Predict Student Dropout And Successes

Aroz Imran,Hashim Ali ,Muneeb Tariq,Hashim Saleem

Eesha tur razia babar

Abstract—

Contents

1. Introduction

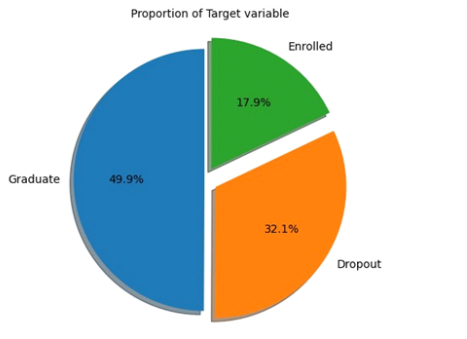
We aim to Predict Students' Dropout and Academic Success using 37 features and 4424 instances: academic path, demographics, and socialeconomic factors. The problem is formulated as a three-category classification task (dropout, enrolled, and graduate) at the end of the normal duration of the course. The dataset consists of discrete and binary labels, making this a Supervised Classification problem. The dataset used in our project has a data split of 80 percent for training and 20 percent for tests. We need to perform rigorous data preprocessing to handle data from anomalies, unexplainable outliers, and missing values.

2. DATA PREPROCESSING :

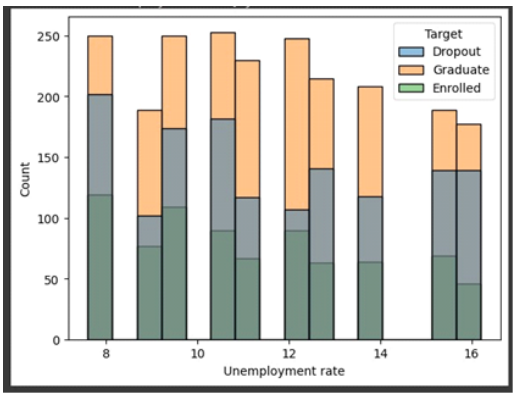
```
df.isnull().sum()

Marital status                                  0
Application mode                                0
Application order                               0
Course                                          0
Daytime/evening attendance\t                    0
Previous qualification                          0
Previous qualification (grade)                  0
Nationality                                     0
Mother's qualification                          0
Father's qualification                          0
Mother's occupation                             0
Father's occupation                             0
Admission grade                                 0
Displaced                                       0
Educational special needs                       0
Debtor                                          0
Tuition fees up to date                         0
Gender                                          0
Scholarship holder                              0
Age at enrollment                               0
International                                   0
Curricular units 1st sem (credited)             0
Curricular units 1st sem (enrolled)             0
Curricular units 1st sem (evaluations)          0
Curricular units 1st sem (approved)             0
Curricular units 1st sem (grade)                0
Curricular units 1st sem (without evaluations)  0
Curricular units 2nd sem (credited)             0
Curricular units 2nd sem (enrolled)             0
Curricular units 2nd sem (evaluations)          0
Curricular units 2nd sem (approved)             0
Curricular units 2nd sem (grade)                0
Curricular units 2nd sem (without evaluations)  0
Unemployment rate                               0
Inflation rate                                  0
GDP                                             0
Target                                          0
dtype: int64
```

The initial step involved examining the dataset for any inconsistencies or anomalies. Fortunately, no missing values were detected in the dataset, which facilitated a streamlined data preparation process. This absence of missing data minimized the need for imputation techniques and ensured the integrity of the dataset.

3. EDA:

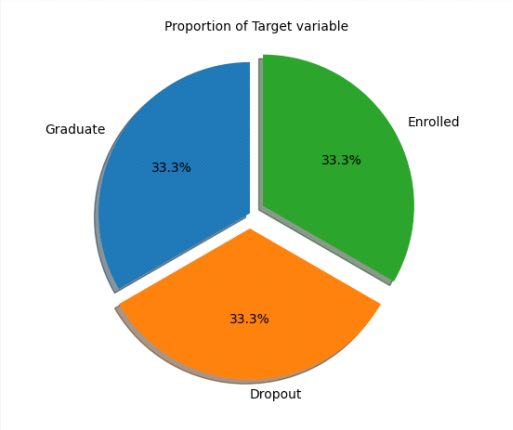


The chart indicates that approximately 49.9 percent of students have successfully graduated, while 17.1 percent are currently enrolled in other educational programs. This distribution highlights that nearly half of the student population in the dataset has attained graduation status.

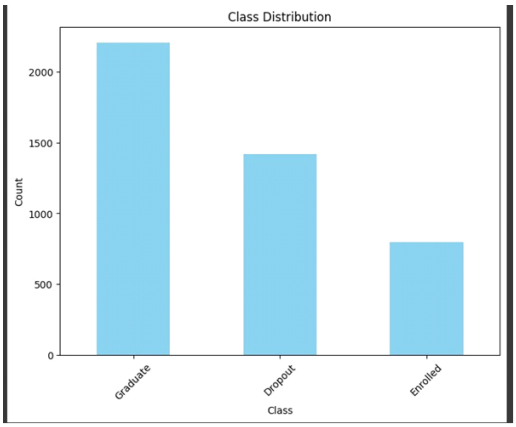


It is surprising to know that the the higher the unemployment is, the less likely that a student will drop out. In fact, this makes sense because if unemployment is low, which means the demand of labor is high, a job seeker has higher chance to have a job. This undermines the need of earning a degree. When unemployment is high, a job seeker is expected to be more competative to be able to have a job.

3.1. Class Imbalance:



4. Class Distribution:



The class distribution shows that the dataset is imbalanced, with significantly more graduates (2209) compared to dropouts (1421) and enrolled students (794). Imbalanced datasets can impact the performance and bias of machine learning models, especially when certain classes are underrepresented.

5. Classification Algorithms :

5.1. LOGISTIC REGRESSION:

Logistic regression is a statistical method used for binary classification tasks, where the goal is to predict whether an instance belongs to one of two classes (e.g., spam or not spam, diseased or not diseased).

It works by modeling the probability that an instance belongs to a particular class given its features. This probability is modeled using the sigmoid function, also known as the logistic function, which transforms the linear combination of features and coefficients into a value between 0 and 1. Mathematically, the sigmoid function is defined as:

$P(y=1|x) \equiv 11+e-(\beta 0+\beta 1x1+\beta 2x2+...+\beta nxn) P(y=1|\mathbf{x}) = 1+e^{-(\beta 0+\beta 1x1+\beta 2x2+...+\beta nxn)}1$

Where:

- $P(y=1|x) P(y=1|\mathbf{x})$ is the probability that the target variable yy is equal to 1 given the features $x\mathbf{x}$.
- ee is the base of the natural logarithm.
- $\beta 0 \beta_0$ is the intercept term.
- $\beta 1, \beta 2, ..., \beta n \beta_1, \beta_2, ..., \beta_n$ are the coefficients associated with each feature $x1, x2, ..., xnx_1, x_2, ..., x_n$ respectively.

During training, logistic regression learns the relationship between the features and the target class by estimating the coefficients ($\beta\beta$) using optimization algorithms like gradient descent. The model assigns appropriate weights to each feature, determining how they influence the predicted probability.

When given a new instance, logistic regression predicts the probability that it belongs to the positive class (e.g., $y=1y=1$). If this probability is above a certain threshold (often 0.5), the instance is classified as belonging to the positive class; otherwise, it's classified as belonging to the negative class. This threshold can be adjusted based on the specific requirements of the task.

Logistic regression is commonly evaluated using metrics like accuracy, precision, recall, and F1 score, which assess how well the model predicts the correct class labels. It's widely used in various fields due to its simplicity, interpretability, and effectiveness in binary classification tasks.

5.2. K-Nearest Neighbor:

K-Nearest Neighbors (K-NN) is an algorithm used for classification and regression tasks. It operates based on the principle that similar instances are often found close to each other in the feature space. Here's how it works:

- **Classification:**
  - When given a new instance to classify, K-NN finds the K nearest neighbors to that instance in the training data based on a distance metric, typically Euclidean distance.
  - The distance between the new instance ($x\mathbf{x}$) and each training instance ($xix\mathbf{x}$) is calculated using the formula: $d(x,xi) = (x1-xi1)2+(x2-xi2)2+\cdots+(xn-xin)2 d(\mathbf{x},\mathbf{x}_i) = (x_1-x_{i1})2+(x_2-x_{i2})2+\cdots+(x_n-x_{in})2$ where nn is the number of features in the dataset.
  - **Voting for Classification:**
    - For classification tasks, K-NN assigns the class label that is most common among the K nearest neighbors to the new instance.
    - If $K=1K=1$, the instance is assigned the class of its nearest neighbor.
    - If $K>1K>1$, the class label is determined by majority voting among the K nearest neighbors.

Regression:

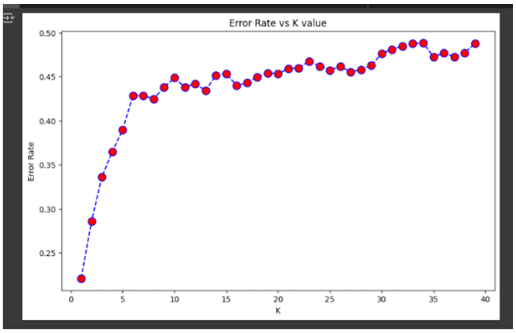
- • For regression tasks, K-NN predicts the target value of the new instance by averaging the target values of its K nearest neighbors.

K-NN doesn't involve explicit training like other algorithms. Instead, it relies on memorizing the training data and finding the closest neighbors when given a new instance. It's simple to understand and implement but can be computationally expensive, especially for large datasets, since it requires calculating distances to all training instances. Additionally, the choice of K and the distance metric can significantly impact its performance.

Accuracy: 0.7790476190476191

	precision	recall	f1-score	support
Dropout	0.82	0.78	0.80	359
Enrolled	0.75	0.91	0.83	351
Graduate	0.77	0.64	0.70	340
accuracy			0.78	1050
macro avg	0.78	0.78	0.77	1050
weighted avg	0.78	0.78	0.78	1050

```
[[280  35  44]
 [  8 321  22]
 [ 53  70 217]]
```



Accuracy: 0.5619047619047619

	precision	recall	f1-score	support
Dropout	0.65	0.56	0.60	359
Enrolled	0.49	0.63	0.55	351
Graduate	0.58	0.49	0.53	340
accuracy			0.56	1050
macro avg	0.57	0.56	0.56	1050
weighted avg	0.57	0.56	0.56	1050

```
[[202 103  54]
 [ 64 220  67]
 [ 44 128 168]]
```

K-Nearst Neighbor does not perform well on this case study

5.3. Decision Tree:

The DecisionTreeClassifier is a machine learning algorithm primarily used for classification tasks. Here's how it works:

- Tree Structure:DecisionTreeClassifier builds a tree-like structure where each internal node represents a "decision" based on a feature, each branch represents the outcome of that decision, and each leaf node represents a class label.
- Splitting Criteria:The algorithm recursively splits the data based on features that best separate the classes. It selects the feature and split point that maximizes the information gain or minimizes impurity at each step. Common impurity measures include Gini impurity and entropy.
- Prediction: When given new data, the DecisionTreeClassifier traverses the tree from the root node down to a leaf node, making decisions based on the feature values of the instance. The final leaf node reached determines the predicted class label.
- Interpretability: Decision trees are interpretable models, meaning the logic behind the classification decisions can be easily understood and visualized. This makes them valuable for explaining the model's predictions to stakeholders.
- Prone to Overfitting: DecisionTreeClassifier is prone to overfitting, especially when the tree becomes too deep or complex. This happens when the model captures noise or idiosyncrasies in the training data rather than general patterns.
- DecisionTreeClassifier provides a straightforward way to classify data and is particularly useful when interpretability is important. However, to mitigate overfitting, techniques like pruning or using ensemble methods such as Random Forests are often employed.

Accuracy: 0.8342857142857143

	precision	recall	f1-score	support
Dropout	0.88	0.84	0.86	359
Enrolled	0.78	0.95	0.86	351
Graduate	0.86	0.71	0.78	340
accuracy			0.83	1050
macro avg	0.84	0.83	0.83	1050
weighted avg	0.84	0.83	0.83	1050

```
[[302  29  28]
 [  6 332  13]
 [ 35  63 242]]
```

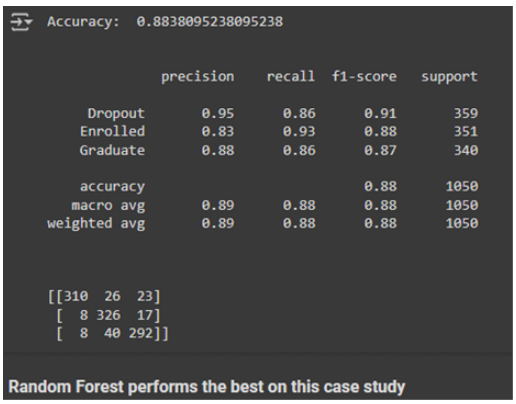
Decision Tree does not perform well on this case study

5.4. RANDOM FOREST:

Random forests, implemented through the RandomForestClassifier algorithm, are a powerful ensemble learning method used for both classification and regression tasks. Here's how it works:

- Multiple Decision Trees: RandomForestClassifier constructs multiple decision trees, each trained on a random subset of the training data and a random subset of features. This randomness helps to create diverse trees.
- Random Sampling: During tree construction, random subsets of the training data (with replacement) are sampled. This process, known as bootstrapping, ensures that each tree sees a slightly different version of the training data.
- Random Feature Selection: At each node of each decision tree, only a random subset of features is considered for splitting. This further introduces diversity among the trees and helps to reduce correlation between them.
- Combining Predictions: For classification tasks, RandomForestClassifier combines the predictions of individual decision trees by taking the mode (most frequent) class prediction across all trees. For regression tasks, it takes the average prediction across all trees.
- Robustness and Generalization: Random forests are robust against overfitting because they aggregate predictions from multiple trees, each trained on different subsets of data and features. This ensemble approach tends to yield more generalized models compared to individual decision trees.
- Feature Importance: RandomForestClassifier can provide estimates of feature importance, indicating the contribution of each feature to the model's predictive performance. This information is valuable for feature selection and understanding the underlying relationships in the data.

Overall, RandomForestClassifier is a versatile and effective algorithm that offers improved performance and robustness over individual decision trees. It's widely used in various machine learning applications due to its ability to handle complex datasets and provide reliable predictions.



5.5. SVM:

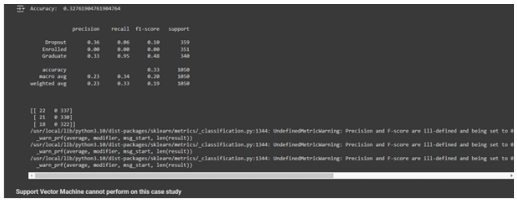
Support Vector Machine (SVM) is a versatile algorithm used for classification, regression, and outlier detection tasks. Here's an explanation of its key components:

- Classification, Regression, and Outlier Detection:
- SVM can be used for various tasks:
- Classification: Predicting the class label of data points.
- Regression: Predicting continuous target values.
- Outlier Detection: Identifying anomalies in the data.
- Finding the Best Hyperplane:
- In classification tasks, SVM finds the hyperplane that best separates classes in the feature space. This hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data points of each class (support vectors).

- For regression tasks, SVM aims to fit a hyperplane that best approximates the relationship between input features and target values.
- Linear and Non-linear Separability:
- SVM can handle both linearly and non-linearly separable data.
- For linearly separable data, the decision boundary is a hyperplane.
- For non-linearly separable data, SVM uses kernel functions to map the data into a higher-dimensional space where classes are linearly separable.

- Effectiveness and Robustness:
- SVM is effective in high-dimensional spaces, making it suitable for data with many features.
- It's robust against overfitting, particularly when using regularization.
- However, SVM requires careful selection of hyperparameters, such as the regularization parameter C and the choice of kernel function.

Overall, SVM is a powerful algorithm known for its ability to handle various types of data and produce accurate predictions. It's widely used in machine learning for both classification and regression tasks



5.6. GRID SEARCH CV:

GridSearchCV is a method used for hyperparameter tuning in machine learning. Here's an explanation of its key components:

- Hyperparameter Tuning:

Hyperparameters are parameters that are not directly learned from the data but rather set before the learning process begins. They control aspects of the model's learning process, such as its complexity or regularization.
Examples of hyperparameters include the regularization parameter, learning rate, and kernel type in Support Vector Machines (SVM), or the number of neighbors k in K-Nearest Neighbors (K-NN).

- Grid Search:

GridSearchCV exhaustively searches through a specified grid of hyperparameters to find the combination that maximizes the model's performance.
The grid consists of possible values for each hyperparameter that the user wants to tune.

- Cross-Validation:

GridSearchCV evaluates each combination of hyperparameters using k-fold cross-validation.
In k-fold cross-validation, the training data is divided into k subsets (or folds). The model is trained k times, each time using k-1 folds for training and the remaining fold for validation. This process ensures that each data point is used for both training and validation, reducing bias in the evaluation.
The cross-validation score (CV Score) is the average performance of the model across all folds.

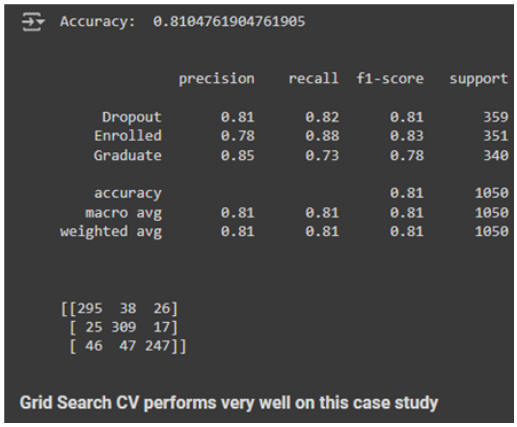
- Best Hyperparameters:

After evaluating all combinations of hyperparameters, GridSearchCV selects the combination that yields the highest average cross-validation score as the best set of hyperparameters.
This best combination is returned to the user, allowing them to use it to train their final model on the entire dataset.

- Usage:

GridSearchCV is commonly used in conjunction with machine learning algorithms to fine-tune their hyperparameters and improve their performance on unseen data.

Overall, GridSearchCV is a valuable technique for optimizing model performance by systematically exploring the hyperparameter space and selecting the best combination based on cross-validation results.



5.6.1. Lime:

LIME works by training a simpler, interpretable model (like linear regression or decision trees) on perturbed versions of the original data around the data point of interest. It then evaluates how changes in the input data affect the output of the simpler model. This provides insights into why a particular prediction was made by the complex model.

LIME is especially useful for black-box models where the internal workings are not easily understandable, such as deep neural networks. It helps data scientists and analysts understand why a model made a certain prediction, which can be crucial for building trust in the model's decisions, debugging, and improving model performance.

	precision	recall	f1-score	support
Dropout	0.64	0.82	0.72	222
Enrolled	0.43	0.41	0.42	150
Graduate	0.89	0.80	0.84	513
accuracy			0.74	885
macro avg	0.65	0.67	0.66	885
weighted avg	0.75	0.74	0.74	885

5.6.2. Shap:

SHAP (SHapley Additive exPlanations) is a method used in machine learning for explaining individual predictions. It's based on game theory concepts and provides a unified approach to interpret the output of any machine learning model. The key use of SHAP is to understand the importance of features in making a particular prediction. Here are some specific use cases and benefits of SHAP:

- 1. Feature Importance: SHAP values quantify the contribution of each feature to the prediction of a particular instance. This helps in understanding which features are driving the model's decision-making process.
- 2. Model Debugging: SHAP can help identify cases where the model behaves unexpectedly. By examining the SHAP values for specific instances, analysts can pinpoint which features are influencing the model's predictions in unexpected ways.
- 3. Model Understanding: SHAP provides intuitive explanations for individual predictions, making it easier for stakeholders to understand why a model makes certain decisions. This is especially important for models with complex architectures, such as neural networks.
- 4. Model Comparison: SHAP values can be used to compare different models and understand the differences in their decision-making processes. This is helpful for model selection and refinement.
- 5. Fairness and Bias Analysis: SHAP can also be used to detect and quantify bias in machine learning models. By examining the SHAP values across different demographic groups, analysts can identify instances where the model's predictions may be biased.

Overall, SHAP is a powerful tool for improving the transparency, interpretability, and trustworthiness of machine learning models, which is crucial for their practical deployment in real-world applications.

6. Conclusion:

On this case study, Random Forest produces the best result, following by Grid Search CV and Logistic Regression. KNN and Decision Tree are not suitable for prediction. Support Vector Machine was not able to produce a feasible result.

7. EDA AND FEATURE ENGINEERING:

7.1. EDA:

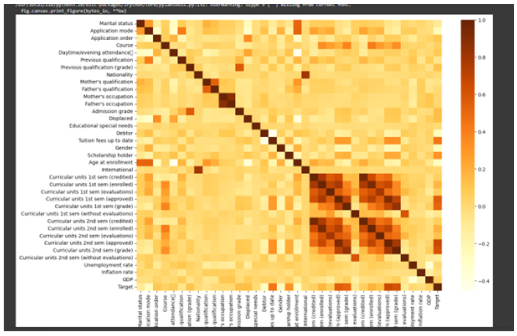
Exploratory Data Analysis (EDA) and Feature Engineering are essential steps in the data preprocessing phase of a machine learning project. Here's an explanation of each:
Exploratory Data Analysis (EDA):

- EDA is the process of analyzing and visualizing data to gain insights and understand its characteristics. It involves:
- Summarizing the main characteristics of the data (e.g., mean, median, distribution).
- Identifying patterns, trends, and relationships between variables.
- Detecting outliers or missing values.
- Visualizing data using graphs, charts, and statistical plots.
- EDA helps data scientists or analysts understand the dataset better, identify potential issues, and inform subsequent steps in the data preprocessing and modeling pipeline.

7.2. Feature Engineering:

- Feature engineering is the process of creating new features or transforming existing features in the dataset to improve the performance of machine learning models. It involves:
- Creating new features that capture relevant information from the existing data. This can include combining or transforming existing features, extracting information from text or categorical variables, or generating new features from domain knowledge.
- Selecting or transforming features to improve model performance. This may involve removing irrelevant or redundant features, scaling or normalizing numerical features, or encoding categorical variables.
- Handling missing values by imputing or encoding them appropriately.
- Feature engineering aims to provide machine learning models with the most informative and relevant features to make accurate predictions.
-

In summary, EDA helps analysts understand the dataset's characteristics and identify potential issues, while feature engineering aims to create or transform features to improve model performance. Together, these steps are crucial for preparing the data for modeling and ensuring the success of machine learning projects.



8. GRADIENT BOOSTING:

Gradient Boosting is a powerful ensemble learning technique used for both regression and classification tasks. Here's an explanation of how it works: **Boosting Ensemble Method:**

- Gradient Boosting is a type of boosting ensemble method. Boosting involves combining multiple weak learners (e.g., decision trees) sequentially to create a strong learner.
- At each step, a new weak learner is trained to correct the errors made by the previous ones.

Gradient Descent:

- Gradient Boosting works by optimizing a loss function using gradient descent.
- Initially, the model starts with an "initial guess" for the target function (e.g., the average target value for regression tasks, or the majority class for classification tasks).
- Subsequent models (weak learners) are trained to minimize the residual errors between the current predictions and the true target values.

Gradient Boosting Algorithm:

- Here's a simplified version of the Gradient Boosting algorithm:
- Initialize the model with an initial guess.
- For each iteration (or boosting round):
- Train a weak learner (e.g., decision tree) to predict the residual errors of the current model.
- Update the model by adding the predictions of the weak learner, weighted by a learning rate, to the current model.
- Repeat the process for a specified number of iterations or until convergence.

Gradient Boosting Trees:

- In practice, Gradient Boosting is often implemented using decision trees as the base learners. This variant is known as Gradient Boosting Trees (GBT) or Gradient Boosted Decision Trees (GBDT).
- Each decision tree is trained to predict the residual errors of the current model, rather than the target values directly.
- The predictions of all the trees are aggregated to make the final prediction.

Benefits:

- Gradient Boosting is highly effective and widely used in practice due to its ability to capture complex relationships in the data and produce accurate predictions.
- It's robust to overfitting, particularly when using regularization techniques such as shrinkage and tree pruning.
- Gradient Boosting can handle a variety of data types and is suitable for both regression and classification tasks.

In summary, Gradient Boosting is a sophisticated ensemble learning technique that combines weak learners sequentially to create a strong predictive model. It's particularly effective for tasks where high predictive accuracy is crucial.