IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of Submission Sunday, July 30, 2023.

Digital Object Identifier

Exploring Naive Bayes Classifier: A Comprehensive Analysis and Optimization for Classification Tasks

PRAJESH SHRESTHA[1], and UJJWAL PAUDEL[1]
[1]Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, Tribhuvan University, Kathmandu, Nepal (e-mail: prajesh.762417@thc.tu.edu.np, ujjwal.762417@thc.tu.edu.np)

ABSTRACT This study explores the application of the Naive Bayes algorithm for classification task. The fundamental concepts of Naive Bayes and its variants, including Gaussian, Categorical, and Bernoulli Naive Bayes are used for performing classification analysis on well-known UCI Predict Students' Dropout and Academic Success dataset. A thorough analysis was done in the dataset and top features were selected using mutual information score before applying naïve bayes classifiers. Results showed percentage accuracy of 87.8%, 86.55% and 89.12% for Gaussian, Categorical and Bernoulli Naïve Bayes respectively.

INDEX TERMS Bernoulli Naive Bayes, Categorical Naive Bayes, Gaussian Naive Bayes, Mutual Information Score

I. INTRODUCTION

NAIVE Bayes is a probabilistic machine learning algorithm used in a wide variety of classification tasks. Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle: every pair of features being classified is independent of each other.

The UCI Predict Students' Dropout and Academic Success dataset is a dataset created from a higher education institution related to students enrolled in different undergraduate degrees. The dataset includes information known at the time of student enrollment (academic path, demographics, and socioeconomic factors) and the students' academic performance at the end of the first and second semesters. By utilizing the Naive Bayes algorithm, one can effectively construct classification models using this dataset. These models have the capability to predict students' likelihood of dropping out or achieving academic success. Naive Bayes' ability to handle the independence assumption between features makes it particularly suitable for this task. [**?**]

II. METHODOLOGY

A. THEORY

Naive Bayes is a classification technique that uses Bayes' theorem to assign probabilities to each possible outcome or class for a given problem instance, represented by a vector of features. Naive Bayes assumes that the features are independent of each other, which is a strong or naive assumption.

1) Bayes Theorem

Bayes' theorem is a mathematical formula that allows us to update our prior beliefs about an event based on new evidence. The prior probability is the initial probability of an event before observing any data, and the posterior probability is the updated probability of an event after observing some data. The formula for Bayes' theorem is:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \tag{1}$$

Where P(A|B) is the posterior probability of A given B, P(B|A) is the likelihood of B given A, P(A) is the prior probability of A, and P(B) is the evidence or marginal probability of B.

In the context of classification, we can consider event A as the class label or the category we want to predict, and event B as the observed features or data. In such case, Baye's theorem can be written as:

$$P(y|x) = \frac{P(x|y) \cdot P(y)}{P(x)} \tag{2}$$

Where, P(y|x) is the conditional probability of class y given the observed features x. P(x|y) is the conditional probability of observing features x given class y. P(y) is the prior probability of class y occurring (i.e., the probability of class y in the dataset). P(x) is the prior probability of observing features x.

2) Assumption of Naïve Bayes

The "naive" assumption in Naive Bayes is that all features are conditionally independent given the class label. This means that the presence or absence of one feature does not affect the presence or absence of other features, given the class label. Mathematically, this can be expressed as:

$$P(x|C_k) = P(x_1|C_k) \cdot P(x_2|C_k) \cdot \ldots \cdot P(x_n|C_k) \quad (3)$$

Where, $P(x|C_k)$ is the joint probability of observing features x given class y. $P(x_i|C_k)$ is the probability of observing feature x_i given class C_k.x_i represents individual features in the feature vector x.

3) Naïve Bayes Classifier

To classify a new data points C_k when the features x_i are given, the naïve bayes classifier assigns the class with the highest posterior probability as the predicted class as follows:

$$\hat{y} = \arg \max_{C_k \in \text{classes}} P(C_k|x)) \quad (4)$$

Using Bayes' theorem and the naive assumption, we can rewrite the classifier's decision rule as:

$$\hat{y} = \arg \max_{C_k \in \text{classes}} \left(P(C_k) \cdot \prod_{i=1}^{n} P(x_i|C_k) \right) \quad (5)$$

Where, $\hat{y}$ is the predicted class label and n is the number of features in the feature vectors. $P(C_k)$ is the prior probability of class C_k. $P(x_i|C_k)$ is the probability of observing feature x_i given class C_k.

4) Gaussian Naïve Bayes

In Gaussian Naive Bayes, it is assumed that the continuous features in the dataset follow a Gaussian (normal) distribution within each class. This assumption allows us to estimate the probabilities $P(x_i|C_k)$ using the mean and standard deviation of feature i within class C_k:

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2} \right) \quad (6)$$

Where, $P(x_i|C_k)$ is the probability of observing feature i given class C_k under the assumption of a Gaussian distribution. $_k$, i is the mean of feature i within class C_k. σ_k, i is the standard deviation of features within class C_k.

5) Categorical Naïve Bayes

In Categorical Naive Bayes, the features are assumed to be generated from a categorical distribution. This assumption is suitable for datasets where the features represent discrete categories or nominal variables, and each feature can take one of a limited set of possible values. For each class C_k , the classifier estimates the probability $P(x_i|C_k)$ for each category x_i of feature i using the relative frequencies of each category within class C_k:

$$P(x_i|C_k) = \frac{\text{Count}(x_i, C_k) + \alpha}{\text{Total Count}(C_k) + \alpha \cdot \text{No. of Cat.}(x_i)} \quad (7)$$

Where: Where, $P(x_i|C_k)$ is the probability of observing feature i given class C_k under the categorical assumption. α is the smoothing parameter (Laplace smoothing) to handle zero probabilities and prevent overfitting.

6) Bernoulli Naive Bayes

Bernoulli Naive Bayes is a variant of the Naive Bayes classifier that is specifically designed for data distributed according to multivariate Bernoulli distributions. In this variant, each feature is assumed to be a binary-valued (Bernoulli, boolean) variable, where it can take only two values, 0 and 1. The decision rule for Bernoulli Naive Bayes is based on the probability $P(x_i|C_k)$ of observing feature xi given class C_k , which is calculated as:

$$P(x_i|C_k) = P(x_i = 1|C_k) \cdot x_i + (1 - P(x_i = 1|C_k)) \cdot (1 - x_i) \quad (8)$$

Where, $P(x_i|C_k)$ is the probability of observing feature x_i in class C_k under the Bernoulli assumption.$P(x_i = 1|C_k)$ is the probability of feature xi being 1 (i.e., its occurrence) given class C_k. x_i is the binary value of feature i (0 or 1) for the data point under consideration.

B. SYSTEM BLOCK DIAGRAM

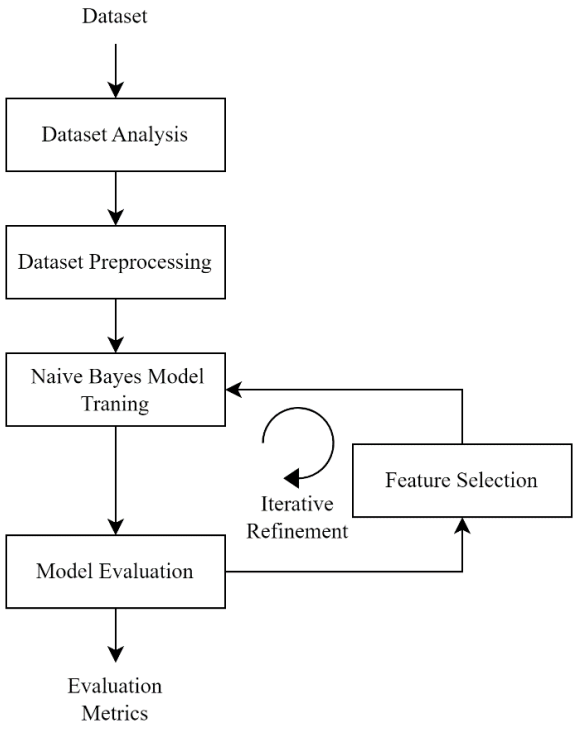

FIGURE 1. System Block Diagram

1) Dataset Analysis

In this step, a comprehensive analysis of the dataset is conducted to understand its structure, characteristics, and potential challenges. Data visualization tools are employed to explore the data and identify patterns, trends, and potential outliers. Table 1 shows the attribute names and their descriptions for the dataset.

2) Dataset Preprocessing

Data preprocessing involves preparing the dataset for model training by applying necessary transformations and handling missing values or outliers. The data is scaled using appropriate techniques to bring all features to a similar scale, ensuring a more effective model training process.

3) Model Training

The dataset is used to train various Naive Bayes models, including Gaussian, Categorical, and Bernoulli Naive Bayes. Each model learns from the data and their respective assumptions to make predictions for classification tasks.

4) Model Evaluation

After training, the Naive Bayes models are evaluated on a separate testing set to assess their performance. Evaluation metrics such as accuracy, precision, recall, and F1-score are used to measure the effectiveness of the models in making accurate predictions.

5) Feature Selection

Feature selection techniques, such as mutual information score, is employed to identify the most relevant and informative features. The selected features are then used to improve the model's efficiency and reduce dimensionality, leading to better classification performance.

C. INSTRUMENTATION TOOLS

1) Pandas and Numpy

- Pandas is employed for data processing, providing data structures like DataFrames that facilitate data manipulation and analysis.
- NumPy serves as the foundation for linear algebra operations, enabling efficient handling of arrays and matrices.

2) Matplotlib and Plotly

- Matplotlib and Plotly are visualization libraries utilized to create informative and visually appealing graphs, charts, and plots, aiding in data exploration and presentation.
- Matplotlib offers various plotting capabilities, while Plotly provides interactive and dynamic visualizations.

3) Seaborn

- Seaborn complements Matplotlib by providing a higher-level interface for statistical graphics, making it easier to create aesthetically pleasing and informative plots.

TABLE 1. Attribute Name and Description

S.N.	Attribute Name	Description
1	Marital status	The marital status of the student.)
2	Application mode	The method of application used by the student.
3	Application order	The order in which the student applied.
4	Course	The course taken by the student.
5	Daytime / evening attendance	Whether the student attends classes during evening.
6	Previous qualification	Qualification obtained by student before enrolling in higher education.
7	Nacionality	The nationality of the student.
8	Mother qualification	The qualification of student's mother.
9	Father qualification	The qualification of student's father.
10	Mother occupation	The qualification of student's mother.
11	Father Occupation	The qualification of student's father.
12	Displaced	Whether the student is a displaced person.
13	Educational special needs	Whether the student has any special educational needs.
14	Debtor	Whether the student is a debtor.
15	Tuition fees upto date	Whether the tuition fees are up to date.
16	Gender	The gender of the student
17	Scholarship holder	Whether the student is a scholarship holder.
18	Age at enrollment	Age of student at the time of enrollment
19	International	Whether the student is an international student.
20	Curricular units 1st sem (Credited)	Number of curricular units credited in the 1st semester
21	Curricular units 1st sem (Enrolled)	Number of curricular units enrolled in the 1st semester
22	Curricular units 1st sem (Evaluations)	Number of evaluations to curricular units in the 1st semester
23	Curricular units 1st sem (Approved)	Number of curricular units approved in the 1st semester
24	Curricular units 1st sem (Grade)	Grade average in the 1st semester (between 0 and 20)
25	Curricular units 1st sem (Without Evaluations)	Number of curricular units without evaluations in the 1st semester
26	Curricular units 2nd sem (Credited)	Number of curricular units credited in the 2nd semester
27	Curricular units 2nd sem (Enrolled)	Number of curricular units enrolled in the 2nd semester
28	Curricular units 2nd sem (Evaluations)	Number of evaluations to curicular units in the 2nd semester
29	Curricular units 2nd sem (Approved)	Number of curricular units approved in the 2nd semester
30	Curricular units 2nd sem (Grade)	Grade average in the 2nd semester (between 0 and 20)
31	Curricular units 2nd sem (Without Evaluations)	Number of curricular units without evaluations in the 2nd semester
32	Unemployment rate	Unemployment rate (%)
33	Inflation rate	Inflation rate (%)
34	GDP	GDP
35	Target	Dropout, Enrolled, or Graduate at the end of the nominal duration of the course

4) Scikit-learn

- Sklearn offers a comprehensive suite of preprocessing and post-processing modules to streamline data preparation and evaluation processes.
- Preprocessing modules include LabelEncoder, StandardScaler, MinMaxScaler, RobustScaler, and OrdinalEncoder, enabling feature scaling and encoding categorical variables.
- Evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix are available from metrics module to assess model performance.
- Sklearn also provides a range of classification algorithms, such as GaussianNB, BernoulliNB and CategoricalNB allowing the implementation of various models for classification tasks.

III. RESULTS AND DISCUSSIONS

A. EXPLORATORY DATA ANALYSIS

The dataset has undergone extensive preprocessing to eliminate any null values, ensuring that it is free from missing data. As depicted in Fig. 2, the heatmap confirms the absence of any null values across all features. Additionally, a thorough assessment for duplications was conducted, and none were identified in the dataset. Consequently, we can confidently assert that the dataset exhibits excellent quality in comparison to others.

The dataset's composition under the target values, which includes categories like Dropout, Graduate, and Enrolled, has been effectively visualized using a pie chart (Fig. 3). Upon analyzing the data presented in the pie chart, it becomes evident that nearly 50% of students are classified as graduates, while approximately 32% have dropedout. The remaining segment represents enrolled students, which was excluded from the modeling process using Naive Bayes, as it does not actively contribute to the classification task.

Figure 4 provides valuable insights into the demographic composition of the dataset. The countplot reveals that a higher number of graduates are female, indicating that females have successfully completed their education at a relatively higher rate. However, it is noteworthy that females also exhibit the highest number of dropouts, albeit with a slight difference compared to males. There could be several plausible explanations for these observations. Firstly, the demographic composition within the dataset might mirror the actual population distribution in the location where the data was collected. In certain regions or communities, it is possible that females tend to have higher educational attainment levels, but they may also face challenges that lead to higher dropout rates. Secondly, the dataset's origin and collection process could influence the observed trends. If the data collection was biased towards certain areas or institutions where females are encouraged and provided with more opportunities to complete their education, it could explain the higher number of female graduates and dropouts. Lastly, societal factors and cultural norms may play a role in shaping these patterns. It's conceivable that in the context of the dataset's location or population, there are specific initiatives or support systems in place to motivate and empower females to pursue education, which might affect both graduation and dropout rates.

In the dataset, students could have six different marital statuses, and Fig. 5 displays a bar plot representing the distribution across all targets. A significant insight from this count plot is that the majority of students are categorized as 'Single,' followed by 'Married' as the second most prevalent status. Interestingly, there are no instances of 'Widower' or 'Legally Separated' students enrolled in any course. The analysis of educational outcomes based on marital status reveals compelling patterns. Single students exhibit a higher graduation rate compared to married students, but they also account for the highest number of dropouts. Similarly, among married students, the number of dropouts exceeds that of graduates, which aligns with the common understanding that marriage may bring increased responsibilities and commitments, potentially leading to a higher likelihood of dropping out from studies. These findings highlight the influence of marital status on educational outcomes and prompt further exploration into the underlying factors contributing to these trends.

In Fig. 6, the courses undertaken by students are presented through a stacked bar chart, which represents the distribution of 'Dropouts,' 'Enrolled,' and 'Graduates.' Several key insights can be drawn from this visualization. Firstly, the Nursing course stands out as the top producer of graduates, aligning with the previous observation that females have the highest number of graduates, and a significant proportion of the Nursing students are females (460 out of 548). This correlation underscores the impact of course selection on graduation rates, with Nursing being a popular and successful choice among students. Conversely, the Management course exhibits the highest number of dropouts, particularly among students enrolled in the evening section. The demanding nature of evening classes could potentially contribute to this trend, leading some students to reconsider their commitment to the course and opt for dropping out. Such insights emphasize the importance of considering course scheduling and its impact on student retention. Another noteworthy finding pertains to the 'Biofuel Production Technologies' course, which shows zero graduates. This observation suggests that either the course might have been discontinued due to various reasons or the students who initially chose this course shifted their interests to other, more promising topics. An interesting yet concerning detail arises from the 'Informatics Engineering' course, where a significant proportion (over 60%) of students have dropped out, and only a small number have successfully graduated. This trend raises questions about the course's level of difficulty and the need for potential improvements or additional support mechanisms to assist students in overcoming challenges.

The dataset primarily focuses on Portuguese students, with a small representation of other nationalities, as evident from Fig. 7's stacked bar plots. The majority of graduates and students belong to Portugal, underscoring the dataset's local context. It is essential to recognize that the insights derived

from this dataset are highly applicable and relevant solely to the education system and students within Portugal. Extrapolating these findings to other countries should be done with caution, as differences in educational policies, cultural factors, and demographics may significantly influence outcomes in distinct contexts.

Figure 8 highlights a significant trend in the dataset, indicating that the majority of students have completed their secondary education, with a notable proportion among them being graduates. This observation aligns with the fact that completing secondary education is a crucial milestone in their nation and is considered the minimum level of literacy attainment. In essence, this level of education in their country might be analogous to high school education in Nepal.

The age distribution plot in Fig. 9 reveals a positively skewed distribution of the age at which students enroll, indicating that the majority of students join at a relatively young age. The mean age at enrollment is approximately 23 years, with a significant concentration of enrollments between the ages of 19 to 25 years. Fig. 10 provides a more detailed composition of student ages through a pie chart for different age groups. Remarkably, students who enrolled under 25 years old and those above 50 years old demonstrated notably higher graduation rates compared to other age groups, as evident from the respective pie charts for age groups 17-19, 20-24, and 50+. Conversely, more than 50% of individuals between the ages of 25 and 50 have experienced dropouts. These observations imply that students are more encouraged to complete their education at an early age, whereas as they progress in their careers, they might face increased responsibilities that lead to a higher likelihood of dropping out. However, at a later stage in life, such as after retirement or during old age, individuals may choose to continue their education, leading to higher graduation rates for those above 50 years old.

The dataset offers valuable insights into the socioeconomic aspect through Fig. 11 and 12, which depict the father's and mother's occupations of the students. A closer examination of Fig. 13, focusing on the snapshot of either the father's or mother's occupation, reveals some noticeable inconsistencies and gender bias. Stereotypical jobs, such as cleaning work, appear more commonly associated with mothers, while roles like military sergeants and army generals are predominantly linked to fathers. Such gender bias, though it may not significantly impact simple classification tasks, can become more problematic in advanced deep neural network models trained to understand natural language, where biases may have a more pronounced effect.

Returning to the context of this dataset, it is intriguing to observe that a majority of students who graduate have parents working in unskilled occupations. This finding suggests that students from families with lower socioeconomic backgrounds might be more motivated to complete their education compared to others. Additionally, when utilizing the dataset for advanced machine learning models, consideration of potential biases becomes paramount to ensure fair and unbiased results in decision-making processes.

Figure 14 presents a box plot depicting the grades obtained by students at the end of the first semester. Notably, both enrolled students and graduates have achieved marks higher than 10 points, indicating a considerable proportion of capable students who have performed well. The presence of outliers in both categories, graduates and enrolled students, suggests that some students have excelled significantly beyond the average grades. Conversely, the box plot also reveals the presence of students who have scored 0 marks, indicating failures, which is typical for dropouts as they are considered failed by default. Similarly, Fig. 15 displays the box plot for the grades obtained by students at the end of the second semester, and the trends observed are consistent with those from the first semester. Once again, both enrolled students and graduates have achieved grades above 10 points, and outliers on both ends signify exceptional performances and failures.

B. FEATURE ENGINEERING

The dataset comprises 35 features with either categorical or continuous data types. To identify the most relevant features for predicting student dropout, a feature selection process was performed using scores like mutual information. Fig. 16 displays a correlational heatmap, indicating the correlations among the features. Notably, a strong correlation is observed between the features 'International' and 'Nacionality,' which is expected and evident due to their inherent relationship.

Another notable group of patterns is found among the continuous attributes related to the first and second semesters. However, the primary focus was to identify features highly correlated with the target variable, which denotes dropout status. Upon observation, no significant signs of correlation between the input features and the output target variable are apparent. This can be attributed to the limitation of using Pearson's correlation, which works best with continuous attributes. Given that most of the input attributes are categorically defined, alternative approaches need to be employed to determine the best set of features for data modeling.

Mutual Information (MI) is a fundamental concept from information theory used to measure the amount of information shared between two random variables. By applying the mutual information score to identify the most important features, Fig. 17 presents the results. According to this scoring metric, the most crucial features for predicting dropout are approvals in the second semester, followed by approvals in the first semester, and then the grades obtained in the second and first semesters, respectively. These features demonstrate a significant dependency or relationship with the target variable, dropout status. To further validate the feature selection process, a Chi-Square test was also conducted. The Chi-Square test calculates p-score values, where a smaller value indicates better feature relevance. Remarkably, the top four features selected using the Chi-Square test aligned with those obtained from the mutual information score. It's important to note that the interpretation of p-score values in the Chi-Square test is opposite to mutual information scores. In the

Chi-Square test, a lower p-score value is considered better, indicating stronger associations between the features and the target variable.

These feature selection metrics were further utilized to determine the best k features for data modeling, as discussed later in this section under the data modeling using Naive Bayes.

C. DATA MODELING

The dataset comprises three target values: 'Dropout,' 'Enrolled,' and 'Graduates.' The primary focus of the analysis is to understand whether students drop out and identify the major contributing factors in case of dropout. Fig. 18 illustrates a pie chart representing the distribution of non-dropout and dropout students after removing the enrolled student data from the dataset. Notably, approximately 60% of the students are classified as graduates, indicating successful completion of their studies.

The dataset was divided into training and test sets with an 80:20 ratio, following the specifications provided by the dataset author. Subsequently, various Naive Bayes classifiers available in the sklearn library were applied to these distinct sets. This process involved training the classifiers on the training set and evaluating their performance on the test set.

1) Gaussian Naïve Bayes

The classification report obtained for the GaussianNB model is presented in Table 2. The achieved accuracy of 85.40% indicates that the model performs reasonably well, with a higher number of non-dropouts correctly classified compared to dropouts. Despite trying various types of normalization and standardization techniques such as MinMaxScaler, StandardScaler, and RobustScaler from the sklearn library, it is surprising to observe that these transformations did not significantly impact the accuracy and other scoring metrics. This lack of effect could be attributed to the fact that Gaussian Naive Bayes assumes a normal (Gaussian) distribution for the features.

When the input features already approximately follow a Gaussian distribution, additional standardization or normalization might not yield substantial improvements. Additionally, GaussianNB is tailored for continuous features, and the original scaling of the features might already be suitable for the classifier. As a result, the model's decision boundaries and probability estimates remain relatively unchanged, leading to similar accuracy scores before and after applying these transformations.

TABLE 2. Classification Report: Gaussian Naive Bayes Model

Class	Precision	Recall	F1-Score	Support
Non-Dropout	86.23%	90.85%	88.48%	448.0
Dropout	83.86%	76.62%	80.08%	278.0
Accuracy			85.40%	726.0
Macro Avg.	85.04%	83.73%	84.28%	726.0
Weighted Avg.	85.32%	85.40%	85.26%	726.0

Also, cross-validation was performed to obtain a better accuracy score, but for a fold of 44, the maximum accuracy that could be obtained was 84.36%, which is lower than what is normally obtained. The reduced accuracy in cross-validation with Gaussian Naive Bayes can be attributed to the limited amount of data available in each fold. Since GaussianNB assumes a normal distribution for features, having smaller subsets (folds) might not provide enough data points to accurately estimate the parameters of the Gaussian distribution. As a result, the model's ability to make accurate predictions is hampered, leading to a decrease in accuracy compared to a single train-test split.

2) Multinomial Naïve Bayes

Table 3 presents the classification report using Multinomial NB. As observed, the accuracy of this classifier decreased compared to the Gaussian NB. This decrease in accuracy can be attributed to the violation of assumptions that MultinomialNB() sets. Specifically, MultinomialNB assumes that features have a multinomial distribution, which is well-suited for discrete count data such as word frequencies in text classification. However, if the dataset contains continuous or normally distributed features, as assumed by GaussianNB, using MultinomialNB may lead to a less appropriate model and, consequently, result in lower accuracy.

TABLE 3. Classification Report: Multinomial Naive Bayes Model

Class	Precision	Recall	F1-Score	Support
Non-Dropout	81.39%	92.56%	86.62%	430.0
Dropout	86.50%	69.26%	76.92%	296.0
Accuracy			83.06%	726.0
Macro Avg.	83.94%	80.91%	81.77%	726.0
Weighted Avg.	83.47%	83.06%	82.66%	726.0

3) Bernoulli Naïve Bayes

Table 4 presents the classification results obtained using the Bernoulli Naive Bayes model on the test set. Remarkably, this model performed significantly better when used with StandardScaler. When using MinMaxScaler, the accuracy was around 83%, but after applying StandardScaler, the accuracy increased by 5%. This improvement can be attributed to many reasons.

Bernoulli Naive Bayes assumes that the features have a binary distribution, representing the presence or absence of a feature. StandardScaler is more suitable for binary features because it centers the data around zero and scales it based on standard deviation, aligning better with the binary nature of the features. On the contrary, MinMaxScaler scales the data to a fixed range (usually [0, 1]), which may not be as appropriate for binary features, as it doesn't consider their inherent binary nature.

StandardScaler is less sensitive to outliers compared to MinMaxScaler. In the context of Bernoulli Naive Bayes, outliers in binary features could lead to misinterpretations, as the model assumes a binary distribution. StandardScaler's ability to center the data and scale it based on standard deviation helps in reducing the impact of outliers, making the model

more robust to extreme values in the binary features. MinMaxScaler, on the other hand, stretches the range of values and can magnify the effect of outliers, potentially negatively affecting the model's performance.

In this scenario, Bernoulli Naive Bayes outperformed Gaussian Naive Bayes due to the dataset's binary feature distribution. Bernoulli Naive Bayes is better suited for datasets with binary or discrete features, while Gaussian Naive Bayes assumes a continuous Gaussian distribution, which may not hold true for datasets predominantly consisting of binary features.

TABLE 4. Classification Report: Bernoulli Naive Bayes Model

Class	Precision	Recall	F1-Score	Support
Non-Dropout	88.25%	92.56%	90.35%	430.0
Dropout	88.36%	82.09%	85.11%	296.0
Accuracy			88.29%	726.0
Macro Avg.	88.31%	87.33%	87.73%	726.0
Weighted Avg.	88.30%	88.29%	88.22%	726.0

4) Categorical Naïve Bayes

Table 5 presents the classification report for evaluating the Categorical Naive Bayes model on the test sets. In this setup, some continuous attributes were transformed into categories using equal-width binning techniques. The accuracy score obtained from this model shows an improvement over Gaussian Naive Bayes but falls short compared to Bernoulli Naive Bayes.

Categorical Naive Bayes is well-suited for datasets with features that have a categorical (discrete) distribution, making it appropriate for data with discrete categories. While this modeling assumption can work effectively for certain datasets, it might not be the optimal choice if the dataset contains more binary features (representing presence or absence of attributes). In such cases, Bernoulli Naive Bayes, which is specifically designed for binary features, tends to perform better due to a more appropriate modeling assumption.

TABLE 5. Classification Report: Categorical Naive Bayes Model

Class	Precision	Recall	F1-Score	Support
Non-Dropout	87.61%	89.63%	88.60%	347.0
Dropout	85.00%	82.26%	83.61%	248.0
Accuracy			86.55%	595.0
Macro Avg.	86.30%	85.94%	86.11%	595.0
Weighted Avg.	86.52%	86.55%	86.52%	595.0

From this comparison, it can be inferred that the chosen dataset exhibits characteristics of a categorical or discrete distribution rather than a normal distribution. However, it also demonstrates a prevalence of binary features. As a result, the performance of Categorical Naive Bayes falls between that of Gaussian Naive Bayes, which is less suitable for categorical features, and Bernoulli Naive Bayes, which is well-tailored for binary features.

5) Hybrid Naïve Bayes

The dataset contains a mix of both continuous and categorical features. To better understand the distribution of each feature, Fig. 19 displays the distribution plots of the selected features, revealing that some features roughly align with a normal distribution. Subsequently, the dataset was separated into categorical and continuous features based on this analysis. The classification task was performed on the features resembling a normal distribution, and the results are presented in Table 6. Additionally, the classification report for the categorical features is shown in Table 7. Remarkably, the performance of the model on the features with a normal distribution outperformed the results obtained from the categorical features. Upon merging the results from both sections, an overall accuracy score of 81.69% was achieved.

TABLE 6. Hybrid Naïve Bayes: Gaussian NB Section

Class	Precision	Recall	F1-Score	Support
Non-Dropout	80.93%	94.98%	87.39%	657.0
Dropout	89.56%	65.81%	75.87%	430.0
Accuracy			83.44%	1087.0
Macro Avg.	85.25%	80.40%	81.63%	1087.0
Weighted Avg.	84.35%	83.44%	82.84%	1087.0

TABLE 7. Hybrid Naïve Bayes: Categorical NB Section

Class	Precision	Recall	F1-Score	Support
Non-Dropout	78.67%	83.11%	80.83%	657.0
Dropout	71.76%	65.58%	68.53%	430.0
Accuracy			76.17%	1087.0
Macro Avg.	75.22%	74.34%	74.68%	1087.0
Weighted Avg.	75.94%	76.17%	75.96%	1087.0

The decrease in accuracy is likely due to the suboptimal selection of features for both categorical and continuous data. To improve results, more advanced feature selection techniques and merging methods should be considered to better capture underlying patterns and relationships in the dataset, leading to enhanced model performance and more accurate predictions.

6) Feature Selection: Top-K Features using Mutual Information Score

Since the dataset wasn't suitable for performing correlation operations with Pearson's correlation, an alternative method was employed using Mutual Information Classifier. This involved fitting the training data to the mutual information classifier from the sklearn library. The classifier provides scores for each feature, indicating the relevance of that particular feature. The mutual information score was used as the scoring function along with SelectKBest from sklearn to identify the best k features from the dataset.

TABLE 8. Gaussian Naive Bayes: Top - K Feature Selected

Class	Precision	Recall	F1-Score	Support
Non-Dropout	85.62%	95.58%	90.33%	430.0
Dropout	92.28%	76.69%	83.76%	296.0
Accuracy			87.88%	726.0
Macro Avg.	88.95%	86.14%	87.05%	726.0
Weighted Avg.	88.34%	87.88%	87.65%	726.0

Various values of k were explored, and during the mod-

eling of the training set with Gaussian Naive Bayes, it was unexpectedly discovered that the highest accuracy score was achieved with just the top 2 features: grades from the first semester and second semester. The evaluation report for this analysis is presented in Table 8. The success of Gaussian Naive Bayes with only these two features can be attributed to their normal distribution, which aligns well with the assumptions of the Gaussian Naive Bayes algorithm.

TABLE 9. Bernoulli Naive Bayes: Top - K Feature Selected

Class	Precision	Recall	F1-Score	Support
Non-Dropout	88.24%	94.19%	91.11%	430.0
Dropout	90.64%	81.76%	85.97%	296.0
Accuracy			89.12%	726.0
Macro Avg.	89.44%	87.97%	88.54%	726.0
Weighted Avg.	89.21%	89.12%	89.02%	726.0

Table 9 displays the classification report for the best result achieved on this dataset using any kind of Naive Bayes algorithm. By utilizing only the top 9 features, the Bernoulli Naive Bayes model demonstrated the highest performance. This outcome can be attributed to the fact that the majority of features among the top k features exhibit a binary distribution, which aligns well with the assumptions of the Bernoulli Naive Bayes algorithm. As a result, the model excels when focusing on these binary features, leading to superior predictive accuracy and overall performance.

7) Logistic Regression

To assess the effectiveness of Naive Bayes algorithms on the dataset, Logistic Regression was applied as a comparative benchmark. The results in Table 10 indicate that Logistic Regression outperforms Naive Bayes classifiers. The superiority of Logistic Regression can be attributed to several factors.

TABLE 10. Classification Report: Logistic Regression Model

Class	Precision	Recall	F1-Score	Support
Non-Dropout	90.53%	95.58%	92.99%	430.0
Dropout	93.01%	85.47%	89.08%	296.0
Accuracy			91.46%	726.0
Macro Avg.	91.77%	90.53%	91.04%	726.0
Weighted Avg.	91.54%	91.46%	91.40%	726.0

First, Naive Bayes classifiers assume conditional independence between features given the class label, which may not hold true in real-world datasets, leading to potential loss of predictive accuracy.In contrast, Logistic Regression does not impose this strong assumption and can capture complex relationships between features and the target variable, making it more flexible in handling correlated features.

Second, Logistic Regression is a linear model capable of modeling non-linear relationships between features and the target. This allows it to learn intricate decision boundaries, even in scenarios where classes may overlap or data may not be well-separated. The added flexibility of Logistic Regression contributes to its better performance in cases where Naive Bayes classifiers may struggle to capture complex decision boundaries accurately. Overall, the results demonstrate that Logistic Regression is a more effective choice than Naive Bayes for this particular dataset, considering its ability to handle feature dependencies and model complex relationships.

IV. CONCLUSION

In conclusion, we observed how naïve bayes algorithm leverages probabilities to make predictions and effectively handles high-dimensional feature spaces. We delved into the core principles of Naive Bayes, understanding its underlying assumptions, and exploring variants like Gaussian, Categorical, and Bernoulli Naive Bayes. All of them proved to be efficient and suitable for diverse classification tasks. However, naive bayes classifiers were outperformed by Logistic Regression because of the assumption of feature independence which may not hold true for real-word correlated data. Moreover, we learned about the impact of feature selection, feature engineering, and the importance of model evaluation to fine-tune the classifier's performance.

REFERENCES

[1] V. Realinho, M. Vieira Martins, J. Machado, and L. Baptista, "Predict students' dropout and academic success," 2021. [Online]. Available: *UCI Machine Learning Repository*. [Accessed: July 29th, 2023]. URL: https://doi.org/10.24432/C5MC89

PRAJESH S. is an enthusiastic undergraduate student pursuing Computer Engineering at Thapathali College, Institute of Engineering (IOE), Nepal. Curious about the vast world of Machine Learning and Artificial Intelligence, Prajesh enjoys immersing himself in learning and experimenting with diverse ML/AI models. He is fascinated by the potential applications of these technologies and aspires to utilize them for addressing real-world challenges. For any queries or potential collaborations, you can contact Prajesh at prajesh.762417@thc.tu.edu.np.

UJJWAL P. is a dedicated undergraduate student currently pursuing Computer Engineering at Thapathali College, Institute of Engineering (IOE), Nepal. With a keen interest in Machine Learning and Artificial Intelligence, he embraces the role of a passionate learner, constantly exploring and experimenting with various ML/AI models. Driven by a passion to make meaningful contributions to the AI field, he is committed to continuous growth and learning in this exciting and dynamic domain. For any inquiries or potential collaborations, you can reach out to Ujjwal at ujjwal.762416@thc.tu.edu.np.

APPENDIX A
PLOTS AND FIGURES

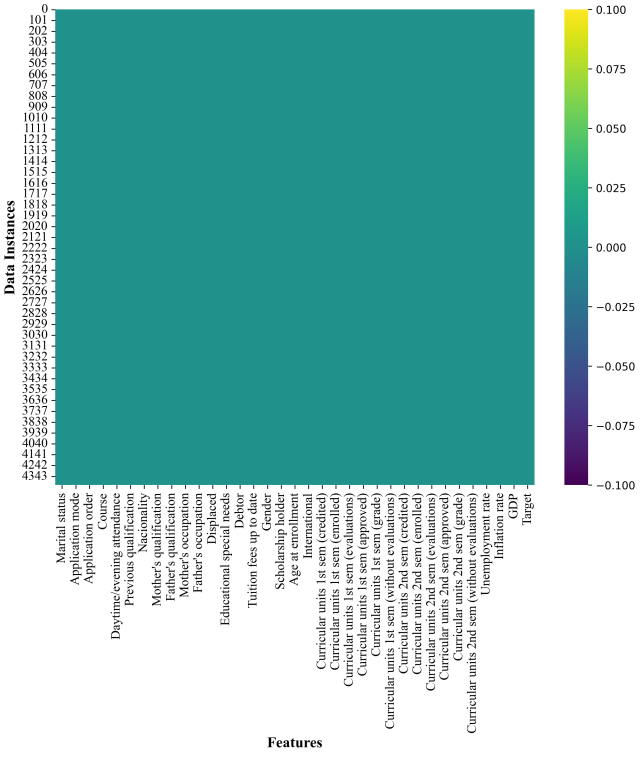

FIGURE 2. Null Values Heatmap

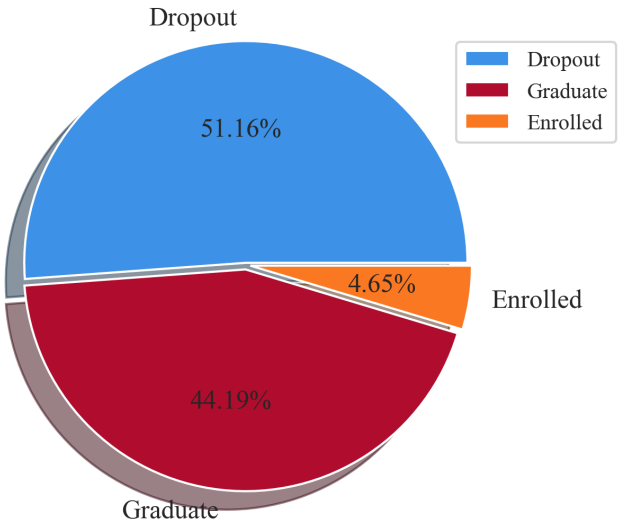

FIGURE 3. Pie Chart: Dataset Composition

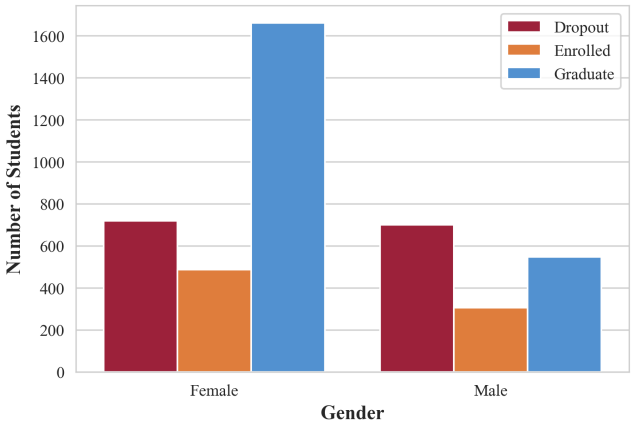

FIGURE 4. Bar Plot: Gender vs Target

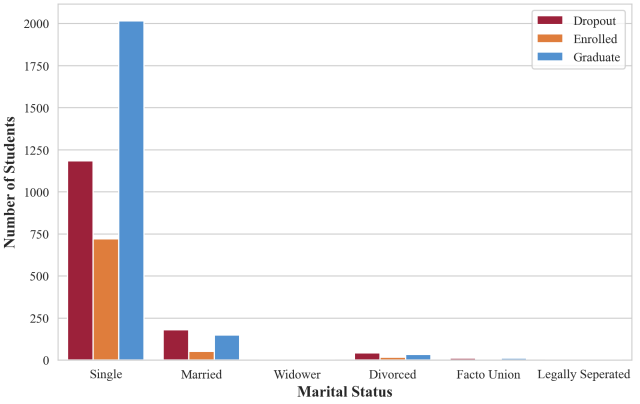

FIGURE 5. Bar Plot: Marital Status vs Target

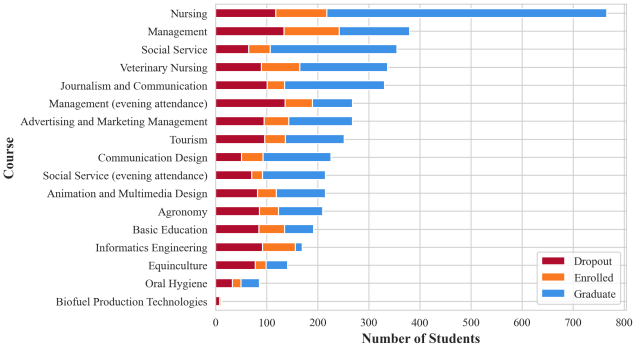

FIGURE 6. Bar Plot: Course vs Target

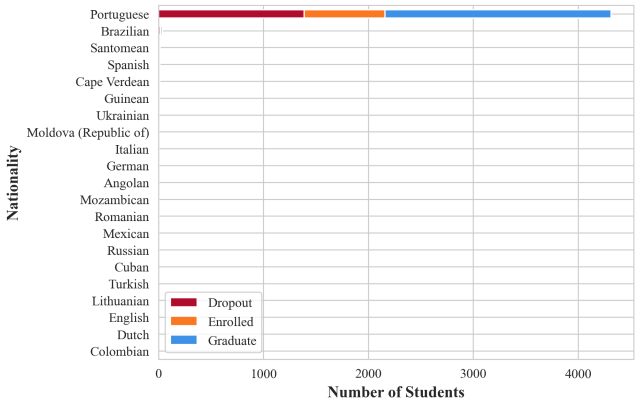

FIGURE 7. Stacked Bar Plot: Nationality vs Target

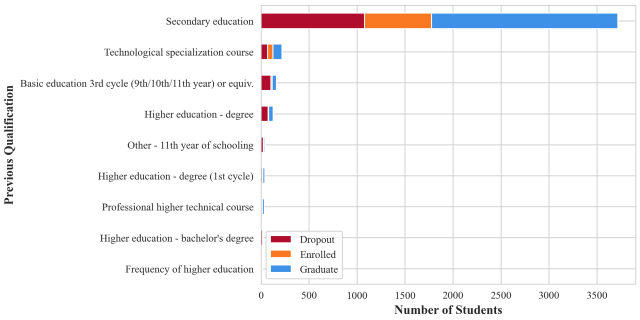

FIGURE 8. Stacked Bar Plot: Previous Qualification vs Target

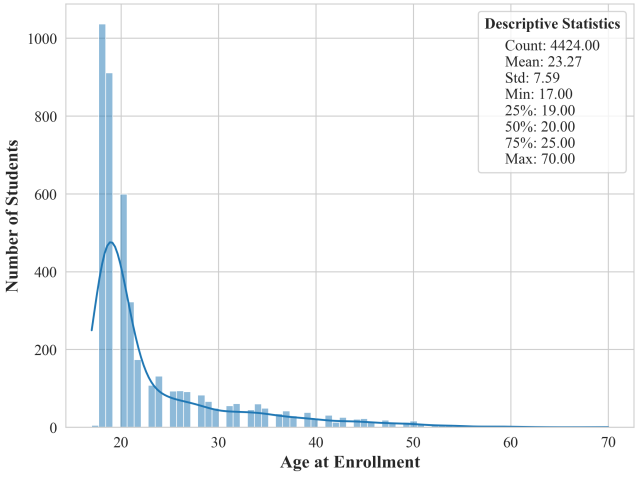

FIGURE 9. Distribution Plot of Age at Enrollment

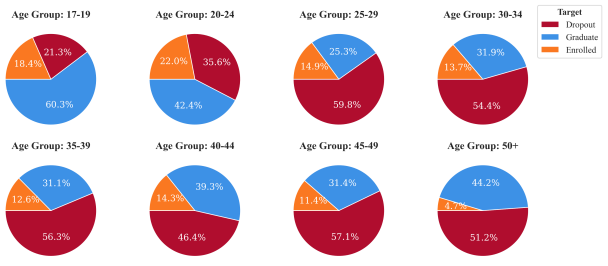

FIGURE 10. Pie Chart: Multiple Age-Group Composition

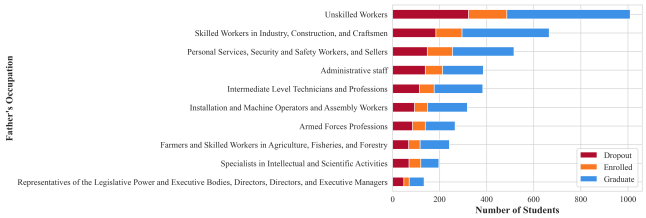

FIGURE 11. Stacked Bar Plot: Father Occupation vs Target

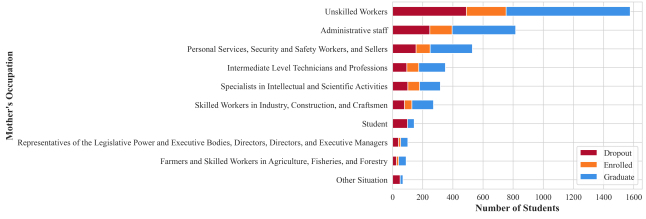

FIGURE 12. Stacked Bar Plot: Mother Occupation vs Target

	Occupation Index	Occupation Name	Father	Mother
74	191	Cleaning workers	False	True
66	143	Data, accounting, statistical, financial services, and registry-related operators	False	True
24	134	Intermediate level technicians from legal, social, sports, cultural and similar services	True	False
65	141	Office workers, secretaries in general, and data processing operators	False	True
75	192	Unskilled workers in agriculture, animal production, fisheries, and forestry	False	True
37	174	Skilled workers in electricity and electronics	True	False
41	183	Vehicle drivers and mobile equipment operators	True	False
34	163	Farmers, livestock keepers, fishermen, hunters and gatherers, subsistence	True	False

FIGURE 13. Occupation Difference Between Father and Mother

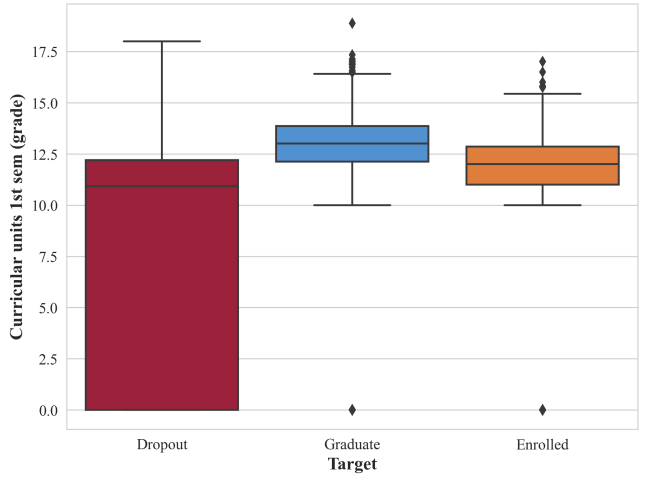

FIGURE 14. Box Plot: First Semester Grade

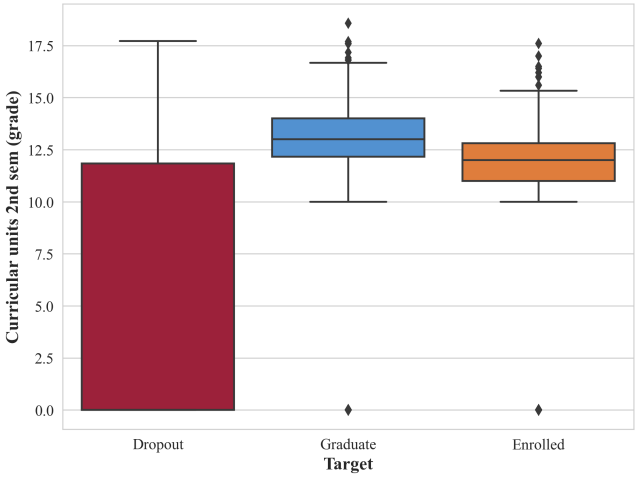

FIGURE 15. Box Plot: Second Semester Grade

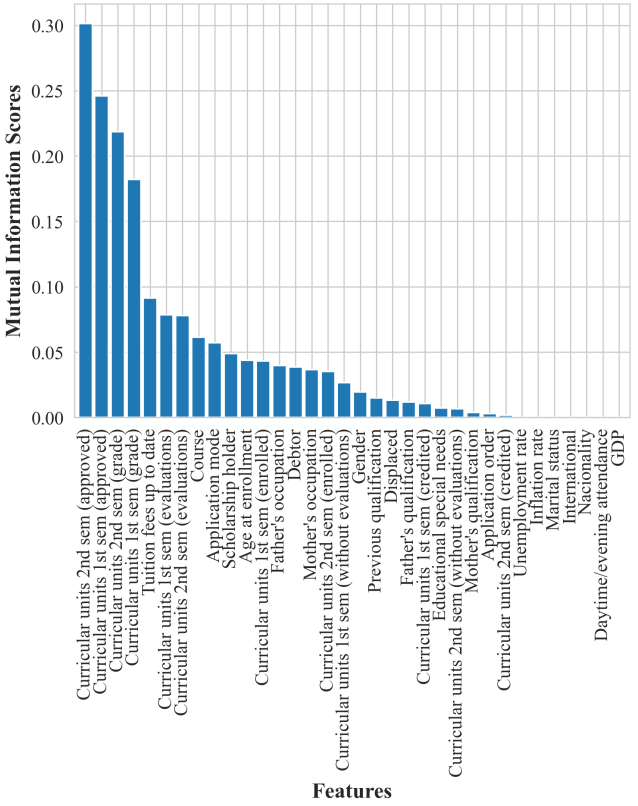

FIGURE 17. Bar Plot: Mutual Information Score

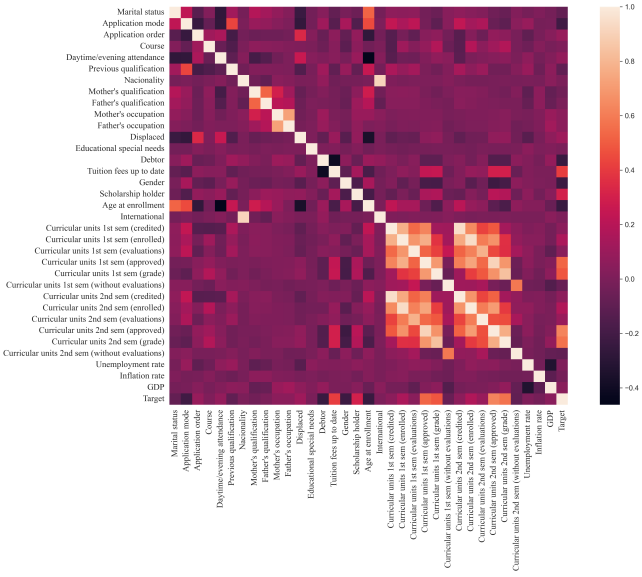

FIGURE 16. Correlational Heatmap

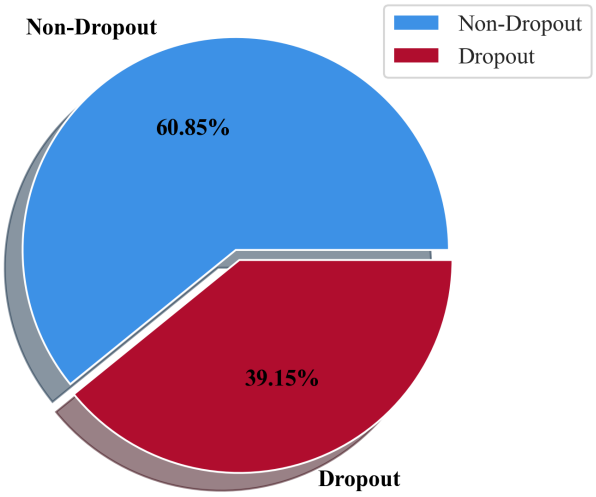

FIGURE 18. Pie Chart: Dropout and Non-Dropout

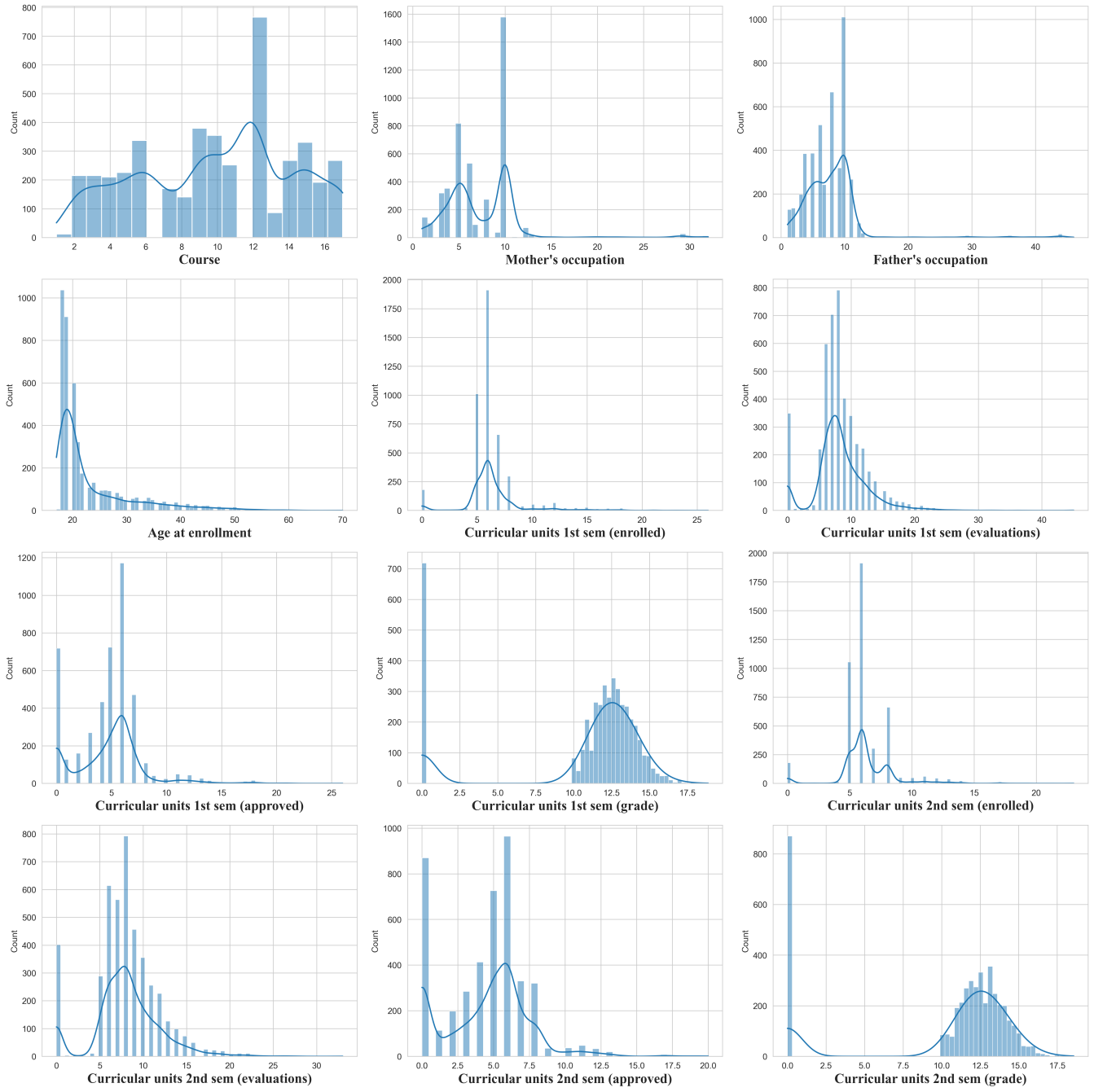

FIGURE 19. Feature Distributions

APPENDIX A
CODE

1) Imported Modules and Functions

```
# Data Processing; Linear algebra; Utility
import pandas as pd # data processing
import numpy as np # for linear algebra
from pprint import pprint
from tqdm import tqdm

# Data Visualization Library
import matplotlib.pyplot as plt
from matplotlib_venn import venn2
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as ex
from matplotlib.font_manager import FontProperties

# Preprocessing and Post-Ops Modules from Sklearn
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, precision_score, recall_score, f1_score, ConfusionMatrixDisplay, PrecisionRecallDisplay, RocCurveDisplay
from sklearn.model_selection import train_test_split, GridSearchCV, RandomizedSearchCV, cross_val_score, StratifiedKFold
from sklearn.feature_selection import mutual_info_classif, chi2, SelectKBest
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, RobustScaler, OrdinalEncoder

# Models and Classifiers from Sklearn
from sklearn.naive_bayes import GaussianNB, MultinomialNB, BernoulliNB, CategoricalNB
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

2) Dataset Exploration

```
df = pd.read_csv('dataset.csv')
pprint(df.columns)
fathers_occu_ = df_father['Occupation Name'].to_list()
mothers_occu_ = df_mother['Occupation Name'].to_list()
father_set = set(fathers_occu_)
mother_set = set(mothers_occu_)
print("Number of occupations for father: ", len(father_set))
print("Number of occupations for mother: ", len(mother_set))
print("Intersected occupations between father and mother: ", len(father_set.intersection(mother_set)))
print("Difference in occupation from father POV: ", len(father_set.difference(mother_set)))
print("Difference in occupation from mother POV: ", len(mother_set.difference(father_set)))

data = []
for value in df_father[['Occupation Index', 'Occupation Name']].iterrows():
    occu_name = value[1][1]
    if occu_name in df_mother['Occupation Name'].to_list():
        data.append((value[1][0], value[1][1], True, True))
    else:
        data.append((value[1][0], value[1][1], True, False))

for value in df_mother[['Occupation Index', 'Occupation Name']].iterrows():
    occu_name = value[1][1]
    if occu_name in df_father['Occupation Name'].to_list():
        data.append((value[1][0], value[1][1], True, True))
    else:
        data.append((value[1][0], value[1][1], False, True))
columns = ['Occupation Index', 'Occupation Name', 'Father', 'Mother']
df_combined = pd.DataFrame(data, columns=columns)

# df_combined.set_index('Occupation Index', inplace=True)
df_combined.drop_duplicates(subset = ['Occupation Index', 'Occupation Name', 'Father', 'Mother'], inplace = True)
df_combined.head()

diffn_occups_ = df_combined[((df_combined['Father'] == True) & (
```

```
    df_combined['Mother'] == False)) | ((
    df_combined['Father'] == False) & (
    df_combined['Mother'] == True))]
diffn_occups_.sort_values('Occupation
    Index').sample(8)
```

3) Dataset Analysis

```
plt.figure(figsize=(10, 8), dpi = 90)
font = FontProperties(family='Times New
    Roman', size = 12)
font_label = FontProperties(family='
    Times New Roman', size = 14, weight =
     "bold")
font_xticks = FontProperties(family='
    Times New Roman', size = 12)
font_yticks = FontProperties(family='
    Times New Roman', size = 12)

ax = sns.heatmap(df.isnull(), cmap = '
    viridis')
plt.tick_params(axis='both', which='both
    ', labelsize = 10)

plt.xticks(rotation = 'vertical', font =
     font_xticks)
plt.yticks(font = font_yticks)
plt.xlabel("Features", font = font_label
    )
plt.ylabel("Data Instances", font =
    font_label)
plt.show()

df.describe()
df.duplicated().head()

target_counts = df['Target'].
    value_counts()
font = FontProperties(family='Times New
    Roman', size = 12)
font_label = FontProperties(family='
    Times New Roman', size = 10)

plt.figure(figsize=(4, 4))
colors = [(0.24, 0.57, 0.90), (0.69,
    0.05, 0.18) , (0.98, 0.47, 0.13)]
plt.pie(target_counts, labels =
    target_counts.index, explode = (0.01,
     0.02, 0.02), autopct='\%1.2f\%\%',
    colors = colors, textprops = {'family
    ':'Times New Roman', 'size' : 12},
    shadow = True)
plt.axis('equal')
legend_1 = plt.legend(prop = font_label)
legend_1.set_bbox_to_anchor((0.90, 0.98)
    )
plt.show()

plt.figure(figsize = (6, 4), dpi = 90)
sns.set_style('whitegrid')
colors = [(0.69, 0.05, 0.18), (0.98,
    0.47, 0.13), (0.24, 0.57, 0.90)]
sns.countplot(data = df, x = 'Gender',
    hue = 'Target', hue_order=['Dropout',
     'Enrolled', 'Graduate'], palette =
    colors)

plt.xticks(ticks=[0,1], labels = ['
    Female','Male'], font = font_ticks)
plt.yticks(font = font_ticks)

plt.ylabel('Number of Students', font =
    font_label)
plt.xlabel('Gender', font = font_label)
plt.legend(prop = font_ticks)
plt.savefig('./Figures/gender and target
    .png', dpi = 300, bbox_inches = '
    tight')
plt.show()

df['Marital status'].value_counts()
ms_target_df = pd.crosstab(index = df['
    Marital status'], columns=df['Target'
    ], margins=True)
ms_target_df.drop('All',axis = 1,
    inplace = True)
ms_target_df.drop('All',axis = 0,
    inplace = True)
ms_target_df.plot(kind='line')
plt.show()

font = FontProperties(family='Times New
    Roman', size = 12)
font_label = FontProperties(family='
    Times New Roman', size = 12, weight =
     'bold')
font_ticks = FontProperties(family='
    Times New Roman', size = 10)
plt.figure(figsize = (8, 5), dpi = 90)

colors = [(0.69, 0.05, 0.18), (0.98,
    0.47, 0.13) , (0.24, 0.57, 0.90)]
sns.countplot(data = df, x = 'Marital
    status', hue = 'Target', hue_order =
    ['Dropout', 'Enrolled', 'Graduate'],
    palette = colors)

plt.xlabel('Marital Status', font =
    font_label)
plt.ylabel('Number of Students', font =
    font_label)
```

```
plt.xticks(ticks=[0,1,2,3,4,5], labels = ['Single','Married','Widower','Divorced','Facto Union','Legally Seperated'], font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)
plt.show()

sns.countplot(x=df['Application mode'])
plt.show()

application_mode_target_df = pd.crosstab(df['Target'],df['Application mode'], margins=True)
course_target_df = pd.crosstab(df['Target'], df['Course'],margins=True)

maxm_dropout_course = course_target_df.drop('All', axis = 1).loc['Dropout'].idxmax()
maxm_graduated_course = course_target_df.drop('All', axis = 1).loc['Graduate'].idxmax()
maxm_enrolled_course = course_target_df.drop('All', axis = 1).loc['Enrolled'].idxmax()

print(f"Student taking course : '{course_list[maxm_dropout_course - 1][1]}' are most dropouts.")
print(f"Student taking course : '{course_list[maxm_graduated_course - 1][1]}' are most graduated.")
print(f"Student taking course : '{course_list[maxm_enrolled_course - 1][1]}' are most enrolled.")

# Create a heatmap
plt.figure(figsize=(10, 2))
sns.heatmap(course_target_df.iloc[:-1, :].drop('All', axis = 1), annot=True, fmt='d', cbar=True)

xticks_labels = [value[1] for value in course_list]
xticks_positions = [pos + 0.5 for pos in range(len(xticks_labels))]


plt.xlabel('Course')
plt.ylabel('Target')
plt.xticks(xticks_positions, xticks_labels , rotation = 90)
# plt.title('Cross-Tabulation Heatmap')

# Show the plot
plt.show()

plt.figure(figsize = (8, 5), dpi = 90)

colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13), (0.24, 0.57, 0.90)]

student_course = df.groupby(['Course', 'Target']).size().reset_index().pivot(columns='Target', index='Course', values=0)

std_course = {count + 1: value[1] for count, value in enumerate(course_list)}
student_course = student_course.rename(index = std_course)

student_course_total = student_course.sum(axis=1)
student_course_sorted = student_course_total.sort_values(ascending=True)
student_course.loc[student_course_sorted.index].plot(kind='barh', stacked=True, color = colors)

plt.xlabel('Number of Students', font = font_label)
plt.ylabel('Course', font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)
plt.show()

temp_df = df[df['Target'] == 'Graduate']
pd.crosstab(index = temp_df['Gender'], columns = temp_df['Course'])

plt.figure(figsize = (8, 5), dpi = 90)

colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13), (0.24, 0.57, 0.90)]
student_nationality = df.groupby(['Nacionality', 'Target']).size().reset_index().pivot(columns='Target', index='Nacionality', values=0)

# Rename the index of the DataFrame
std_nationality = {count + 1 : value[1] for count, value in enumerate(nationality_list)}
student_nationality = student_nationality.rename(index = std_nationality)
```

```
student_nationality_total = student_nationality.sum(axis=1)
student_nationality_sorted = student_nationality_total.sort_values(ascending=True)
student_nationality.loc[student_nationality_sorted.index].plot(kind = 'barh', stacked = True, color = colors)

plt.xlabel('Number of Students', font = font_label)
plt.ylabel('Nationality', font = font_label)

plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)
plt.show()

sns.countplot(data = df, x='Displaced', hue='Target', hue_order=['Dropout', 'Enrolled', 'Graduate'])

plt.xticks(ticks=[0,1], labels=['No','Yes'])
plt.ylabel('Number of Students')
plt.show()

colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13),  (0.24, 0.57, 0.90)]
sns.countplot(data = df, x='International', hue='Target', hue_order=['Dropout', 'Enrolled', 'Graduate'])

plt.xticks(ticks=[0,1], labels=['No','Yes'])
plt.ylabel('Number of Students')
plt.show()

plt.figure(figsize = (8, 5), dpi = 90)

colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13), (0.24, 0.57, 0.90)]

student_prequal = df.groupby(['Previous qualification', 'Target']).size().reset_index().pivot(columns='Target', index='Previous qualification', values=0)

pre_qua = {count + 1 : value[1] for count, value in enumerate(previous_qualification_list)}

student_prequal = student_prequal.rename(index = pre_qua)

student_prequal_total = student_prequal.sum(axis=1)
student_prequal_sorted = student_prequal_total.sort_values(ascending=True)
student_prequal_top = student_prequal_sorted[8:]
student_prequal.loc[student_prequal_top.index].plot(kind='barh', stacked=True, color = colors)

plt.xlabel('Number of Students', font = font_label)
plt.ylabel('Previous Qualification', font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)
plt.show()

legend_title = FontProperties(family='Times New Roman', size = 12, weight = 'bold')

plt.figure(figsize = (8, 6), dpi = 90)

sns.histplot(data=df, x='Age at enrollment', kde=True)
stats = df['Age at enrollment'].describe()
stats_str = '\n'.join([f"{key.capitalize()}: {value:.2f}" for key, value in stats.items()])

legend_1 = plt.legend([plt.Line2D([], [], linestyle='none', marker=None, color='none')], [stats_str], handlelength=0, handletextpad=0, fancybox=True, prop = font, title = 'Descriptive Statistics', title_fontproperties = legend_title, loc = 'upper right')

plt.xlabel('Age at Enrollment', fontproperties=font_label)
plt.ylabel('Number of Students', fontproperties=font_label)
plt.xticks(fontproperties=font_ticks)
plt.yticks(fontproperties=font_ticks)
plt.show()


plt.figure(figsize = (15, 6), dpi = 90)
```

```

sns.set_style('whitegrid')
student_foccupation = df.groupby(["Father's occupation", 'Target']).size().reset_index().pivot(columns='Target', index="Father's occupation", values=0)

foccupation = {count + 1: value for count, value in enumerate(df_father['Occupation Name'].to_list())}
student_foccupation = student_foccupation.rename(index = foccupation)
student_foccupation_total = student_foccupation.sum(axis = 1)
student_foccupation_sorted = student_foccupation_total.sort_values(ascending = True)
student_foccupation_top10 = student_foccupation_sorted[36:]
colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13),  (0.24, 0.57, 0.90)]
student_foccupation.loc[student_foccupation_top10.index].plot(kind='barh', stacked=True, color = colors)

plt.xlabel('Number of Students', font = font_label)
plt.ylabel("Father's Occupation", font = font_label)

plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)

plt.show()

plt.figure(figsize = (15, 6), dpi = 90)

sns.set_style('whitegrid')

student_moccupation = df.groupby(["Mother's occupation", 'Target']).size().reset_index().pivot(columns='Target', index="Mother's occupation", values=0)

moccupation = {count + 1: value for count, value in enumerate(df_mother['Occupation Name'].to_list())}
student_moccupation = student_moccupation.rename(index = moccupation)
student_moccupation_total = student_moccupation.sum(axis = 1)
student_moccupation_sorted = student_moccupation_total.sort_values(ascending = True)
student_moccupation_top10 = student_moccupation_sorted[22:]
colors = [(0.69, 0.05, 0.18), (0.98, 0.47, 0.13),  (0.24, 0.57, 0.90)]
student_moccupation.loc[student_moccupation_top10.index].plot(kind='barh', stacked=True, color = colors)

plt.xlabel('Number of Students', font = font_label)
plt.ylabel("Mother's Occupation", font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.legend(prop = font_ticks)
plt.show()

bins = [17, 20, 25, 30, 35, 40, 45, 50, float('inf')]
labels = ['17-19', '20-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50+']

df['age_group'] = pd.cut(df['Age at enrollment'], bins=bins, labels=labels)

fig, axes = plt.subplots(2, 4, figsize=(14, 6))
# fig.suptitle('Distribution of Targets in Different Age Groups', fontsize=15)


target_colors = {'Graduate':(0.24, 0.57, 0.90), 'Dropout': (0.69, 0.05, 0.18), 'Enrolled': (0.98, 0.47, 0.13)}

for idx, age_group in enumerate(labels):
    ax = axes[idx // 4, idx % 4]

    data_subset = df[df['age_group'] == age_group]
    target_counts = data_subset['Target'].value_counts()

    target_colors_list = [target_colors[target] for target in target_counts.index]

```

```
    wedges, texts, autotexts = ax.pie(
        target_counts, labels =
        target_counts.index, autopct='
        \%1.1f\%\%', startangle=180,
        colors=target_colors_list,
        textprops={'color': 'white', '
        font': 'serif', 'size': 12})
    ax.set_title(f'Age Group: {age_group
        }', font = font_label, weight='
        bold')

legend_labels = [f'{label}: {color}' for
     label, color in zip(['Graduate'],
    colors)]

legend_labels = ['Dropout', 'Graduate',
    'Enrolled']
fig.legend(wedges, legend_labels, loc='
    center', bbox_to_anchor=(0.92, 0.85),
     title='Target', frameon = True,
    title_fontproperties = legend_title,
    prop = font_xticks)

plt.subplots_adjust(wspace = 0.0, hspace
    =0.15)
plt.savefig("./Figures/agegroup.png",
    dpi = 300, bbox_inches = 'tight')
plt.show()

sem_1st_grade = df[df['Curricular units
    1st sem (grade)'] > 0]
sns.histplot(data = sem_1st_grade, x = '
    Curricular units 1st sem (grade)',
    hue = 'Target')
plt.show()

sem_2nd_grade = df[df['Curricular units
    2nd sem (grade)'] > 0]
sns.histplot(data = sem_2nd_grade, x = '
    Curricular units 2nd sem (grade)',
    hue = 'Target')
plt.show()

colors = [(0.69, 0.05, 0.18),   (0.24,
    0.57, 0.90), (0.98, 0.47, 0.13)]

plt.figure(figsize = (8, 6))
sns.boxplot(x = 'Target', y = '
    Curricular units 1st sem (grade)',
    data = df, palette = colors)

plt.xlabel("Target",font = font_label)
plt.ylabel("Curricular units 1st sem (
    grade)", font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.show()

colors = [(0.69, 0.05, 0.18),   (0.24,
    0.57, 0.90), (0.98, 0.47, 0.13),]
plt.figure(figsize = (8, 6))
sns.boxplot(x = 'Target', y = '
    Curricular units 2nd sem (grade)',
    data = df, palette = colors)
plt.xlabel("Target",font = font_label)
plt.ylabel("Curricular units 2nd sem (
    grade)", font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.show()
```

4) Feature Selection

```
df_copy = df.copy()
df_copy['Target'] = df_copy['Target'].
    map({
     'Dropout': 0,
     'Enrolled': 1,
     'Graduate': 2
})
fix,ax=plt.subplots(1, 1, figsize=(16,
    10))
sns.heatmap(df_copy.drop('age_group',
    axis = 1).corr(), annot = False, ax =
     ax, fmt = '0.1', square = True)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.show()

X_train,X_test,y_train,y_test=
    train_test_split(df_copy.drop(labels
    = ['Target', 'age_group'], axis=1),
    df_copy['Target'], test_size = 0.2,
    random_state=0)

mf = mutual_info_classif(X_train,
    y_train)
mf = pd.Series(mf)
mf.index = X_train.columns
mf.sort_values(ascending=False).plot(
    kind = 'bar', width = 0.85)
plt.xlabel('Features', font = font_label
    )
plt.ylabel('Mutual Information Scores',
    font = font_label)
plt.xticks(font = font_ticks)
plt.yticks(font = font_ticks)
plt.show()

top_col=SelectKBest(mutual_info_classif,
     k = 5)
top_col.fit(X_train,y_train)
```

```
X_train.columns[top_col.get_support()]

numeric_columns = X_train.select_dtypes(
    include='number')
non_negative_columns = numeric_columns.
    columns[numeric_columns.min() >= 0]
print("Negative Columns", set(df_copy.
    columns.to_list()).difference(set(
    non_negative_columns.to_list())))

X_train_chi2 = X_train[
    non_negative_columns.to_list()]
f_score_p = chi2(X_train_chi2, y_train)
p_values = pd.Series(f_score_p[1])
p_values.index = X_train_chi2.columns
p_values.sort_values()
top_col=SelectKBest(chi2, k = 5)
top_col.fit(X_train_chi2, y_train)
X_train_chi2.columns[top_col.get_support
    ()]
```

5) Data Modeling

```
df_data_modeling = df.copy()
df_data_modeling['Target'] =
    df_data_modeling['Target'].map({
     'Dropout': 0,
     'Enrolled': 1,
     'Graduate': 2
})
df_data_modeling.drop(df_data_modeling[
    df_data_modeling['Target'] == 1].
    index, inplace = True)
df_data_modeling.tail()
df_data_modeling['Dropout'] =
    df_data_modeling['Target'].apply(
    lambda x: 1 if x == 0 else 0)
df_data_modeling.head()

plt.figure(figsize=(5, 5))
colors = [(0.24, 0.57, 0.90), (0.69,
    0.05, 0.18)]
plt.pie(df_data_modeling['Dropout'].
    value_counts(), labels = ['Non-
    Dropout', 'Dropout'], explode =
    (0.05, 0.0), autopct='\%1.2f\%\%',
    shadow = True, textprops = {'family':
    'Times New Roman', 'size': 12, '
    weight': 'bold', 'color':'black'},
    colors = colors)
plt.legend(bbox_to_anchor=(0.75, 1.0),
    prop = font_ticks)
plt.show()

X = df_data_modeling.iloc[:, :34].values
y = df_data_modeling['Dropout'].values

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size =
    0.2, random_state = 5)

model_nb_4 = GaussianNB()
model_nb_4.fit(X_train, y_train)

y_pred_nb = model_nb_4.predict(X_test)
scores = cross_val_score(model_nb_4,
    X_train, y_train, scoring = 'accuracy
    ', cv = 44)
verbose_result(y_pred_nb, y_test)
print("With Scaling and With Cross-
    Validation: ", scores.mean())

X = df_data_modeling.iloc[:, :34].values
X = StandardScaler().fit_transform(X)
# X = MinMaxScaler().fit_transform(X)
y = df_data_modeling['Dropout'].values

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size =
    0.2, random_state = 5)

model_BNB_2 = BernoulliNB()
model_BNB_2.fit(X_train, y_train)

y_pred_nb = model_BNB_2.predict(X_test)
verbose_result(y_pred_nb, y_test)
scores = cross_val_score(model_BNB_2,
    X_train, y_train, cv = 13)
print("With Scaling and With Cross-
    Validation: ", scores.mean())

scores_list = []
for i in tqdm(range(2, 50), ncols = 100)
    :
     scores_list.append(cross_val_score(
        model_BNB_2, X_train, y_train,
        scoring = 'accuracy', cv = i).
        mean())

print(f"Maximum scores is {np.max(np.
    array(scores_list))} at {np.argmax(np
    .array(scores_list)) + 2} cv value")


categorical_df = df_data_modeling.copy()
categorical_df['Curricular units 1st sem
     (grade)'] = pd.cut(categorical_df['
    Curricular units 1st sem (grade)'],
    6)
categorical_df['Curricular units 2nd sem
     (grade)'] = pd.cut(categorical_df['
    Curricular units 2nd sem (grade)'],
```

```
    6)
categorical_df['Unemployment rate'] = pd
    .cut(categorical_df['Unemployment
    rate'], 6)

age_group = categorical_df['age_group']
categorical_df = categorical_df.drop('
    age_group', axis = 1)
categorical_df['Age at enrollment'] =
    age_group
categorical_df.drop(['Inflation rate', '
    GDP', 'Target'], axis = 1, inplace =
    True)

categorical_df['Age at enrollment'] =
    categorical_df['Age at enrollment'].
    fillna('17-19')

categorical_columns = ['Curricular units
     1st sem (grade)', 'Curricular units
    2nd sem (grade)', 'Unemployment rate'
    , 'Age at enrollment']
ord_encoder = OrdinalEncoder()
categorical_df[categorical_columns] =
    ord_encoder.fit_transform(
    categorical_df[categorical_columns])

X = categorical_df.iloc[:, :-1].values
# X = MinMaxScaler().fit_transform(X)
y = categorical_df['Dropout'].values

X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size =
    0.2, random_state = 5)

model_CNB = CategoricalNB()
model_CNB.fit(X_train, y_train)

y_pred_nb = model_CNB.predict(X_test
    [0:595])
verbose_result(y_pred_nb, y_test[:595])

continuous_columns = ['Course', "Mother'
    s occupation", "Father's occupation",
     'Age at enrollment', 'Curricular
    units 1st sem (enrolled)', '
    Curricular units 1st sem (evaluations
    )', 'Curricular units 1st sem (
    approved)', 'Curricular units 1st sem
     (grade)', 'Curricular units 2nd sem
    (enrolled)', 'Curricular units 2nd
    sem (evaluations)', 'Curricular units
     2nd sem (approved)', 'Curricular
    units 2nd sem (grade)']

font_xlabel = FontProperties(family='
    Times New Roman', size = 16, weight =
     'bold')
num_rows = (len(continuous_columns) + 2)
     // 3  # Limit to 4 columns per row
num_cols = min(len(continuous_columns),
    3)

fig, axes = plt.subplots(num_rows,
    num_cols, figsize = (18, 18))
for i, column in enumerate(
    continuous_columns):
    sns.histplot(data = df, x = column,
        kde = True, ax = axes[i //
        num_cols, i % num_cols])
    axes[i // num_cols, i % num_cols].
        set_xlabel(column, font =
        font_xlabel)

# Remove the empty subplots (axes)
if len(continuous_columns) < (num_rows *
     num_cols):
    for i in range(len(
        continuous_columns), num_rows *
        num_cols):
        plt.delaxes(axes.flatten()[i])

plt.subplots_adjust(wspace = 0.5, hspace
    = 0.9)
plt.tight_layout()
plt.savefig('./Figures/feature
    distribution.png', dpi = 300,
    bbox_inches = 'tight')
plt.show()

continuous_columns = ['Course', "Mother'
    s occupation", "Father's occupation",
     'Age at enrollment', 'Curricular
    units 1st sem (enrolled)', '
    Curricular units 1st sem (evaluations
    )', 'Curricular units 1st sem (
    approved)', 'Curricular units 1st sem
     (grade)', 'Curricular units 2nd sem
    (enrolled)', 'Curricular units 2nd
    sem (evaluations)', 'Curricular units
     2nd sem (approved)', 'Curricular
    units 2nd sem (grade)', 'Unemployment
     rate']

categorical_columns = ['Marital status',
     'Application mode', 'Application
    order', 'Daytime/evening attendance',
     'Previous qualification', '
    Nacionality', "Mother's qualification
    ", "Father's qualification", '
    Displaced', 'Educational special
```

```
    needs', 'Debtor', 'Tuition fees up to date', 'Gender', 'Scholarship holder', 'International', 'Curricular units 1st sem (credited)', 'Curricular units 1st sem (without evaluations)', 'Curricular units 2nd sem (credited)', 'Curricular units 2nd sem (without evaluations)']

# Turn off the SettingWithCopyWarning globally
pd.options.mode.chained_assignment = None

hybrid_data = df.copy()
hybrid_data['Target'] = hybrid_data['Target'].map({
    'Dropout': 0,
    'Enrolled': 1,
    'Graduate': 2
})
hybrid_data.drop(hybrid_data[hybrid_data['Target'] == 1].index, inplace = True)
hybrid_data['Dropout'] = hybrid_data['Target'].apply(lambda x : 1 if x == 0 else 0)
hybrid_data.drop([1618, 4420, 3578], axis = 0, inplace = True)
hybrid_data.tail()

X = hybrid_data.drop('Target', axis = 1)
y = hybrid_data['Dropout']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 5)

X_test_ = pd.concat([X_test.iloc[:594], X_test.iloc[596:]], axis = 0)
y_test_ = pd.concat([y_test.iloc[:594], y_test.iloc[596:]], axis = 0)


X_train_gaussian = X_train[continuous_columns]
X_test_gaussian = X_test_[continuous_columns]

X_train_categorical = X_train[categorical_columns]
X_test_categorical = X_test_[categorical_columns]


model_GNB = GaussianNB()
model_GNB.fit(X_train_gaussian, y_train)
y_pred_gaussian = model_GNB.predict(X_test_gaussian)
# verbose_result(y_pred_gaussian, y_test_)

print(build_classification_rep(y_pred_gaussian, y_test_))

model_CNB = CategoricalNB()
model_CNB.fit(X_train_categorical, y_train)
y_pred_categorical = model_CNB.predict(X_test_categorical)
# verbose_result(y_pred_categorical, y_test_)
print(build_classification_rep(y_pred_categorical, y_test_))

count = 0
y_pred_hybrid = []
for pred_cat, pred_gauss in zip(y_pred_categorical, y_pred_gaussian):
    if pred_cat == pred_gauss:
        y_pred_hybrid.append(pred_cat)
    else:
        y_pred_hybrid.append(1)
print("Accuracy of hybrid NB: ", (y_pred_hybrid == y_test_).mean())
```