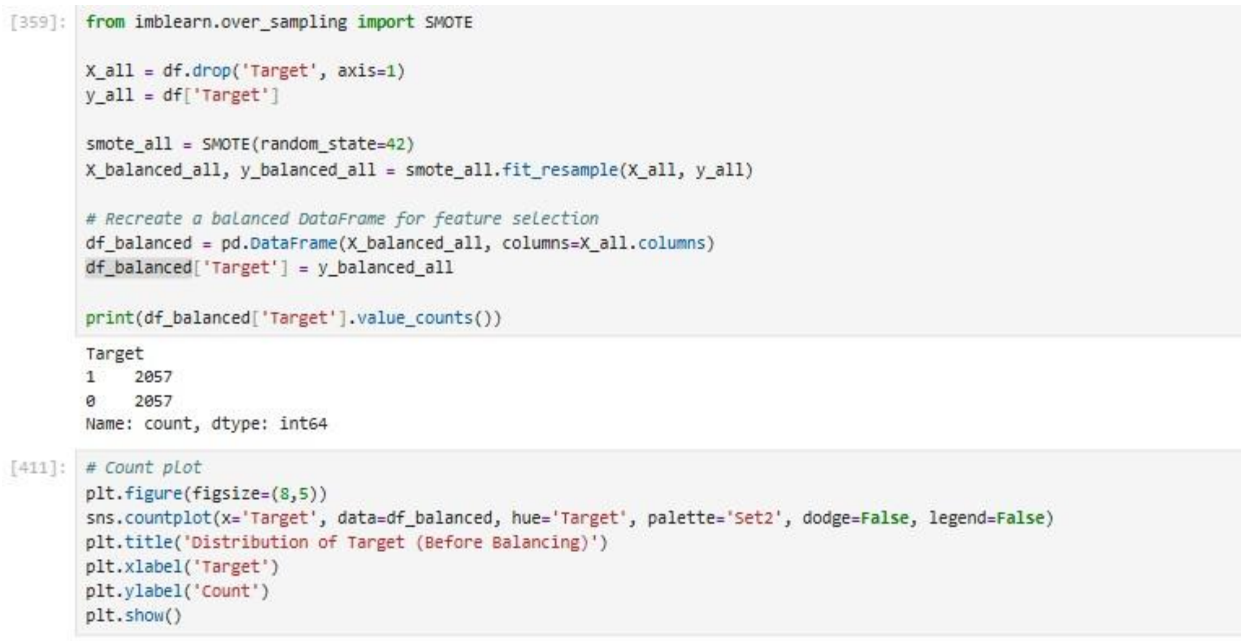
3. **Target Variable Encoding:** The refined categorical target variable was converted into a numerical format. The mapping applied was: **"Dropout" -> 0** and **"Graduate" -> 1**.



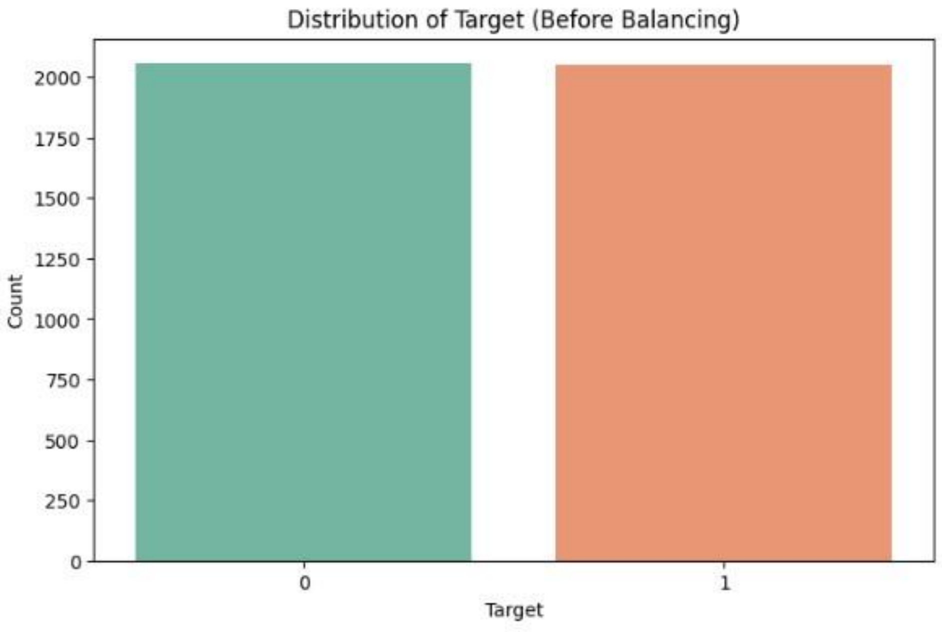
label encode the target column

```
[352]: df["Target"] = df["Target"].map({"Dropout": 1, "Graduate": 0})
       df.head()
```

4. **Target Variable Balancing:** After encoding, the dataset showed an imbalance in the target classes. To address this, the SMOTE (Synthetic Minority Oversampling Technique) was applied to oversample the minority class, ensuring a more balanced distribution for model training.



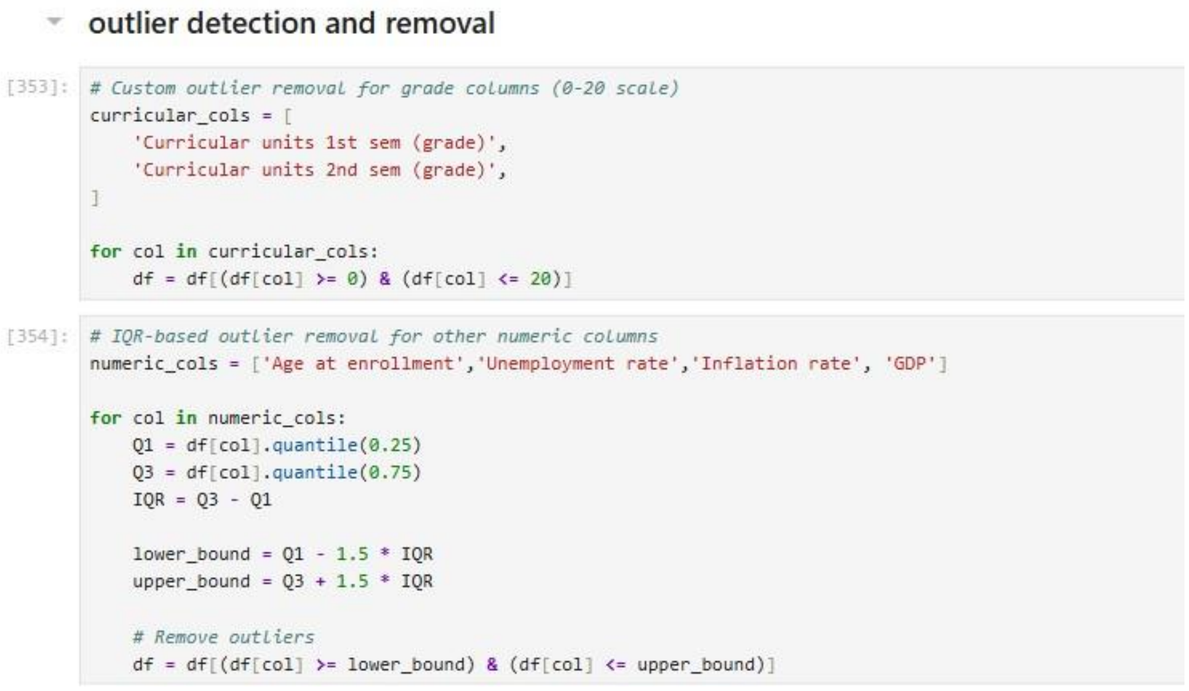
```
[359]: from imblearn.over_sampling import SMOTE

       X_all = df.drop('Target', axis=1)
       y_all = df['Target']

       smote_all = SMOTE(random_state=42)
       X_balanced_all, y_balanced_all = smote_all.fit_resample(X_all, y_all)

       # Recreate a balanced DataFrame for feature selection
       df_balanced = pd.DataFrame(X_balanced_all, columns=X_all.columns)
       df_balanced['Target'] = y_balanced_all

       print(df_balanced['Target'].value_counts())

       Target
       1    2057
       0    2057
       Name: count, dtype: int64

[411]: # Count plot
       plt.figure(figsize=(8,5))
       sns.countplot(x='Target', data=df_balanced, hue='Target', palette='Set2', dodge=False, legend=False)
       plt.title('Distribution of Target (Before Balancing)')
       plt.xlabel('Target')
       plt.ylabel('Count')
       plt.show()
```



5. **Outlier Detection and Removal:**

   Grade Columns: Custom filtering was applied to curricular unit grade columns (both semesters) to ensure they fell within the valid range of 0 to 20. Any entries outside this range were considered erroneous and removed.

   Other Numeric Features: For other continuous variables (Age at enrollment, Unemployment rate, Inflation rate, GDP), the Interquartile Range (IQR) method was used. Data points falling below Q1 - 1.5*IQR or above Q3 + 1.5*IQR were identified as outliers and removed to prevent them from unduly influencing the model.

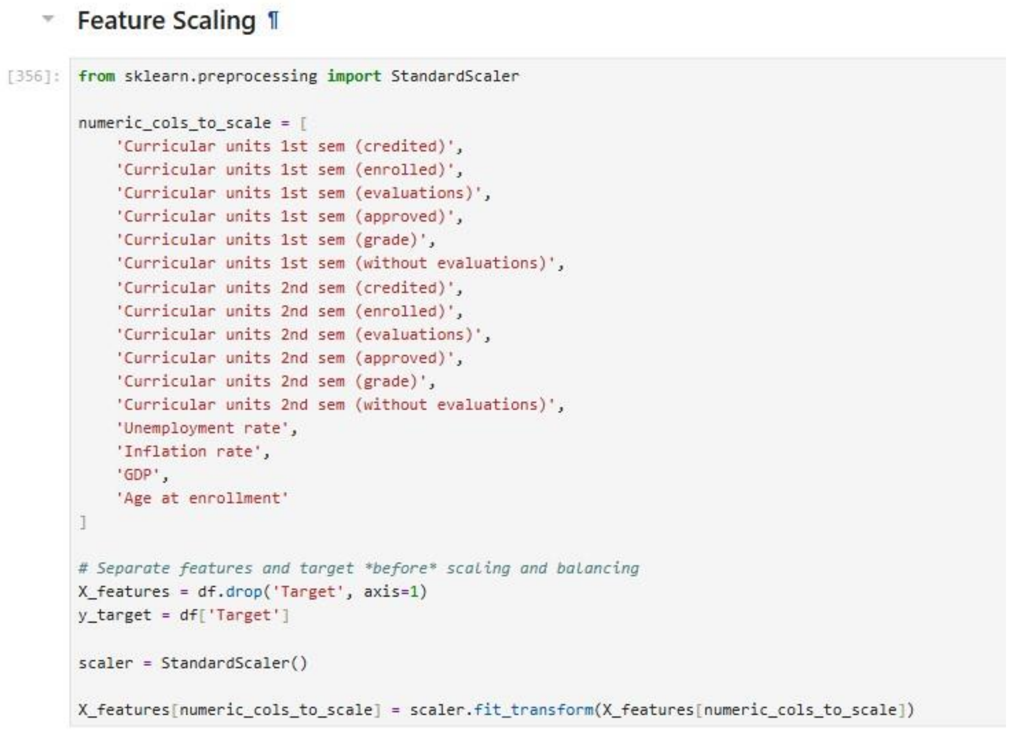


outlier detection and removal

```
[353]: # Custom outlier removal for grade columns (0-20 scale)
curricular_cols = [
    'Curricular units 1st sem (grade)',
    'Curricular units 2nd sem (grade)',
]

for col in curricular_cols:
    df = df[(df[col] >= 0) & (df[col] <= 20)]
```

```
[354]: # IQR-based outlier removal for other numeric columns
numeric_cols = ['Age at enrollment','Unemployment rate','Inflation rate', 'GDP']

for col in numeric_cols:
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Remove outliers
    df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

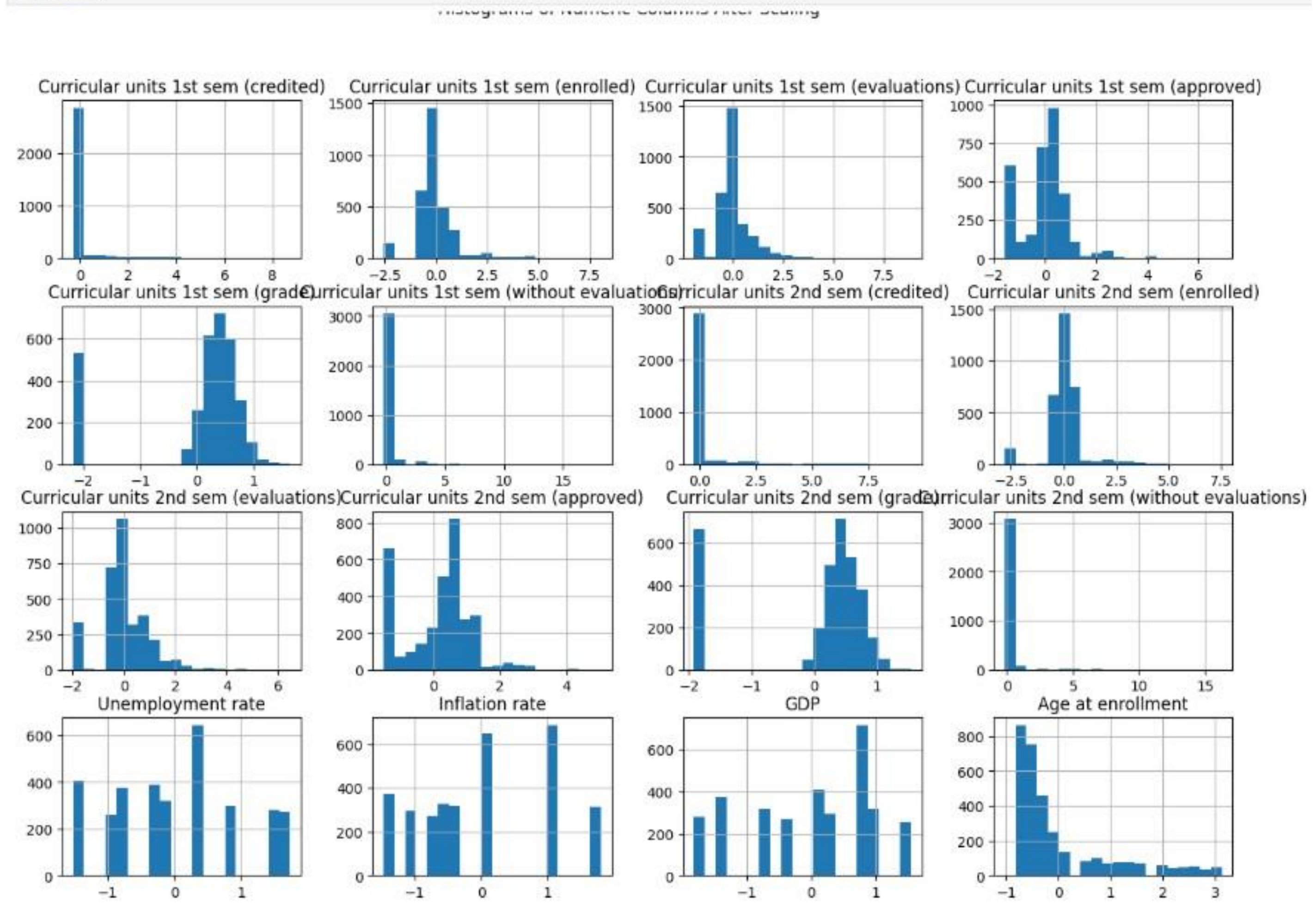
6. **Feature Scaling:** All 16 numeric features (including curricular unit counts, grades, and macroeconomic indicators) were standardized using StandardScaler. This transformation gives each feature a mean of 0 and a standard deviation of 1, which is a critical step for models that are sensitive to the scale of input data, such as Logistic Regression and SVM.



Feature Scaling ¶

```
[356]: from sklearn.preprocessing import StandardScaler

numeric_cols_to_scale = [
    'Curricular units 1st sem (credited)',
    'Curricular units 1st sem (enrolled)',
    'Curricular units 1st sem (evaluations)',
    'Curricular units 1st sem (approved)',
    'Curricular units 1st sem (grade)',
    'Curricular units 1st sem (without evaluations)',
    'Curricular units 2nd sem (credited)',
    'Curricular units 2nd sem (enrolled)',
    'Curricular units 2nd sem (evaluations)',
    'Curricular units 2nd sem (approved)',
    'Curricular units 2nd sem (grade)',
    'Curricular units 2nd sem (without evaluations)',
    'Unemployment rate',
    'Inflation rate',
    'GDP',
    'Age at enrollment'
]

# Separate features and target *before* scaling and balancing
X_features = df.drop('Target', axis=1)
y_target = df['Target']

scaler = StandardScaler()

X_features[numeric_cols_to_scale] = scaler.fit_transform(X_features[numeric_cols_to_scale])
```

```
X_features[numeric_cols_to_scale].hist(bins=20, figsize=(15,10))
plt.suptitle("Histograms of Numeric Columns After Scaling")
plt.show()
```



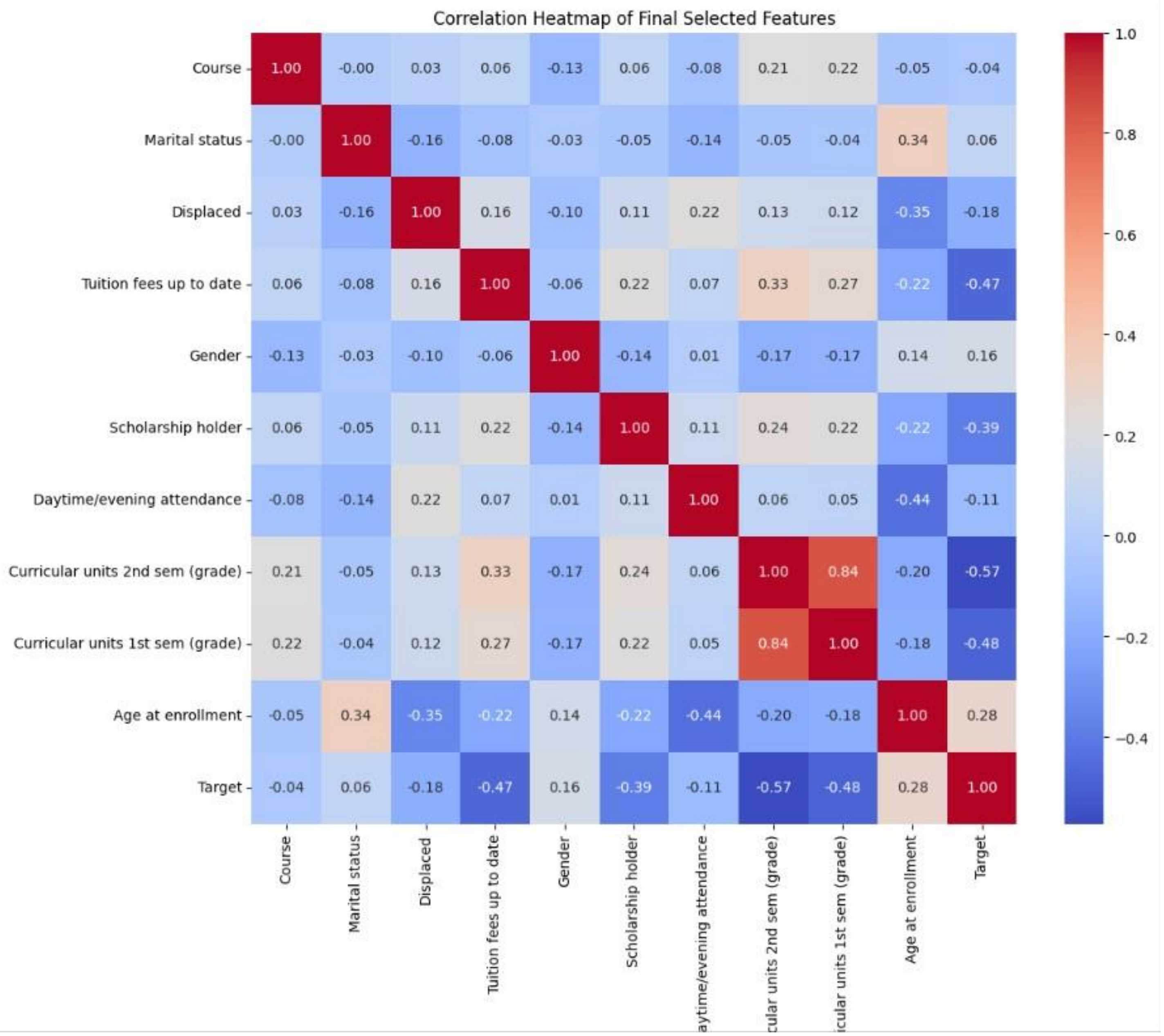
Exploratory Data Analysis (EDA):

With the focus now on dropouts versus graduates, our EDA sought to identify the key differentiators.

After filtering, the distribution of the target variable became more balanced. The dataset now consists of approximately 61% "Graduate" and 39% "Dropout" outcomes. This represents a much healthier class distribution for training a binary classifier compared to the original imbalanced three-class problem.

A correlation heatmap of numerical features highlighted strong positive correlations between academic performance variables. For instance, 'Curricular units 2nd sem (approved)' was highly correlated with the target, indicating that strong academic performance is a key predictor of graduation.

Further visualizations confirmed these trends. Box plots showed a significant difference in the distribution of 'Admission grade' and 'Curricular units 2nd sem (approved)' between students who dropped out and those who graduated. Similarly, students with 'Tuition fees up to date' were far more likely to be in the 'Graduate' category. These insights were instrumental in confirming feature relevance for the binary classification task.



	Course	Marital status	Displaced	Tuition fees up to date	Gender	Scholarship holder	aytime/evening attendance	cular units 2nd sem (grade)	icular units 1st sem (grade)	Age at enrollment	Target
Course	1.00	-0.00	0.03	0.06	-0.13	0.06	-0.08	0.21	0.22	-0.05	-0.04
Marital status	-0.00	1.00	-0.16	-0.08	-0.03	-0.05	-0.14	-0.05	-0.04	0.34	0.06
Displaced	0.03	-0.16	1.00	0.16	-0.10	0.11	0.22	0.13	0.12	-0.35	-0.18
Tuition fees up to date	0.06	-0.08	0.16	1.00	-0.06	0.22	0.07	0.33	0.27	-0.22	-0.47
Gender	-0.13	-0.03	-0.10	-0.06	1.00	-0.14	0.01	-0.17	-0.17	0.14	0.16
Scholarship holder	0.06	-0.05	0.11	0.22	-0.14	1.00	0.11	0.24	0.22	-0.22	-0.39
Daytime/evening attendance	-0.08	-0.14	0.22	0.07	0.01	0.11	1.00	0.06	0.05	-0.44	-0.11
Curricular units 2nd sem (grade)	0.21	-0.05	0.13	0.33	-0.17	0.24	0.06	1.00	0.84	-0.20	-0.57
Curricular units 1st sem (grade)	0.22	-0.04	0.12	0.27	-0.17	0.22	0.05	0.84	1.00	-0.18	-0.48
Age at enrollment	-0.05	0.34	-0.35	-0.22	0.14	-0.22	-0.44	-0.20	-0.18	1.00	0.28
Target	-0.04	0.06	-0.18	-0.47	0.16	-0.39	-0.11	-0.57	-0.48	0.28	1.00

4. Model Design and Implementation

For this project, we implemented and compared six different machine learning models to identify the most effective one for the binary prediction task. The models chosen represent a range of algorithmic approaches:

- Logistic Regression: A linear model serving as a strong baseline.
- Decision Tree: A non-linear, tree-based model known for its interpretability.
- Random Forest: An ensemble method that builds multiple decision trees.
- Gradient Boosting: A powerful boosting ensemble technique.
- K-Nearest Neighbors (KNN): An instance-based learning algorithm.
- Support Vector Machine (SVM): A robust model that finds an optimal separating hyperplane.

To ensure a fair and rigorous comparison, a systematic methodology was applied to all six models explored in this project. This structured process involved the following steps for each algorithm:

1. Libraries: All models were implemented in Python using the scikit-learn library, the industry standard for machine learning.
2. Baseline Model (No Tuning): Each model was first trained with its default parameters to establish a baseline performance. This initial evaluation helped to gauge the model's out-of-the-box effectiveness on our dataset.
3. Hyperparameter Tuning: For more complex models like Random Forest and Gradient Boosting, GridSearchCV was employed to systematically search for the optimal hyperparameter settings.
4. Tuned Model Evaluation: The performance of the tuned models was then rigorously evaluated on the held-out test set to measure their predictive accuracy on unseen data.
5. Validation Method: A robust 5-Fold Cross-Validation strategy was integrated within the hyperparameter tuning process. This ensures that the performance estimates are stable and not dependent on a single random split of the data.
6. Conclusion & Limitations: Finally, the results from all models were compiled and compared to select the best-performing model, while also noting its specific limitations, such as potential for overfitting.

Following this methodology, we implemented and compared six different machine learning models: Logistic Regression, Decision Tree, Random Forest, Gradient Boosting, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM). Based on the comprehensive evaluation, the Random Forest model was determined to be the most effective for this binary prediction task.

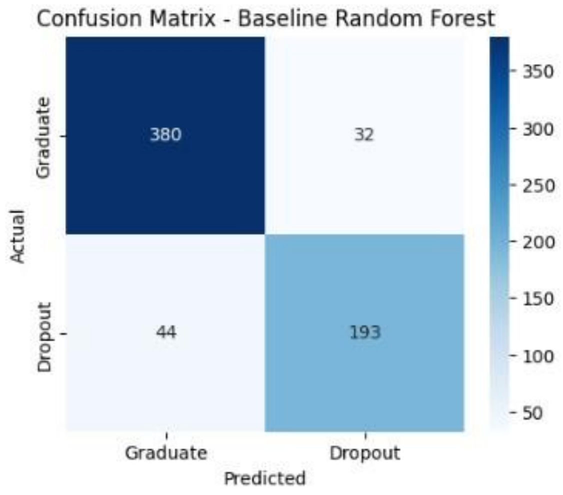
Libraries

```
[375]: from sklearn.ensemble import RandomForestClassifier
```

Baseline Model (No Tuning)

```
[376]: # Baseline Random Forest
       base_model_rf = RandomForestClassifier(n_estimators=100, random_state=42)
       base_model_rf.fit(X_train, y_train)

       y_pred_base_rf = base_model_rf.predict(X_test)

       base_acc_rf = accuracy_score(y_test, y_pred_base_rf)
       print(f"Baseline Accuracy: {base_acc_rf:.4f}")
       print("\nClassification Report (Baseline):\n", classification_report(y_test, y_pred_base_rf))

       cm_base_rf = confusion_matrix(y_test, y_pred_base_rf)
       plt.figure(figsize=(5,4))
       sns.heatmap(cm_base_rf, annot=True, fmt='d', cmap='Blues', xticklabels=['Graduate', 'Dropout'], yticklabels=['Graduate', 'Dropout'])
       plt.xlabel("Predicted")
       plt.ylabel("Actual")
       plt.title("Confusion Matrix - Baseline Random Forest")
       plt.show()
```

```
Baseline Accuracy: 0.8829

Classification Report (Baseline):
               precision    recall  f1-score   support

           0       0.90      0.92      0.91       412
           1       0.86      0.81      0.84       237

    accuracy                           0.88       649
   macro avg       0.88      0.87      0.87       649
weighted avg       0.88      0.88      0.88       649
```



Hyperparameter Tuning (RandomizedSearchCV)

```
[377]: param_dist_rf = {
           'n_estimators': [100, 200, 300, 500],
           'max_depth': [None, 10, 20, 30],
           'min_samples_split': [2, 5, 10],
           'min_samples_leaf': [1, 2, 4],
           'max_features': ['sqrt', 'log2', None],
           'bootstrap': [True, False],
           'class_weight': [None, 'balanced']
       }

       rf_random = RandomizedSearchCV(
           RandomForestClassifier(random_state=42),
           param_distributions=param_dist_rf,
           n_iter=30,
           cv=5,
           scoring='accuracy',
           n_jobs=-1,
           random_state=42,
           verbose=1
       )

       rf_random.fit(X_train, y_train)
       print("Best Parameters Found:", rf_random.best_params_)
       best_rf = rf_random.best_estimator_
```

```
Fitting 5 folds for each of 30 candidates, totalling 150 fits
Best Parameters Found: {'n_estimators': 300, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 20, 'class_weigh
t': None, 'bootstrap': True}
```

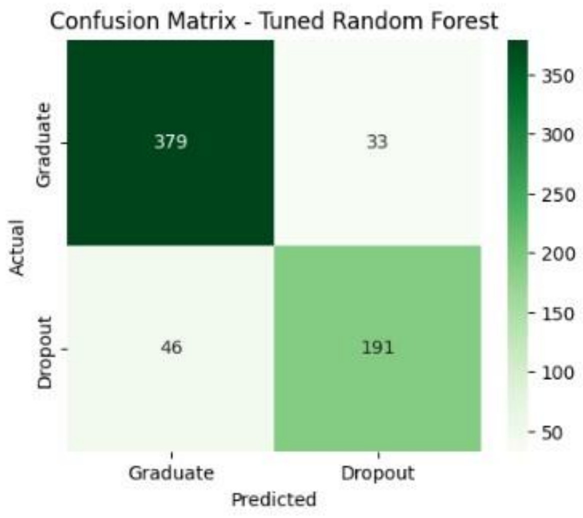
Tuned Model Evaluation

```
[378]: y_pred_tuned_rf = best_rf.predict(X_test)

       tuned_acc_rf = accuracy_score(y_test, y_pred_tuned_rf)
       print(f"Tuned Accuracy: {tuned_acc_rf:.4f}")
       print("\nClassification Report (Tuned):\n", classification_report(y_test, y_pred_tuned_rf))

       # Confusion Matrix - Tuned
       cm_tuned_rf = confusion_matrix(y_test, y_pred_tuned_rf)
       plt.figure(figsize=(5,4))
       sns.heatmap(cm_tuned_rf, annot=True, fmt='d', cmap='Greens', xticklabels=['Graduate', 'Dropout'], yticklabels=['Graduate', 'Dropout'])
       plt.xlabel("Predicted")
       plt.ylabel("Actual")
       plt.title("Confusion Matrix - Tuned Random Forest")
       plt.show()
```

Tuned Model Evaluation

[378]:
```
y_pred_tuned_rf = best_rf.predict(X_test)

tuned_acc_rf = accuracy_score(y_test, y_pred_tuned_rf)
print(f"Tuned Accuracy: {tuned_acc_rf:.4f}")
print("\nClassification Report (Tuned):\n", classification_report(y_test, y_pred_tuned_rf))

# Confusion Matrix - Tuned
cm_tuned_rf = confusion_matrix(y_test, y_pred_tuned_rf)
plt.figure(figsize=(5,4))
sns.heatmap(cm_tuned_rf, annot=True, fmt='d', cmap='Greens', xticklabels=['Graduate', 'Dropout'], yticklabels=['Graduate', 'Dropout'])
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix - Tuned Random Forest")
plt.show()
```

```
Tuned Accuracy: 0.8783

Classification Report (Tuned):
               precision    recall  f1-score   support

           0       0.89      0.92      0.91       412
           1       0.85      0.81      0.83       237

    accuracy                           0.88       649
   macro avg       0.87      0.86      0.87       649
weighted avg       0.88      0.88      0.88       649
```



Validation Method (K-Fold Cross-Validation)

[379]:
```
cv_scores_rf = cross_val_score(best_rf, X_train, y_train, cv=5, scoring='accuracy', n_jobs=-1)

print("Cross-validation scores:", np.round(cv_scores_rf, 4))
print(f"Mean CV Accuracy: {cv_scores_rf.mean():.4f} ± {cv_scores_rf.std():.4f}")
```

```
Cross-validation scores: [0.8647 0.8541 0.8678 0.9088 0.8784]
Mean CV Accuracy: 0.8748 ± 0.0187
```

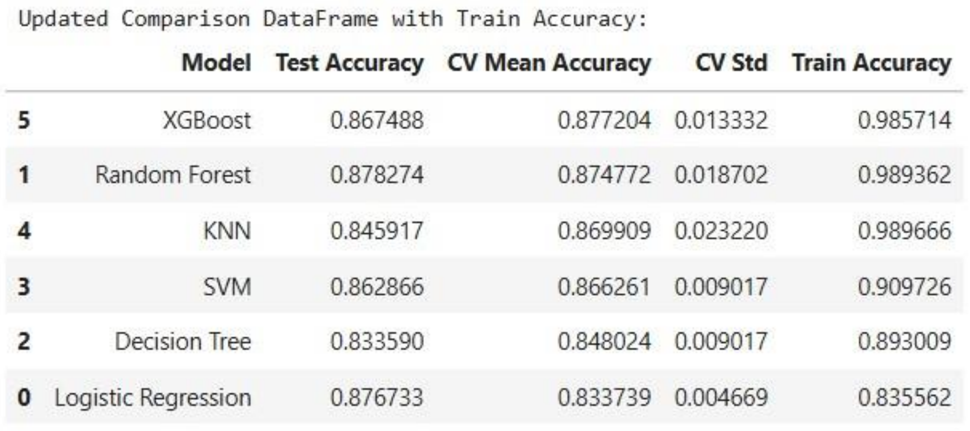
Conclusion & Limitations

[380]:
```
model_results.append({
    'Model': 'Random Forest',
    'Test Accuracy': tuned_acc_rf,
    'CV Mean Accuracy': cv_scores_rf.mean(),
    'CV Std': cv_scores_rf.std()
})
```

5. Evaluation and Comparison

The performance of each model was rigorously evaluated using accuracy, F1-Score, and AUC-ROC. We assessed performance on both the training and testing sets to check for overfitting.

The results for the binary classification task are summarized in the table below:



```
Updated Comparison DataFrame with Train Accuracy:
```

	Model	Test Accuracy	CV Mean Accuracy	CV Std	Train Accuracy
5	XGBoost	0.867488	0.877204	0.013332	0.985714
1	Random Forest	0.878274	0.874772	0.018702	0.989362
4	KNN	0.845917	0.869909	0.023220	0.989666
3	SVM	0.862866	0.866261	0.009017	0.909726
2	Decision Tree	0.833590	0.848024	0.009017	0.893009
0	Logistic Regression	0.876733	0.833739	0.004669	0.835562

6. Ethical Considerations and Bias Mitigation

The development and deployment of a predictive model for student success carry significant ethical responsibilities. As outlined in our AI module, bias in AI occurs when an algorithm produces systematically prejudiced results. This is a primary concern for our project, as the historical data used for training may reflect existing societal or institutional inequalities.

Sources of Potential Bias:

- **Historical Data Bias:** The dataset may contain patterns that reflect past societal prejudices. For example, if students from certain socio-economic backgrounds (indicated by parental occupation or qualifications) have historically faced more systemic barriers to completing their education, the model may incorrectly learn to associate these demographic features with a higher propensity to drop out, rather than treating them as indicators of a need for greater support. This could lead to the creation of a self-fulfilling prophecy, where at-risk students are unfairly labeled and inadvertently disadvantaged.
- **Sampling Bias:** Although the dataset is comprehensive for its source institution, it represents a single Portuguese university. A model trained on this data may not be representative of or fair to student populations in different cultural or economic contexts, limiting its generalizability.

Bias Mitigation Strategies:

To address these ethical challenges and promote a fair, accountable, and transparent system, the following mitigation strategies are essential:

1. **Promoting Transparency with Explainable AI (XAI):** The model should not operate as an unexplainable "black box." By implementing XAI techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations), we can interpret *why* the model makes a specific prediction for an individual. This allows educators to understand the contributing factors (e.g., poor

first-semester grades, non-payment of tuition) and use this information to initiate a constructive and supportive conversation, rather than making an automated, opaque decision.

2. **Conducting Fairness Audits:** It is crucial to rigorously evaluate the model's performance and error rates across different demographic subgroups present in the data (e.g., by gender, nationality, or socio-economic status). If the model's predictive accuracy is significantly lower for one group compared to another, it signals a fairness issue. Identifying such disparate impacts is the first step toward correction, which could involve data re-sampling or applying fairness-aware machine learning algorithms.
3. **Data-Level Interventions:** If a fairness audit reveals significant bias, techniques like re-sampling (e.g., oversampling the underrepresented group within the 'Graduate' class) could be explored to create a more balanced training dataset. This can help prevent the model from developing a prejudice against minority groups.
4. **Ethical Application Framework:** The ultimate purpose of this model must be to serve as a tool for positive intervention, not for penalization. A clear institutional policy must be established to govern its use. This framework should ensure that a prediction of "at-risk" triggers supportive actions—such as offering academic counseling, tutoring, or financial aid resources—rather than being used to restrict a student's opportunities.

By integrating these ethical principles into the model's lifecycle, from data selection to deployment, we can work towards building a trustworthy AI system that supports educational equity rather than perpetuating inequality.

7. Reflections and Lessons Learned

This project provided valuable hands-on experience in the end-to-end machine learning workflow. A key lesson was the impact of problem framing. By converting the task from a multi-class to a binary classification problem, we were able to build a more focused and potentially more accurate model for the most critical business question: will a student drop out or graduate?

One of the main challenges was managing the trade-off between model performance and overfitting. While the Random Forest model gave the best accuracy, its high variance was a concern. This highlighted the importance of cross-validation as a robust measure of a model's true potential. In the future, more extensive hyperparameter tuning or regularization techniques could be explored to mitigate this.

From a team perspective, this project reinforced the importance of clear communication and collaborative problem-solving. In the future, we would aim to delve deeper into model explainability to provide more actionable insights for educators based on the binary predictions.

8. References

[1] The Devastator, "Higher Education Predictors of Student Retention," Kaggle Dataset, 2022. [Online]. Available: https://www.kaggle.com/datasets/thedevastator/higher-education-predictors-of-studentretention

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.

[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[4] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Oversampling Technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, 2002.