Classification

Niels Van den Broeck

April 25, 2025

1 Introduction

In this assignment, I created a Jupyter Notebook to walk through the process of testing different classification models, and then applying the best scoring model to new data. Although the fundamentals are explained in the notebook, I will tell more about the reasoning and discuss the results in this report.

2 Building and Evaluating Classification Models

2.1 Data Preprocessing

Just like last assignment, we have to make sure that the data is not missing any important parts or has corrupted values in any way. Luckily, after inspecting, no values seems to be missing and all values are in correct ranges. I did not find any attributes or groups of attributes that could be useful to group for later use like in previous assignment.

2.2 Feature Selection

To improve the performance of the models, I applied feature selection using the SelectKBest method from scikit-learn, with the f_classif scoring function. This method ranks features based on their statistical relationship with the target variable using ANOVA F-tests. By selecting only the most informative features, the model can focus on the strongest signals in the data while ignoring less relevant attributes.

Initially, I tested with all available features (k='all'), but after experimenting with different values of k, I found that using the top 10 features (k=10) consistently produced better results across most models. This helped reduce overfitting, while also speeding up training and making the models simpler and easier to interpret. The feature selection step was fit only on the training data to avoid data leakage, and the same transformation was applied to the test set.

2.3 Classification Models

To evaluate different approaches for predicting student graduation success, I implemented and compared a range of classification algorithms. Each model was trained using the same preprocessed dataset and evaluated using accuracy, precision, recall, and AUC to ensure fair comparison.

1. **Decision Tree**
   A simple, interpretable model that splits the data into branches based on feature values. It is prone to overfitting but useful for gaining insight into feature importance and decision logic. In Figure 1, you can see the resulting decision tree. Even with only 10 features, the tree has 334 leaves.

2. **k-Nearest Neighbours (k-NN)**
   A non-parametric method that classifies a sample based on the majority class among its k nearest neighbors in feature space. Performance is sensitive to feature scaling and the choice of k.

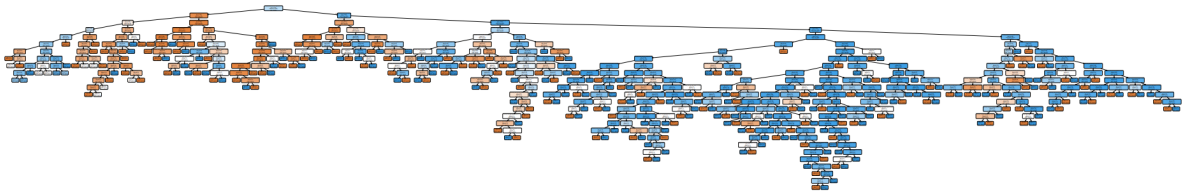


Figure 1: The Decision Tree of the trained model with k = 10 best selected features.

3. **Naïve Bayes**
   Based on Bayes' Theorem, this probabilistic model assumes feature independence. Despite its simplicity, it performs surprisingly well on high-dimensional data and serves as a solid baseline.

4. **Random Forest**
   An ensemble of decision trees built using bootstrap aggregation (bagging). It improves accuracy and reduces overfitting by combining the predictions of multiple diverse trees.

5. **Gradient Boosting**
   A sequential ensemble method that builds trees iteratively, where each new tree corrects the errors made by the previous ones. This often results in high performance but requires careful tuning.

6. **AdaBoost**
   An ensemble method that combines weak learners, focusing more on the misclassified samples at each iteration. It tends to perform well on clean datasets with minimal noise.

7. **XGBoost (Extreme Gradient Boosting)**
   An optimized implementation of gradient boosting that includes regularization, parallel computation, and early stopping.

8. **Logistic Regression**
   A linear model that estimates the probability of graduation using a logistic function. It is highly interpretable and serves as a useful baseline for binary classification tasks.

Each model was evaluated using the same feature selection setup and cross-validated on the training data. Final performance comparisons helped identify the most suitable model for deployment on new student data.

2.4 Results

To evaluate the performance of each classification model, I computed four main metrics: accuracy, precision, recall, and AUC (Area Under the ROC Curve). The results are summarized in Figure 2, which compares all models side by side. Random forest and Gradient Boosting achieved the highest overall accuracy and AUC, making it the most suitable models for deployment on new student data. Logistic Regression and XGBoost also performed well, while simpler models like Naïve Bayes and Decision Tree showed slightly lower precision and recall.

Model	Accuracy
Decision Tree	0.861
k-Nearest Neighbours	0.881
Naïve Bayes	0.861
Random Forest	0.910
Gradient Boosting	0.910
AdaBoost	0.895
XGBoost	0.910
Logistic Regression	0.905

Table 1: Accuracy of each classification model.

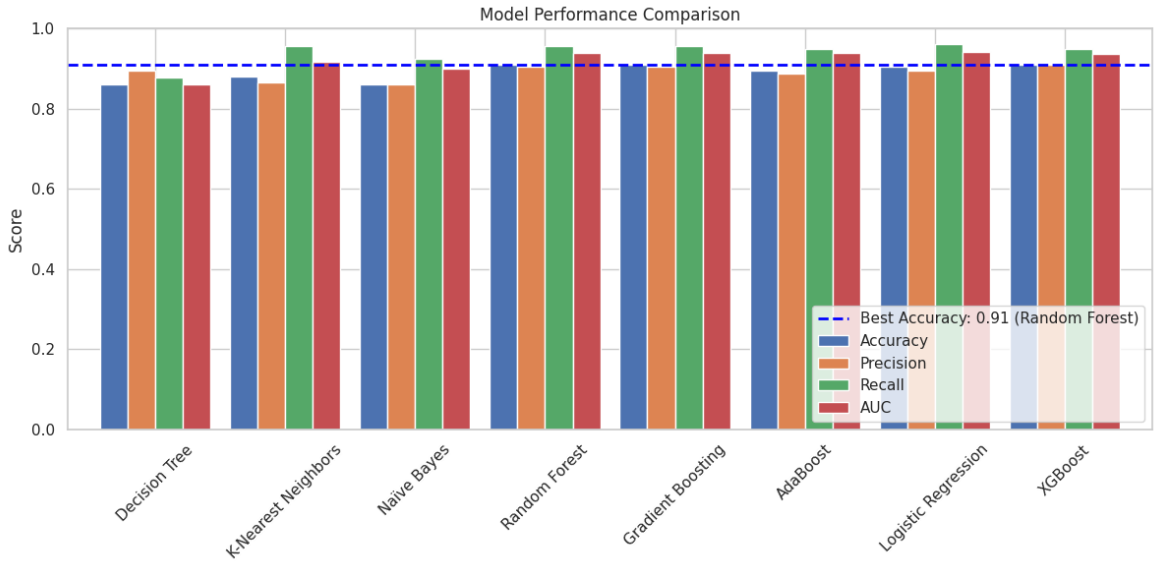
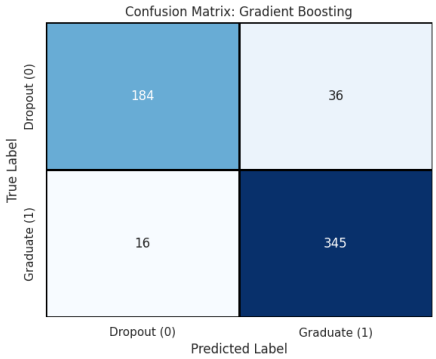
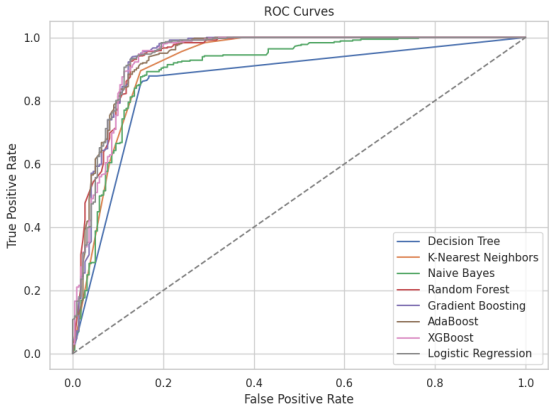


Figure 2: Performance comparison of all classification models across Accuracy, Precision, Recall, and AUC. The dashed red line highlights the best-performing model in terms of accuracy.

To further understand each model's strengths and weaknesses, confusion matrices were analyzed. For example, Figure 3a shows the confusion matrix of the Gradient Boosting classifier. The model demonstrates a strong ability to correctly identify both students likely to graduate and those at risk of dropping out. Additionally, ROC curves were plotted to visualize the trade-off between true positive and false positive rates. ROC curves plot the true positive rates on the y-axis and the false positives on the x-axis. A perfect model would have a 1.0 AUC, hugging the left top part of the plot, while a random classifier follows the dotted line shown in Figure 3b. It is hard to see, but this plot confirms that Gradient Boosting and XGBoost offer the best discriminative performance.



(a) Confusion Matrix for Gradient Boosting Classifier.



(b) ROC curves for the top-performing models.

Figure 3: Model evaluation visualizations

Since we did not want our model to be biased in any harmful way, I excluded the gender and nationality column while training the model. In Figure 4 is shown that one students of gender 0 (in our dataset) more often graduate than drop out, with a significant difference. Now, lets see what the impact is of gender and nationality in this dataset. These results are actually slightly lower which suggests that these features may not contribute meaningful predictive value and could introduce noise or unintended bias into the model. By excluding them, not only was fairness improved by design, but the models also achieved better generalization.

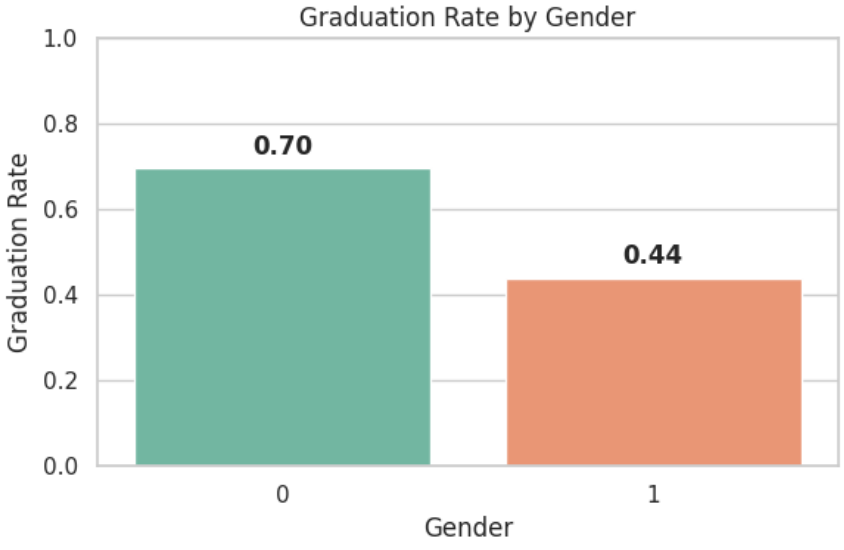


Figure 4: Graduation rate by gender.

Model	Accuracy
Decision Tree	0.843
k-Nearest Neighbours	0.84
Naïve Bayes	0.861
Random Forest	0.895
Gradient Boosting	0.898
AdaBoost	0.882
XGBoost	0.900
Logistic Regression	0.886

Table 2: Accuracy of each classification model with biases.

3 Applying the Model to New Student Data

After evaluating all models on the validation set, the Gradient Boosting classifier was selected as the final model due to its consistently strong performance across multiple metrics, including accuracy, recall, and AUC. This model was then applied to a separate dataset, `graduation_test.csv`, containing data for recently admitted students. As this dataset does not include the target variable, predictions were generated to estimate each student's likelihood of successfully graduating. The model predicted that out of 726 students in the test set, 474 are likely to graduate and 252 may be at risk of dropping out. The final predictions were saved in `predictions.csv`, which includes the student ID and the model's graduation prediction (1 for graduate, 0 for dropout).