Date of submission June , 2024

Lab 3

Naive Bayes Classification

ATUL SHREEWASTAV (THA077BCT013)[1], OM PRAKASH SHARMA (THA077BCT030)[2]
[1,2]Department of Electronics and Computer Engineering, Thapathali Campus, Institute of Engineering, Kathmandu 44600 NP)
[1]e-mail: atul.077bct013@tcioe.edu.np
[2]e-mail: om.077bct030@tcioe.edu.np

For the fulfillment of Lab Requirement, *CT72502 Data Mining*

ABSTRACT In this study, the efficacy of the Naïve Bayes classification algorithm was investigated using two distinct datasets: the Titanic dataset and another from the UCI Machine Learning Repository. The goal was to predict passenger survival on the Titanic and perform general data classification tasks. For the Titanic dataset, an extensive preprocessing phase was involved, including feature selection, data visualization, and conversion of categorical values to numerical values. Different visualizations like count plots, scatter plots, and histograms were employed to understand data distributions. Data was split into training and testing sets, and Naïve Bayes models—both Gaussian and Categorical—were trained and evaluated. The performance was assessed using confusion matrices and error rates. Identical procedures were applied to the second dataset, enabling a comparative analysis and solidifying the understanding and application of Naïve Bayes classifiers on both textual, numerical, and categorical data.

INDEX TERMS Naive Bayes Classifier, Multinomial Naive Bayes, Bernoulli Naive Bayes, Gaussian Naive Bayes, Bayes' Theorem

I. INTRODUCTION

NAive Bayes classifiers are a family of probabilistic classifiers based on Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the class label. Despite this simplification, naive Bayes classifiers have proven to be effective in a variety of complex real-world applications due to their simplicity, scalability, and robust performance.

These classifiers are designed to predict class labels for given problem instances, which are represented as vectors of feature values. The primary assumption is that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. For instance, when classifying fruits, a naive Bayes classifier might consider attributes like color, shape, and size independently to determine if a fruit is an apple.

Naive Bayes models are highly scalable and require a number of parameters that is linear with respect to the number of features in a given learning problem. This scalability is enhanced by the fact that parameter estimation for these models can be achieved through maximum likelihood estimation, which is computationally efficient and avoids the need for iterative approximations.

One of the significant advantages of naive Bayes classifiers is their efficiency in terms of training data. They require relatively small amounts of data to accurately estimate the necessary parameters for classification. This efficiency makes them particularly useful in situations where acquiring large datasets is impractical.

Despite their apparent oversimplification, naive Bayes classifiers have demonstrated strong performance in various domains, such as text classification, spam filtering, and medical diagnosis. The theoretical underpinning of their effectiveness was further reinforced by a 2004 analysis, which provided sound reasons for their robust performance. However, a comprehensive comparison in 2006 indicated that other classification techniques, like boosted trees or random forests, can outperform naive Bayes in certain scenarios. Naive Bayes classifiers offer a straightforward yet powerful approach to classification tasks. Their ease of implementation, combined with their ability to deliver reliable results even with limited data, makes them a valuable tool in the arsenal of machine learning techniques.

Naive Bayes classifiers are highly effective in various practical applications due to their simplicity and efficiency. One of the most common applications is spam filtering. By analyzing the frequency and presence of specific words or phrases in emails, naive Bayes classifiers can effectively distinguish between spam and legitimate messages. This is achieved by training the model on a labeled dataset where emails are already marked as spam or not. The classifier then calculates the probability of an email being spam based on

the occurrence of these words, making it a powerful tool for email providers to filter unwanted messages.

In sentiment analysis, naive Bayes classifiers are used to detect the sentiment expressed in a piece of text, such as reviews, tweets, or comments. The classifier is trained on a dataset where the sentiment is already labeled as positive, negative, or neutral. By analyzing the presence and frequency of words associated with different sentiments, the model can predict the sentiment of new, unseen texts. This application is widely used in customer feedback analysis, social media monitoring, and market research to understand public opinion and sentiments towards products, services, or topics.

Document classification is another significant application of naive Bayes classifiers. In this context, the classifier categorizes documents into predefined classes based on their content. For instance, in a news aggregation service, articles can be classified into categories such as politics, sports, entertainment, and technology. The classifier is trained on a labeled dataset of documents and learns to associate certain words or phrases with specific categories. When a new document is presented, the classifier calculates the probabilities of the document belonging to each category and assigns it to the one with the highest probability.

In the field of medical diagnosis, naive Bayes classifiers assist in diagnosing diseases by evaluating the probabilities of various conditions given the symptoms. The classifier is trained on medical records where the presence of symptoms and the corresponding diagnoses are recorded. By analyzing this data, the model learns the likelihood of different diseases associated with specific symptoms. When a new patient presents with certain symptoms, the classifier calculates the probability of each potential disease and helps doctors make informed decisions about the diagnosis and treatment.

II. RELATED WORKS

The Naive Bayes classifier is a probabilistic model that leverages Bayes' theorem to compute the conditional probability of a class given a set of features. The model is termed "naive" because it assumes that all features are mutually independent given the class label, which significantly simplifies the computation, though this assumption rarely holds true in real-world scenarios. The classifier begins by estimating two sets of probabilities from the training data: the prior probabilities of each class, which reflect the relative frequency of each class in the training dataset, and the likelihood of each feature given each class, which is determined based on how often each feature occurs within each class. Once these probabilities are established, Bayes' theorem is employed to invert the conditional probability, making it feasible to compute the posterior probability of each class given new observations. The model classifies new data points by selecting the class with the highest posterior probability.

There are several variants of Naive Bayes classifiers designed for different types of data. For discrete data often found in text classification problems, the Multinomial Naive Bayes model is commonly used. It handles feature vectors representing the frequency of words in documents, operating under the assumption that word frequencies follow a multinomial distribution. For binary or Boolean features, like the presence or absence of words, the Bernoulli Naive Bayes model is better suited and explicitly models the absence of features, which is particularly valuable for short text classification. When dealing with continuous data, Gaussian Naive Bayes is frequently employed, assuming that the continuous features follow a normal (Gaussian) distribution within each class. The mean and variance of each feature are computed for each class, and the probability of a new data point is computed using these Gaussian distributions.

In practice, the Naive Bayes classifier must address the issue of zero probabilities, which occur when a particular feature value is not present in the training data for a given class. To prevent zero probabilities from nullifying the posterior probability computation, smoothing techniques like Laplace smoothing are applied, adding a small pseudo-count to each possible feature value. This ensures that no probability is ever exactly zero, making the model more robust.

Further extending its capabilities, Naive Bayes can be used in semi-supervised learning scenarios where both labeled and unlabeled data are available. Using the Expectation-Maximization (EM) algorithm, the classifier is first trained on the labeled data. In the E-step, it predicts the class probabilities for the unlabeled data, and in the M-step, it re-trains itself using both the labeled data and the probabilities from the E-step. This iterative process continues until the model converges.

Gaussian Naive Bayes is used for continuous data, assuming that the continuous values associated with each class are distributed according to a Gaussian (normal) distribution. When the assumption holds, this classifier performs quite well. The Gaussian Naive Bayes model calculates the likelihood of the features based on the Gaussian distribution. This is particularly useful in applications where the feature distribution closely resembles a normal distribution, such as in medical data analysis and sensor data classification [1].

Multinomial Naive Bayes is typically used for discrete data, such as word counts or term frequencies in text classification problems. This classifier assumes that the features follow a multinomial distribution, which makes it well-suited for document classification tasks like spam detection and sentiment analysis. The Multinomial Naive Bayes model calculates the probability of a document belonging to a particular class based on the frequency of words within the document. This method has been widely applied in natural language processing tasks and has demonstrated significant effectiveness in handling large vocabularies [2].

Bernoulli Naive Bayes is similar to the Multinomial Naive Bayes but is designed for binary/boolean features. It assumes that all features are binary-valued (0s and 1s). In the context of text classification, this would mean considering whether a term occurs or not in a document rather than the term frequency. This classifier is particularly useful for tasks where the absence or presence of a feature is more important than

its frequency, such as email classification into spam and non-spam categories [3].

Multinomial Naive Bayes is typically used for discrete data, such as word counts or term frequencies in text classification problems. This classifier assumes that the features follow a multinomial distribution, which makes it well-suited for document classification tasks like spam detection and sentiment analysis. The Multinomial Naive Bayes model calculates the probability of a document belonging to a particular class based on the frequency of words within the document. This method has been widely applied in natural language processing tasks and has demonstrated significant effectiveness in handling large vocabularies [2].

Complement Naive Bayes (CNB) is an adaptation of the standard Multinomial Naive Bayes classifier. It is particularly effective for imbalanced datasets where certain classes dominate. Unlike traditional Multinomial Naive Bayes, which estimates class-specific probabilities for each feature independently, CNB considers the complement of each class's frequencies relative to the whole dataset. This approach makes CNB suitable for text classification tasks where class imbalance is prevalent, such as sentiment analysis and news categorization [4].

Categorical Naive Bayes is similar to Multinomial Naive Bayes but is specifically designed for categorical features that can take on a finite number of values. It assumes that the categorical features follow a categorical distribution, where each feature value is treated as an independent category. This classifier is useful in scenarios where features are represented as categorical variables, such as in customer segmentation based on demographic data or recommendation systems based on user preferences [5].

Despite the simplifications inherent in its assumptions, the Naive Bayes classifier remains effective in a wide range of applications due to its simplicity and speed. It's particularly well-suited for tasks like spam detection, document classification, sentiment analysis, and medical diagnosis. The ease of implementation, combined with its surprisingly good performance even in the face of strong feature dependencies, makes it a valuable tool in the field of machine learning and data science.

III. METHODOLOGY

A. DATASET DESCRIPTION

The dataset used in this lab report originates from the UCI Machine Learning Repository and is titled "Predict Students Dropout and Academic Success." This dataset comprises various academic and personal attributes of students, aiming to predict dropout rates and academic success through classification models, particularly the Naive Bayes classifier in this context.The dataset contains 33 attributes divided into three main categories: demographic data, academic background, and social and economic context. These attributes provide comprehensive information about the students, which are essential for predictive modeling.

1) Demographic Data

- **Maritalstatus:** Marital status of the student
- **Nationality:** Student's nationality
- **Gender:** Gender of the student
- **Ageatenrollment:** Age of the student at enrollment
- **International:** Whether the student is an international student

2) Academic Background

- **Applicationmode:** Mode of application
- **Applicationorder:** Order of application preference
- Course: The course in which the student is enrolled
- **Daytimeeveningattendance:** Indicates if the student attends classes during the day or evening
- **Previousqualification:** The previous educational qualification of the student
- **Previousqualificationgrade:** Grade of the previous qualification (0 to 200)
- **Admissiongrade:** Grade obtained during admission
- **Curricularunitsstsemcredited:** Number of curricular units credited in the first semester
- **Curricularunitsstsemenrolled:** Number of curricular units enrolled in the first semester
- **Curricularunitsstsemevaluations:** Number of evaluations in the first semester
- **Curricularunitsstsemapproved:** Number of approved curricular units in the first semester
- **Curricularunitsstsemgrade:** Grade of curricular units in the first semester
- **Curricularunitsstsemwithoutevaluations:** Number of curricular units without evaluations in the first semester
- **Curricularunitsndsemcredited:** Number of curricular units credited in the second semester
- **Curricularunitsndsemenrolled:** Number of curricular units enrolled in the second semester
- **Curricularunitsndsemevaluations:** Number of evaluations in the second semester
- **Curricularunitsndsemapproved:** Number of approved curricular units in the second semester
- **Curricularunitsndsemgrade:** Grade of curricular units in the second semester
- **Curricularunitsndsemwithoutevaluations:** Number of curricular units without evaluations in the second semester
- **Unemploymentrate:** Unemployment rate at the time of enrollment
- **Inflationrate:** Inflation rate at the time of enrollment
- **GDP:** Gross Domestic Product at the time of enrollment

3) Social and Economic Context

- **Mothersqualification:** Mother's education level
- **Fathersqualification:** Father's education level
- **Mothersoccupation:** Mother's occupation
- **Fathersoccupation:** Father's occupation
- **Displaced:** Whether the student is displaced (Yes, No)

- **Educationalspecialneeds:** Indicates if the student has special educational needs (Yes, No)
- **Debtor:** Whether the student is a debtor (Yes, No)
- **Tuitionfeesuptodate:** Indicates if tuition fees are up to date (Yes, No)
- **Scholarshipholder:** Indicates if the student holds a scholarship (Yes, No)

4) Target Variable

- **Target:** Classification of the student's status (Dropout=0, Enrolled=1, Graduate=2)

The dataset is designed to develop machine learning models that can predict student dropout rates and academic success. By analyzing these features, educational institutions can identify at-risk students and implement supportive measures to improve retention and success rates.

B. PROPOSED METHODOLOGY

The first phase is data preparation, which will begin with data cleaning. This step involves addressing missing values by either removing or imputing them, and identifying and treating anomalies or outliers appropriately. Data types will be checked and corrected to ensure compatibility with subsequent processing steps. Ensuring the data is clean and well-prepared at this stage will lay a strong foundation for building a reliable model.

Next, categorical features will be transformed into numerical format using 'OrdinalEncoder'. This transformation is essential because Naive Bayes requires numerical input. 'OrdinalEncoder' will convert categorical variables into numerical values while preserving the ordinal relationship between categories. This step is crucial for allowing the Naive Bayes classifier to process the data effectively.

Feature scaling will then be considered, particularly when working with algorithms that assume normally distributed data. Although Naive Bayes is not highly sensitive to feature scaling, it can still benefit from standardization to ensure consistency across features. Standardization involves transforming features to have a mean of zero and a standard deviation of one, which helps in aligning the scales of different features and can improve the model's performance.

The dataset will be split into training and test sets using the *train_test_split* function. This step is crucial for evaluating the model's performance on unseen data, ensuring that the model does not overfit to the training data. Typically, 70-80% of the data will be used for training the model, and the remaining 20-30% will be set aside for testing. This split will allow for a fair assessment of the model's generalization capabilities.

For model training, the appropriate variant of the Naive Bayes algorithm will be selected based on the nature of the data. Gaussian Naive Bayes will be used for continuous data, Multinomial Naive Bayes for count data or multinomially distributed data, and Bernoulli Naive Bayes for binary/boolean features. The choice of the variant depends on the specific characteristics of the features in the dataset. Once the variant is selected, the model will be trained by fitting it to the training data. This involves estimating the parameters of the probability distributions of the features given the class labels.

After training the model, its performance will be evaluated using the test set. Predictions will be made on the test data, and various performance metrics will be computed to assess the model's effectiveness. Key metrics will include accuracy, which measures the overall correctness of the model; precision, which evaluates the proportion of positive identifications that are actually correct; recall, which assesses the ability of the model to find all relevant instances; and the F1 score, which balances precision and recall. The confusion matrix will provide a detailed breakdown of the model's performance across different classes, highlighting areas where the model performs well and where it may need improvement.

To further refine the model, hyperparameter tuning will be conducted. Although Naive Bayes has relatively few hyperparameters, exploring different priors or smoothing parameters can lead to performance improvements. Additionally, cross-validation will be an essential step to ensure the model's robustness and generalizability. Cross-validation involves splitting the data into several folds and training the model multiple times on different training and validation sets. This process will help to mitigate the risk of overfitting and provide a more reliable estimate of the model's performance.

Visualization will play a crucial role in model evaluation and interpretation. By plotting the confusion matrix as a heatmap, ROC curves, and precision-recall curves, deeper insights into the model's strengths and weaknesses will be gained. These visualizations will help in understanding how well the model differentiates between classes and where it might be making errors.

Finally, the methodology will conclude with a comprehensive summary of the findings. This includes highlighting the model's performance metrics, discussing any limitations encountered during the analysis, and proposing potential areas for improvement. Future steps may involve experimenting with different models, incorporating additional features, or applying more advanced techniques to further enhance the model's performance. This detailed and systematic approach will ensure a robust implementation and thorough evaluation of the Naive Bayes classifier, providing valuable insights and practical recommendations for further work.

C. IMPLEMENTATION

The implementation was conducted using Python along with several widely-used machine learning libraries. Key among these were pandas, which facilitated efficient data manipulation, `scikit-learn` for constructing and assessing the Naive Bayes classifier, and `matplotlib` for visualizing the resultant model performance. The Naive Bayes classifier from `scikit-learn` was specifically employed to predict categorical outcomes based on input features, leveraging its simplicity and effectiveness in tasks requiring probabilistic inference.

The use of `train_test_split` ensured the dataset was appropriately partitioned for training and evaluation, while `metrics.accuracy_score` provided a clear measure of the classifier's predictive accuracy. This approach exemplifies a robust methodology for applying Naive Bayes classifiers in practical machine learning scenarios, emphasizing both model performance evaluation and interpretation of results.

D. MATHEMATICAL FORMULAE

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable. Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)} \tag{1}$$

Using the naive conditional independence assumption that

$$P(x_i \mid y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i \mid y), \tag{2}$$

for all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{3}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{4}$$

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y) \tag{5}$$

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

Gaussian Naive Bayes assumes the likelihood of the features is Gaussian:

$$P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right) \tag{6}$$

where σ_y and μ_y are estimated using maximum likelihood.

Multinomial Naive Bayes is used for multinomially distributed data, such as text classification:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_y + \alpha n} \tag{7}$$

where N_{yi} is the number of times feature i appears in a sample of class y, N_y is the total count of all features for class y, and α is a smoothing parameter.

Complement Naive Bayes (CNB) is adapted for imbalanced datasets:

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} d_{ij}}{\alpha + \sum_{j:y_j \neq c} \sum_k d_{kj}} \tag{8}$$

where α_i is a smoothing hyperparameter and d_{ij} is the count or tf-idf value of term i in document j.

Bernoulli Naive Bayes assumes binary-valued features:

$$P(x_i \mid y) = P(x_i = 1 \mid y)x_i + (1 - P(x_i = 1 \mid y))(1 - x_i) \tag{9}$$

Categorical Naive Bayes is used for categorically distributed data:

$$P(x_i = t \mid y = c\,;\, \alpha) = \frac{N_{tic} + \alpha}{N_c + \alpha n_i} \tag{10}$$

where N_{tic} is the number of times category t appears in samples x_i belonging to class c, and N_c is the number of samples with class c.

E. INSTRUMENTATION DETAILS

This section outlines the technical and methodological specifics involved in implementing Naive Bayes classification.

1) Software and Libraries Used

- **Python**: Chosen for its versatility and extensive ecosystem of libraries suitable for data analysis and machine learning tasks.
- **UciMlRepo**: Used to load the dataset as a dataframe in the program directly.
- **Scikit-learn**: Provides efficient implementations of various machine learning algorithms, including Naive Bayes classifiers, and offers tools for model selection, evaluation, and preprocessing.
- **Pandas**: Utilized for data manipulation and analysis, offering powerful data structures like DataFrames that simplify the process of cleaning and preparing data. It is essential for preprocessing tasks such as handling missing values and encoding categorical variables.
- **Seaborn**: Employed for statistical data visualization, making it easier to create informative and attractive visual representations of the data and PCA results.
- **Matplotlib**: Used alongside Seaborn to customize and enhance plots, providing fine-grained control over the visual elements of the graphs.

2) Data Preparation

- **Dataset Acquisition:** The datasets used include the "Simple Weather Forecast" dataset and the "Adult Census Income" dataset, both suitable for demonstrating different aspects of Naive Bayes classifiers.
- **Data Loading:** Both datasets were loaded into Pandas DataFrames for easy manipulation and analysis. The "Simple Weather Forecast" dataset was manually typed, while the "Adult Census Income" dataset was obtained from the UCI Machine Learning Repository.

3) Data Preprocessing

- **Handling Missing Values:** The "Adult Census Income" dataset contained missing values, represented by '?'. These missing values were either removed or imputed to ensure the dataset's integrity. The "Simple Weather Forecast" dataset did not contain any missing values.
- **Encoding Categorical Values:** Both datasets contained categorical variables that needed to be converted into numerical format for use in Naive Bayes algorithms. One-hot encoding was employed to transform categorical attributes into a binary matrix, ensuring that the Naive Bayes algorithm could process the data effectively.
- **Splitting Data:** Each dataset was split into training and testing sets, with a 67-33 split ratio and random_state set to 10 to facilitate model training and evaluation.

4) Naive Bayes Classification

- **Model Implementation:** Naive Bayes classifiers were implemented using `scikit-learn`. Specifically:
  - **Gaussian Naive Bayes (GaussianNB):** Implemented assuming the likelihood of features being Gaussian.
  - **Categorical Naive Bayes (CategoricalNB):** Implemented assuming categorical distribution of features.
  - **Hybrid Naive Bayes Model:** Custom implementation combining Gaussian and Categorical Naive Bayes classifiers for improved performance.
- **Hyperparameter Tuning:** Hyperparameters such as smoothing parameters for Gaussian and Categorical Naive Bayes were tuned to optimize model performance.
- **Performance Metrics:** Model performance was evaluated using metrics such as accuracy, precision, recall, and F1-score, calculated using scikit-learn's *classification_report()* function.

5) Visualization

- **Model Visualization:** The structure and decision-making process of trained Naive Bayes models were visualized using appropriate methods provided by `scikit-learn`.
- **Feature Importance:** The importance of each feature in making predictions was visualized using bar plots, aiding in understanding the significance of different features in the classification process.

6) Development Environment

- **Jupyter Notebook**: The Jupyter Notebook environment was used for developing and running Python scripts, facilitating an interactive and iterative approach to data analysis and model development.

F. SYSTEM BLOCK DIAGRAM

The block diagram as shown in 15 Naive Bayes Classification begins with raw data, which includes both categorical and numerical features, serving as the input for the subsequent steps in the system. The initial step involves data cleaning, which handles missing values, corrects errors, and removes or imputes outliers, ensuring that the dataset is of high quality and suitable for analysis. Following this, categorical features are transformed into numerical values using Ordinal Encoding, essential because machine learning algorithms, including Naive Bayes, require numerical input, and this encoding preserves the ordinal relationship between categories.

The cleaned and encoded data is then split into training and test sets, 75% used for training the model and the remaining 25% for testing and evaluating the model's performance. Numerical data undergoes further preprocessing, including feature scaling. Standardization using StandardScaler transforms the features to have a mean of zero and a standard deviation of one, ensuring that all features contribute equally to the model.

Depending on the nature of the data, different variants of the Naive Bayes model are applied: GaussianNB for continuous data that follows a Gaussian (normal) distribution, CategoricalNB for categorical data, and a HybridNBModel for handling datasets with both continuous and categorical features effectively. The selected Naive Bayes model is then trained on the training set, where the model learns the parameters of the probability distributions of the features given the class labels. Once trained, the model is used to make predictions on the test set.

The final phase involves visualizing the results of the model. Visualization techniques such as confusion matrices, ROC curves, and precision-recall curves help interpret the model's performance, providing insights into how well the model is performing, highlighting its strengths, and indicating areas for improvement. This structured approach ensures a robust and reliable implementation of Naive Bayes classification, covering all critical phases from data preparation to result interpretation and visualization.

IV. EXPERIMENTAL RESULTS

The evaluation of Naive Bayes classifiers on the dataset yielded varied performance across three distinct variants: Gaussian Naive Bayes, Categorical Naive Bayes, and Hybrid Naive Bayes. Each variant was assessed based on precision, recall, F1-score, and overall accuracy across three classes: Dropout, Enrolled, and Graduate.

Starting with **Gaussian Naive Bayes** as shown in Figure: 1, the classifier achieved an overall accuracy of 0.70. It demonstrated the highest precision for Dropout (0.83) and Graduate (0.77) classes, indicating strong performance in correctly identifying these categories. However, its recall for Enrolled students was lower at 0.53, suggesting some difficulty in accurately predicting this class. The F1-scores were generally robust, particularly for Graduate students (0.80), reflecting a balanced performance in both precision and recall metrics.

Categorical Naive Bayes as shown in Figure: 2 exhibited a lower overall accuracy of 0.62 compared to Gaussian Naive Bayes. This variant struggled notably with the Enrolled class,

where it achieved the lowest precision (0.45) and recall (0.05) among the three models. While it showed competitive performance in some areas, such as the high recall for Graduate students (0.87), its overall effectiveness was hindered by disparities in handling the Enrolled category.

Hybrid Naive Bayes as shown in Figure: 3 combined elements of Gaussian and categorical models to achieve an accuracy of 0.72. This approach yielded balanced results across precision and recall metrics, particularly enhancing recall for Enrolled students to 0.19 compared to Categorical Naive Bayes. It demonstrated strong precision for Dropout (0.81) and Graduate (0.71) classes, with an overall F1-score improvement, indicating effective integration of both continuous and categorical data aspects.

The confusion matrix for the Hybrid Naive Bayes Model provides a detailed breakdown of the model's performance in classifying instances across three different classes (0, 1, and 2). The matrix is a 3x3 grid where each cell (i, j) represents the number of instances from actual class i that were predicted as class j. The diagonal elements (243, 34, and 467) indicate the instances correctly classified by the model for classes 0, 1, and 2 respectively. For class 0, the model correctly predicted 243 instances but misclassified 24 instances as class 1 and 80 instances as class 2. For class 1, it correctly classified 34 instances but incorrectly classified 30 instances as class 0 and 115 instances as class 2. For class 2, the model showed strong performance with 467 correct predictions, but it misclassified 27 instances as class 0 and 8 instances as class 1. This matrix allows for a visual assessment of the model's accuracy, highlighting where the model performs well and where it struggles, thus guiding further refinement and tuning efforts to improve classification performance.

The confusion matrix for the Gaussian Naive Bayes Model provides a detailed breakdown of the model's performance in classifying instances across three different classes (0, 1, and 2). The matrix is a 3x3 grid where each cell (i, j) represents the number of instances from actual class i that were predicted as class j. The diagonal elements (164, 9, and 357) indicate the instances correctly classified by the model for classes 0, 1, and 2 respectively. For class 0, the model correctly predicted 164 instances but misclassified 38 instances as class 1 and 51 instances as class 2. For class 1, it correctly classified 9 instances but incorrectly classified 1 instances as class 0 and 8 instances as class 2. For class 2, the model showed strong performance with 357 correct predictions, but it misclassified 125 instances as class 0 and 103 instances as class 1. This matrix allows for a visual assessment of the model's accuracy, highlighting where the model performs well and where it struggles, thus guiding further refinement and tuning efforts to improve classification performance.

The confusion matrix for the Hybrid Naive Bayes Model provides a detailed breakdown of the model's performance in classifying instances across three different classes (0, 1, and 2). The matrix is a 3x3 grid where each cell (i, j) represents the number of instances from actual class i that were predicted as class j. The diagonal elements (243, 34, and 467) indicate the instances correctly classified by the model for classes 0, 1, and 2 respectively. For class 0, the model correctly predicted 243 instances but misclassified 30 instances as class 1 and 27 instances as class 2. For class 1, it correctly classified 34 instances but incorrectly classified 24 instances as class 0 and 8 instances as class 2. For class 2, the model showed strong performance with 467 correct predictions, but it misclassified 80 instances as class 0 and 115 instances as class 1. This matrix allows for a visual assessment of the model's accuracy, highlighting where the model performs well and where it struggles, thus guiding further refinement and tuning efforts to improve classification performance.

The Histogram and Q-Q Plot are shown in figures: 4, 5, 6, 7, 8, 9, 10, 11 and 12. For the feature "HouseAge," the histogram indicates a left-skewed distribution with most values concentrated between 20 and 60. The Q-Q plot shows deviation from normality, especially in the tails, suggesting non-normal distribution. The "AverageRooms" feature displays a more normal distribution, with the histogram depicting a bell-shaped curve centered around 5. The Q-Q plot confirms this, closely aligning with the diagonal line, indicating normality.

The "AverageBedrooms" feature is heavily right-skewed, with a significant number of observations clustered around 1. The Q-Q plot for this feature shows substantial deviation from the normal distribution, particularly in the upper quantiles. The "Population" feature's histogram reveals a right-skewed distribution with a few high values, while the Q-Q plot shows deviation from normality, mainly in the upper quantiles.

"Households" feature displays a distribution similar to "Population," with a right skew in the histogram and the Q-Q plot indicating deviations from normality. The "MedianIncome" histogram suggests a right-skewed distribution with most values between 2 and 10, and the Q-Q plot shows deviations from normality in the tails.

For the "MedianHouseValue," the histogram exhibits a multimodal distribution, and the Q-Q plot reflects deviations from the normal distribution, particularly in the upper quantiles. The features "Longitude" and "Latitude" both show bimodal distributions in their histograms, with Q-Q plots indicating significant deviations from normality.

The bar chart as shown in figure 14 illustrates the percentage distribution of students based on their scholarship status, gender, and academic status, categorized as Dropout, Enrolled, and Graduate. For male scholarship holders, approximately 10% are dropouts, around 13% are currently enrolled, and about 8% have graduated. Female scholarship holders show a different distribution, with roughly 12% being dropouts, 5% currently enrolled, and a significant 30% having graduated. Among male students without scholarships, the dropout rate is notably higher at around 47%, with about 20% enrolled and a small 5% graduated. Female students without scholarships exhibit a dropout rate of approximately 30%, with about 35% enrolled and around 22% having graduated. This chart highlights the varying educational outcomes based on gender and scholarship status, with female scholarship holders having the highest graduation rate.

The bar graph in Figure: 13 shows the Student Status. The count of Dropout was 1180, Graduate was 1610 and Enrolled was 600.

V. DISCUSSION AND ANALYSIS

This analysis aims to compare the performance of three Naive Bayes classification models—Categorical, Gaussian, and Hybrid—applied to the dataset for predicting student dropout and academic success. The evaluation of these models is based on confusion matrices, precision, recall, f1-score, and accuracy metrics. The analysis seeks to determine which model offers the best predictive performance and insights into student outcomes.

The Gaussian Naive Bayes model demonstrates strong performance, particularly for the 'Graduate' category, with a precision of 0.77, recall of 0.82, and f1-score of 0.80. The model achieves an overall accuracy of 70%. The confusion matrix indicates that the model is effective in correctly identifying graduates, but it struggles with the 'Dropout' and 'Enrolled' categories, showing lower recall values of 0.61 and 0.53, respectively. This suggests that while the model is generally accurate, it may not be as reliable in distinguishing between students who will drop out or remain enrolled.

The Categorical Naive Bayes model shows lower overall performance compared to the Gaussian model, with an accuracy of 62%. The model has a recall of 0.87 for the 'Graduate' category, indicating that it is good at identifying students who will graduate. However, its recall for the 'Enrolled' category is particularly low at 0.05, reflecting a significant challenge in predicting students who will remain enrolled. The precision for the 'Dropout' category is 0.66, but the low f1-score of 0.09 for the 'Enrolled' category points to poor performance in predicting ongoing students. This model's confusion matrix highlights its difficulty in correctly classifying 'Enrolled' students, which affects its overall effectiveness.

The Hybrid Naive Bayes model combines features of both Gaussian and Categorical models, resulting in the highest overall accuracy of 72%. This model exhibits a balanced performance across all categories. It achieves a precision of 0.81 and a recall of 0.70 for the 'Dropout' category, along with a recall of 0.93 for the 'Graduate' category, indicating a high ability to identify students who will graduate. The f1-score of 0.28 for the 'Enrolled' category is still low but improved compared to the Categorical model. The confusion matrix reveals that the Hybrid model reduces misclassification errors for all categories, providing a more reliable prediction across different student outcomes.

The analysis reveals that the Hybrid Naive Bayes model outperforms the other two models in terms of overall accuracy and balanced performance across different categories. The Gaussian model is particularly strong in predicting graduates but falls short in distinguishing dropouts and enrolled students. The Categorical model struggles significantly with the 'Enrolled' category, which limits its practical application. The confusion matrices for each model underscore the importance of considering multiple metrics when evaluating classification models. Precision, recall, and f1-score provide a comprehensive understanding of each model's strengths and weaknesses. For instance, the high recall for graduates in both Gaussian and Hybrid models suggests these models can be effectively used to predict successful student outcomes, which is crucial for educational institutions aiming to enhance graduation rates.

VI. CONCLUSION

In this lab, we investigated the effectiveness of three Naive Bayes classification models—Categorical, Gaussian, and Hybrid—on predicting student outcomes, including dropout, enrollment, and graduation. The models were evaluated based on several key metrics: confusion matrices, precision, recall, f1-score, and overall accuracy. Our goal was to determine which model provided the most reliable predictions and offered the greatest insight into student performance. This endeavor was driven by the necessity to enhance educational outcomes through precise identification of at-risk students, thereby enabling timely interventions.

The Categorical Naive Bayes Model provided initial insights but demonstrated limitations in its predictive power, particularly for students who remain enrolled. While it managed to achieve moderate success in predicting graduates and dropouts, its inability to accurately identify enrolled students highlighted its constrained applicability in a real-world educational context. This model's performance indicated that relying solely on categorical data might overlook nuances captured by other data types, leading to suboptimal intervention strategies.

Conversely, the Gaussian Naive Bayes Model offered a more balanced approach by incorporating continuous data. This model showed improved prediction capabilities for both dropouts and graduates, suggesting a higher sensitivity to the variations within continuous predictors such as admission grades and age at enrollment. However, similar to the Categorical model, it still faced challenges in accurately predicting enrolled students. This shortcoming underscores the complexity of student enrollment dynamics and suggests that continuous data alone, while valuable, may not be sufficient for comprehensive prediction.

The Hybrid Naive Bayes Model emerged as the most robust and reliable predictor among the three. By combining elements of both Gaussian and Categorical models, it harnessed the strengths of each approach to deliver a more nuanced and accurate prediction across all student categories. The Hybrid model's balanced performance indicates that an integrative approach, leveraging both categorical and continuous data, can significantly enhance predictive accuracy. This model's success underscores the importance of considering a multifaceted view of student data, encompassing diverse attributes such as socioeconomic factors, academic history, and personal circumstances.

This comparative analysis highlights the critical role of model selection in predictive analytics within the educational domain. The insights gained from this lab reinforce the need

for educational institutions to adopt sophisticated, integrative models like the Hybrid Naive Bayes to better understand and predict student outcomes. Such models not only improve the accuracy of predictions but also provide a deeper understanding of the factors influencing student success. This, in turn, can inform the development of targeted support programs, ultimately fostering higher retention and graduation rates.

REFERENCES

[1] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.

[2] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *AAAI-98 workshop on learning for text categorization*, vol. 752. Citeseer, 1998, pp. 41–48.

[3] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *European conference on machine learning*. Springer, 1998, pp. 4–15.

[4] R. Lopez, C. Romero, and S. Ventura, "Understanding complement naive bayes," *International Journal of Computational Intelligence Systems*, vol. 7, no. 3, pp. 542–553, 2014.

[5] G. S. Li and W. Ding, "Data mining techniques for marketing, sales, and customer relationship management," in *Proceedings of the International Conference on Data Mining*. Springer, 2018, pp. 281–289.

OM PRAKASH SHARMA is currently pursuing his undergraduate degree in Computer Engineering at IOE, Thapathali Campus. His research interests encompass various areas, including computational genomics, cloud computing, and system designing. Additionally, he is intrigued by artificial intelligence, cybersecurity, data analytics, machine learning, Internet of Things (IoT), quantum computing, software development methodologies, network security, computer vision, and natural language processing.(THA077BCT030)

ATUL SHREEWASTAV is currently pursuing his undergraduate degree in Computer Engineering at IOE, Thapathali Campus. His research interests encompass various areas, including Machine Learning Techniques, Natural Language Processing, and Computer Vision. Additionally, he is intrigued by automation, attention mechanism in transformers, agile development methodologies, advancement in neural net architectures, linux.(THA077BCT013)

APPENDIX A
CONFUSION MATRIX

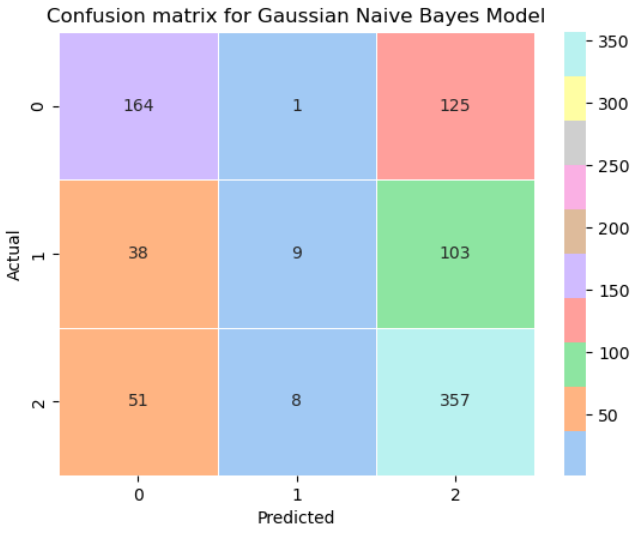

FIGURE 1. Confusion Matrix of Gaussian Model

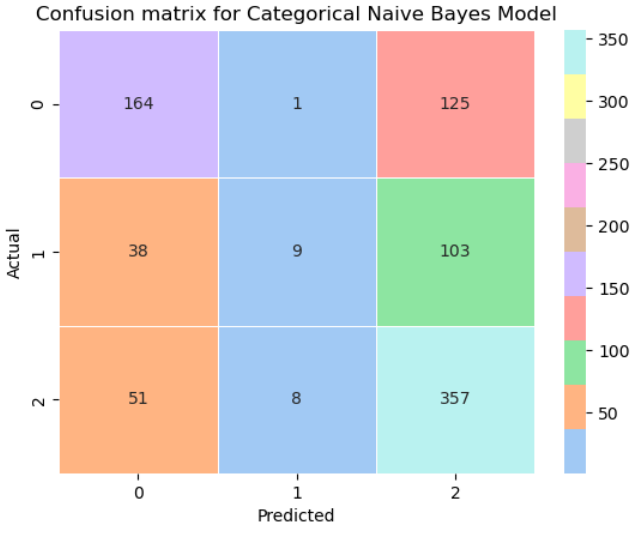

FIGURE 2. Confusion Matrix of Categorical Model

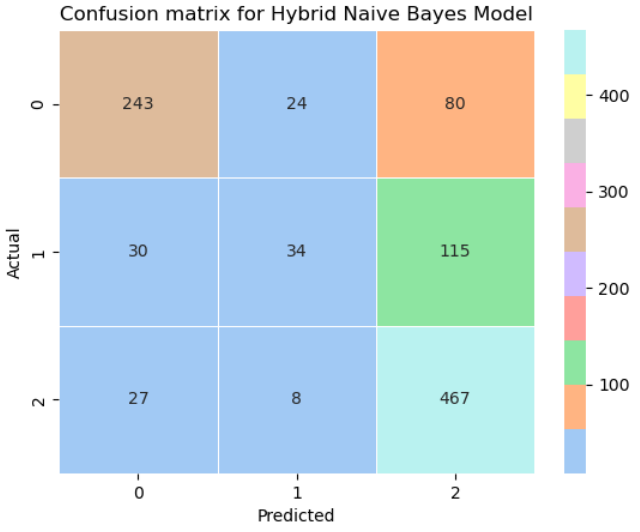

FIGURE 3. Confusion Matrix of Hybrid Model

APPENDIX B
HISTOGRAM AND Q-Q PLOT

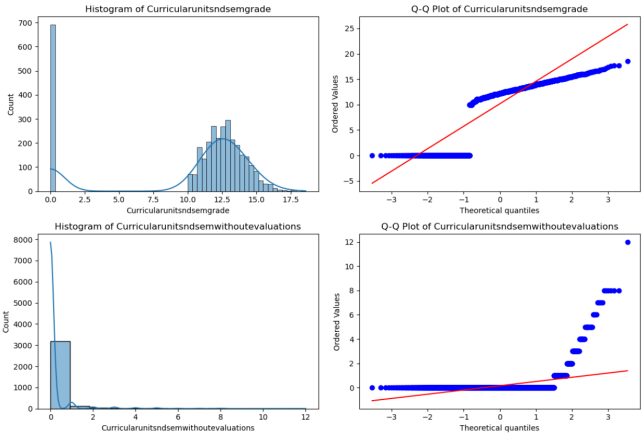

FIGURE 4. Curricular Unit Semester Grade and Curricular Unit Semester Grade without Evaluation

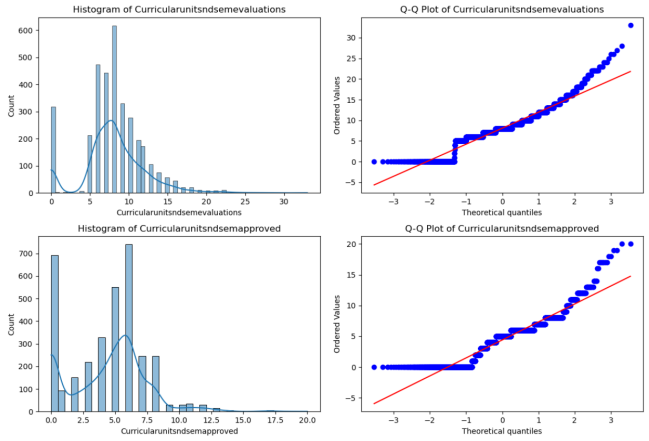

FIGURE 5. **Curricular Unit Semester with Evaluation and Curricular Unit Semester Semester Approved**

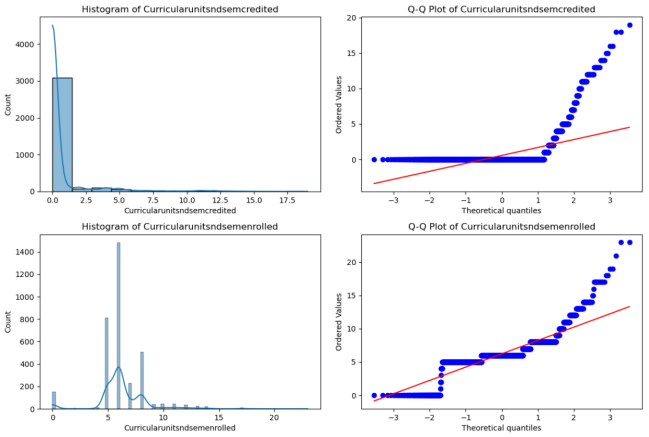

FIGURE 6. **Curricular Unit Semester Credited and Curricular Unit Semester Semester Enrolled**

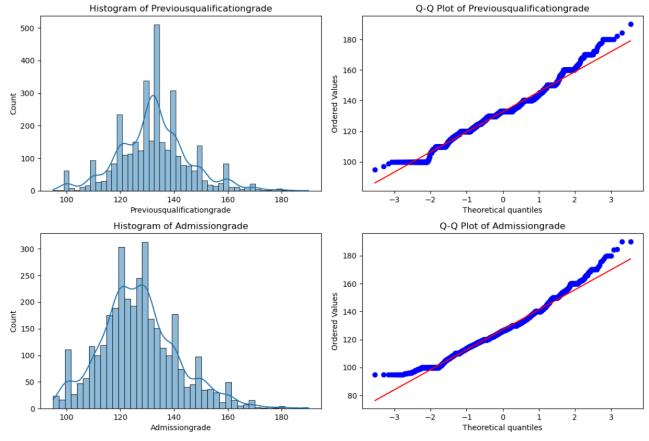

FIGURE 7. **Previous Qualification Grade and Admission Grade**

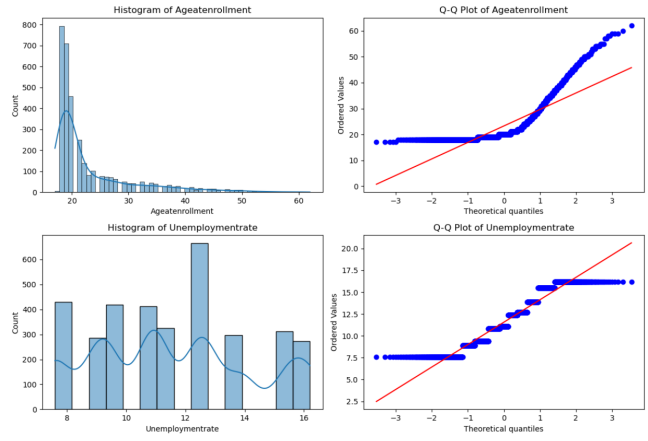

FIGURE 8. **Age and Unemployment Rate**

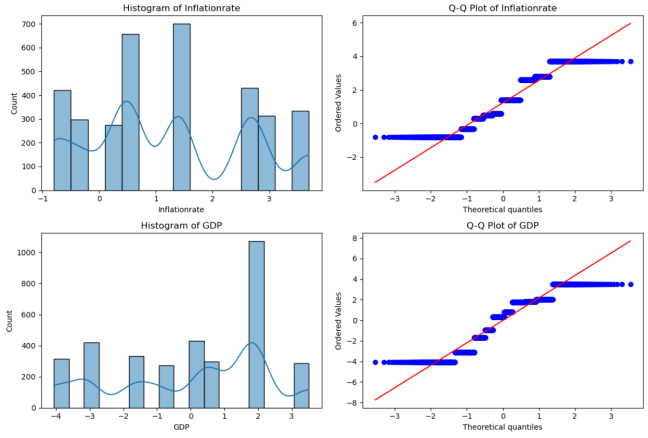

FIGURE 9. **Inflation Rate and GDP**

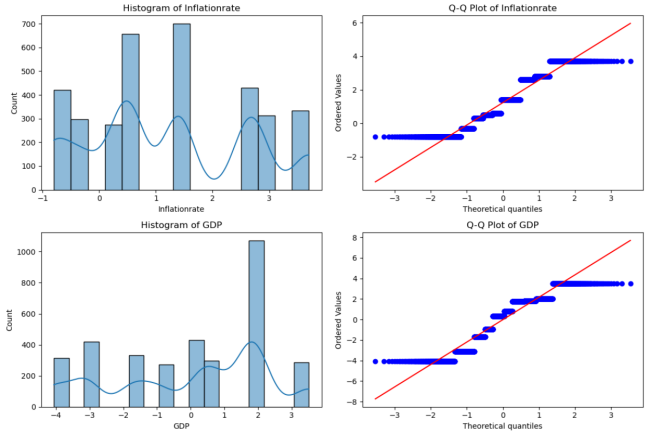

FIGURE 10. **Curriculum Units Semester Grade and Curricular Semester Grade without Evaluation**

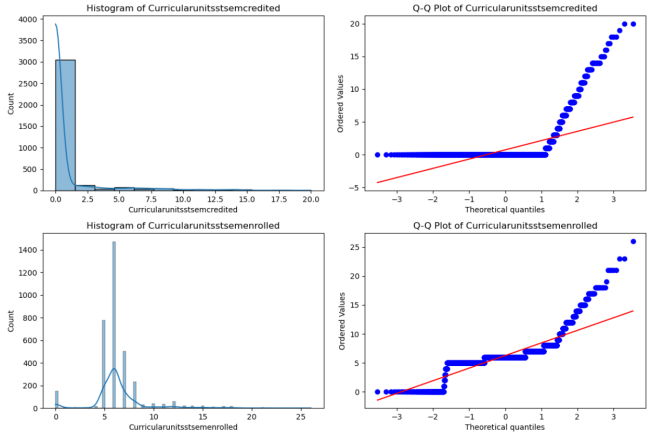

FIGURE 11. **Curriculum Units Semester Credited and Curricular Semester Grade without Enrolled**

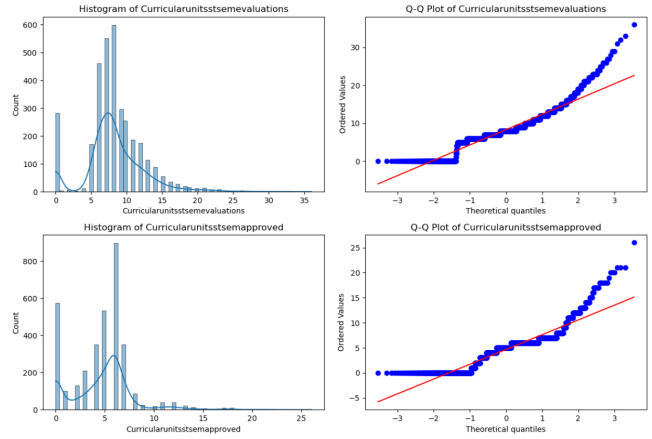

FIGURE 12. **Curriculum Units Semester Evaluations and Curricular Semester Approved**

APPENDIX C
DATA VISUALIZATION PLOTS

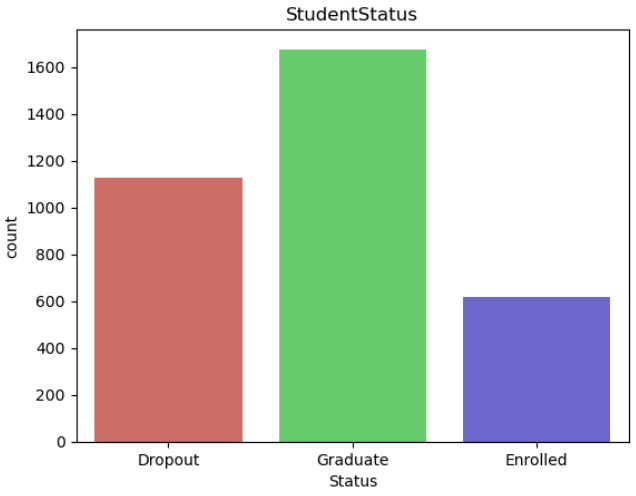

FIGURE 13. **Count of Student Status**

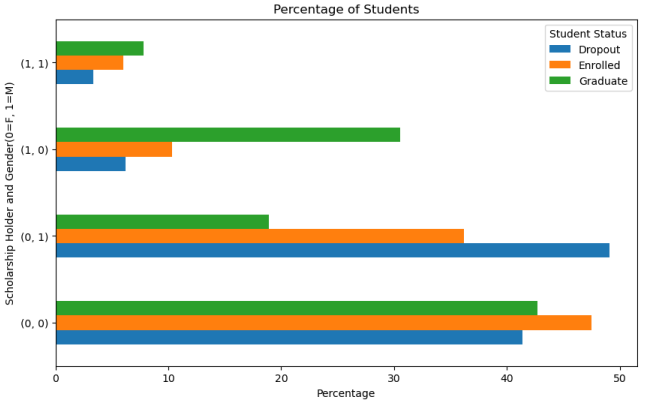

FIGURE 14. **Bar Graph Grouped by Scholarship Holder and Gender with Student Status**

APPENDIX D
SYSTEM BLOCK DIAGRAM

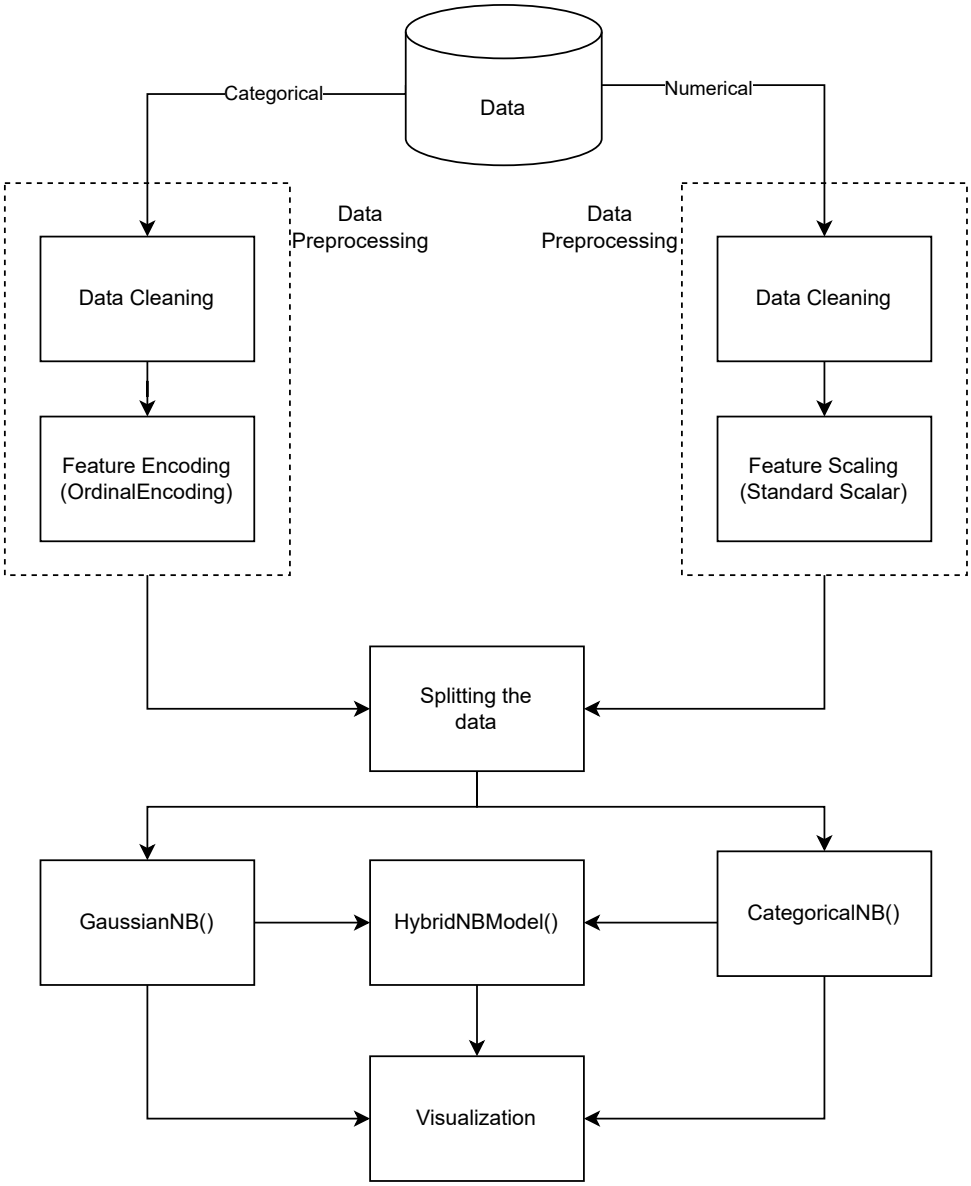

FIGURE 15. Block Diagram for Decision Tree Classifier