IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication 25 July 2023, date of current version 25 July 2023.

Digital Object Identifier

Predicting Customer Attrition in a Banking, A Naive Bayes Classifier Approach to Multivariate Classification

PRABIN BOHARA[1], PRABIN SHARMA POUDEL[2], AND SANTOSH PANDEY.[3]
[1]Institute of Engineering, Thapathali Campus, Kathmandu, Nepal (e-mail: prabinbohara10@gmail.com)
[2]Institute of Engineering, Thapathali Campus, Kathmandu, Nepal (e-mail: prabinsharmapoudel@gmail.com)
[3]Institute of Engineering, Thapathali Campus, Kathmandu, Nepal (e-mail: suntoss.pandey@gmail.com)

* All Authors contributed equally to this work.

ABSTRACT This project explores the application of the Naive Bayes classifier, a powerful machine learning algorithm, to predict customer churn in a banking context, using the 'BankChurners' dataset. The dataset encompasses a diverse range of attributes, including textual, numerical, and categorical features, such as age, gender, education level, and financial information. Leveraging the concept of multivariate analysis, the Naive Bayes classifier takes into account the interdependencies among multiple variables to make informed predictions about customer attrition. Through a comprehensive pre-processing of the dataset, the Naive Bayes algorithm was trained and evaluated, employing essential performance metrics like accuracy, precision, recall, and F1-score. The model showcased robust predictive capabilities, making it a promising tool for real-world prediction problems in customer churn. This report presents the outcomes, model insights, and implications for the bank's customer retention strategy, offering a comprehensive analysis of the Naive Bayes classifier's efficiency in multivariate customer churn prediction within a banking environment.

INDEX TERMS Bayesian Theorem, Conditional Independence, Conditional Probability, Posterior

I. INTRODUCTION

THIS report explores the application of the Naive Bayes Algorithm to solve the customer churn prediction problem in the banking industry. The Naive Bayes Algorithm is a fundamental machine learning technique based on Bayes' probability theory, widely used for text classification, spam filtering, and sentimental analysis. It is a probabilistic classifier that learns the probability of objects and their features, enabling effective classification of high-dimensional datasets. We leverage the power of the Naive Bayes Algorithm to develop a classification model that can accurately predict customer churn in a banking scenario.

Customer churn, also known as attrition, is a critical challenge faced by banks in retaining their valuable customers. Predicting customer churn is crucial for financial institutions to identify and retain at-risk customers, allowing them to implement targeted retention strategies. Through this project, we aim to showcase the simplicity and effectiveness of the Naive Bayes Algorithm in classifying bank customers based on their attributes and predicting potential churners. By applying the Bayes theorem, we analyze the probability of customers belonging to different churn categories and gain insights into the factors contributing to customer attrition. Our findings can assist the banking industry in devising data-driven customer retention strategies and enhancing customer satisfaction and loyalty.

II. METHODOLOGY

A. THEORY

Naïve Bayesian Classifier is one of the probability based method that is built upon Bayes Theorem which particularly suits for the tasks involving categorization and pattern recognization. The overall formulation of this algorithm is repose on the assumption of conditional independence. Naive Bayes algorithm all starts with the hypothesis that corresponds with each class of target value. The hypothesis is then tested and chosen on the basis of maximum posterior probability given the inputs as prior.

1) Bayes Theorem

Two random variables X and Y of any type are presented in terms of Marginal Probability as $P(X)$ and $P(Y)$, Joint

Probability as $P(X, Y)$, and Conditional Probability as $P(X|Y)$. The Conditional Probability $P(X|Y)$ is given by,

$$P(X|Y) = \frac{P(X, Y)}{P(Y)} \tag{1}$$

Similarly, the Conditional Probability $P(Y|X)$ is given by,

$$P(Y|X) = \frac{P(Y, X)}{P(X)} \tag{2}$$

The Joint Probability can also be obtained as,

$$P(X, Y) = P(X|Y) \cdot P(Y) \tag{3}$$

$$P(Y, X) = P(Y|X) \cdot P(X) \tag{4}$$

From the above equations, it can be inferred that,

$$P(Y|X) \cdot P(X) = P(X|Y) \cdot P(Y) \tag{5}$$

$$P(Y|X) = \frac{P(X|Y) \cdot P(Y)}{P(X)} \tag{6}$$

Thus, Bayes Theorem gives the expression for computing the marginal probability $P(Y)$ given the condition X.

Utilizing this theorem, we can incorporate the prior $P(Y)$, evidence $P(X)$, likelihood $P(X|Y)$, and posterior $P(Y|X)$ probabilities. The resulting posterior probability describes the updated belief in hypothesis Y being positive, conditioned on the evidence X.

2) Bayes Theorem for Classification

In the context of classification, Bayes' theorem allows us to make predictions about the class label Y of an observation X based on its attribute set. The goal is to compute the posterior probability of each class given the observed attribute values and then choose the class with the highest posterior probability as the predicted class for the observation.

Mathematically, Bayes' theorem for classification can be written as:

$$P(H|X) = \frac{P(X|H) \cdot P(H)}{P(X)} \tag{7}$$

Where:

- $P(H|X)$ is the posterior probability of hypothesis H given the attribute set X. This hypothesis H represents the probability of a class instance being the true class for the observation X.
- $P(X|H)$ is the class-conditional probability of the attribute set X given hypothesis H is positive. It represents the likelihood of observing the attribute set X when an instance class is applied.
- $P(H)$ is the prior probability of a class of attribute X. It represents our initial belief or knowledge about the probability of the class being true, before considering any specific attribute set X.
- $P(X)$ is the marginal probability of the attribute set X. It represents the overall probability of observing the attribute set X, regardless of the class.

The intuition behind Bayes' theorem for classification is similar to the general Bayes' theorem. We update our belief about the probability of a class being true (the posterior probability) based on the likelihood of observing the attribute set X for that class (the class-conditional probability) and our prior belief about the probability of that class (the prior probability). The denominator $P(X)$ serves as a normalization factor to ensure that the probabilities sum to 1.

To classify an observation X, we compute the posterior probability $P(H|X)$ for each possible class m, and then choose the class with the highest posterior probability as the predicted class for the observation.

It's important to note that estimating the class-conditional probabilities $P(X|H)$ and the prior probabilities $P(H)$ is a crucial step in applying Bayes' theorem for classification. Various algorithms, such as Naïve Bayes classifiers and Bayesian belief networks, employ different strategies to estimate these probabilities from training data.

Bayes' theorem is a foundational concept in probabilistic classification and is widely used in machine learning and data analysis for making predictions and inference based on observed data and prior knowledge.

3) Conditional Independence

Conditional independence is a concept in probability theory and statistics that describes the independence of two random variables, given the knowledge of a third random variable. It means that the occurrence or value of one random variable does not affect the occurrence or value of the other random variable, given the knowledge of the third variable.

Mathematically, two random variables X and Y are conditionally independent given a third random variable Z if the conditional probability of X given both Y and Z is equal to the conditional probability of X given Z alone. This can be represented as:

$$P(X|Y, Z) = P(X|Z) \tag{8}$$

In other words, knowing the value of Y provides no additional information about X, beyond what is already known from Z.

4) Naïve Bayes Classifier

The Naïve Bayes classifier is a popular probabilistic classification algorithm based on Bayes' theorem. It is simple, efficient, and often performs surprisingly well on a wide range of classification tasks, especially when dealing with high-dimensional data. The principle behind the Naïve Bayes classifier lies in the assumption of conditional independence between attributes and given the class label, which simplifies the estimation of class-conditional probabilities. This is the "naïve" part of the classifier. While this assumption may not hold in many real-world scenarios, the classifier can still be

surprisingly effective in practice, especially when the number of attributes is large.

The key assumption of conditional independence allows us to express the joint probability of the attribute set X and the class label Y as a product of individual conditional probabilities:

$$P(X,Y) = P(X_1|Y) \cdot P(X_2|Y) \cdot ... \cdot P(X_d|Y) \cdot P(Y) \quad (9)$$

where $X_1, X_2, ..., X_d$ are the d attributes in the attribute set X.

In the context of the Naïve Bayes classifier, conditional independence plays a central role in simplifying the computation of class-conditional probabilities. The Naïve Bayes classifier assumes that all attributes are conditionally independent given the class label, which allows it to estimate the class-conditional probabilities more efficiently.

Similar to the general Bayes' theorem, to classify an observation X, the Naïve Bayes classifier computes the posterior probability $P(Y|X)$ for each possible class Y, and then chooses the class with the highest posterior probability as the predicted class for the observation.

Estimating the conditional probabilities $P(X|Y)$ from the training data is a crucial step in building the Naïve Bayes classifier. The estimation methods depend on the types of attributes (categorical or continuous) in the dataset.

5) Conditional Probabilities for Categorical Attributes

For categorical attributes, the conditional probabilities in the Naïve Bayes classifier are estimated by calculating the fraction of training instances in each class that take on a particular attribute value. Let's consider a categorical attribute X_a with n possible values $\{r_1, r_2, ..., r_n\}$, and a class variable Y with m possible values $\{g_1, g_2, ..., g_m\}$. The conditional probability $P(X_a = r_i|Y = g_j)$ is estimated as follows:

$$P(X_a = r_i|Y = g_j) = \frac{\#\,|X_a = r_i \text{ and } Y = g_j|}{\#\,|Y = g_j|} \quad (10)$$

In other words, it is the ratio of the number of training instances where X_a takes on the value r_i and Y takes on the value g_j to the total number of training instances where Y takes on the value g_j.

6) Conditional Probabilities of Continuous Attributes

For continuous attributes, there are two common approaches to estimating the class-conditional probabilities in the Naïve Bayes classifier:

Approach 1: Discretize and Assume Uniform Probability

One approach is to discretize the continuous attribute X_a into intervals and then assume a uniform probability distribution within each interval for each class. This transforms the continuous attribute into an ordinal attribute. The conditional probability $P(X_a \text{ in interval } I|Y = g_j)$ is estimated as follows:

$$P(X_a \text{ in } I|Y = g_j) = \frac{\#|X_a \text{in} I \text{ and } Y = g_j|}{\#\,|Y = g_j|} \quad (11)$$

The intervals can be determined using various techniques, such as equal-width binning or equal-frequency binning. The assumption of uniform probability distribution within each interval simplifies the computation of probabilities.

Approach 2: Assume a Parametric Probability Distribution

Another approach is to assume a certain form of probability distribution for the continuous attribute, such as the Gaussian (Normal) distribution. For each class g_j, the distribution is characterized by two parameters: the mean (μ) and the variance (σ^2). The class-conditional probability $P(X_a = x|Y = g_j)$ is then estimated using the Gaussian probability density function (PDF) for attribute X_a, given its mean and variance for that class:

$$P(X_a = x|Y = g_j) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (12)$$

The parameters μ and σ^2 can be estimated from the sample mean and variance of the attribute values in the training data for each class.

Both of these approaches allow the Naïve Bayes classifier to handle continuous attributes effectively and incorporate them into the probabilistic model for classification. The choice of approach depends on the nature of the data and the specific problem at hand.

7) Zero Conditional Probability Case

In the Naïve Bayes classifier, there is a potential issue when estimating conditional probabilities for categorical attributes. If a certain attribute value does not appear in the training data for a particular class, the conditional probability for that attribute value in that class would be zero. This poses a problem because a zero conditional probability would cause the entire posterior probability to be zero, making it impossible to classify any test instance with that attribute value for the given class.

a: M-Estimate

To address this problem, the M-estimate is a method that adds a small constant value (m) to the count of occurrences of each attribute value for each class. This is equivalent to considering that we have observed m pseudo-instances of each attribute value in each class, regardless of whether they appear in the training data. The constant m is known as the equivalent sample size and is a hyperparameter that controls the smoothing effect.

The M-estimate formula for the conditional probability $P(X_a = x|C_i)$ is given as:

$$P(X_a = x|C_i) = \frac{\text{Count}(X_a = x, C_i) + mp}{\text{Count}(C_i) + m \cdot N} \quad (13)$$

where:

Count$(X_a = x, C_i)$ is the number of instances in the training set where attribute X_a has value x and belongs to class C_i.

Count(C_i) is the number of instances in the training set belonging to class C_i.

N is the total number of unique attribute values for X_a.

m is the equivalent sample size hyperparameter, a small positive constant.

p is the prior probability of the attribute value x given class C_i (usually set to $\frac{1}{N}$).

The value of m determines the strength of smoothing. Smaller values of m provide stronger smoothing, while larger values of m reduce the smoothing effect. In practice, m is often set to a small positive constant, such as 1.

b: Laplacian (Add-One) Smoothing

Laplacian smoothing, also known as add-one smoothing, is a specific case of M-estimate where m is set to 1. This technique adds one pseudo-count to the occurrences of each attribute value for each class, making sure that no conditional probability is exactly zero.

The Laplacian-smoothed formula for the conditional probability $P_{\text{lap},k}(X_a = x)$ is as follows:

$$P_{\text{lap},k}(X_a = x) = \frac{\text{Count}(X_a = x) + k}{N + k \cdot |X_a|} \tag{14}$$

where:

Count$(X_a = x)$ is the number of instances in the training set where attribute X_a has value x.

N is the total number of instances in the training set.

$|X_a|$ is the number of unique attribute values for X_a.

k is the smoothing parameter (usually set to 1, hence "Add-One" smoothing).

Laplacian smoothing ensures that each attribute value has a nonzero probability in each class, which prevents the issue of zero probabilities and allows the classifier to handle unseen attribute values during classification.

Both M-estimate and Laplacian smoothing are effective ways to deal with the zero probability problem in the Naïve Bayes classifier, providing a more robust and stable estimation of conditional probabilities, especially when the training data is limited. The choice between M-estimate and Laplacian smoothing depends on the specific requirements of the classification task and the desired level of smoothing.

B. SYSTEM BLOCK DIAGRAM

The system architecture for the Naive Bayes Classifier, as depicted in Figure 1, outlines the sequence of blocks involved in the classification process. For the visualization of the collected dataset, various charts like bar graph, scatter plot, Pearson correlation heatmap, histogram of features, etc are used. The initial phase of data preprocessing includes handling diverse datasets comprising textual, numeric, and categorical attributes. Textual data is transformed using techniques like tokenization and vectorization to convert it into numerical form, enabling it to be processed by the classifier. Numeric features are standardized or normalized to bring them to a common scale, ensuring equal contribution to the classification. Categorical variables are encoded to numerical values, facilitating their inclusion in the model. Data visualization techniques are employed to gain insights into the data distribution and relationships between features.

After preprocessing, the dataset is divided into training and testing sets. The Naive Bayes Algorithm is trained using the training data, where it learns the probability of objects and their features for different classes. For training on the data, different Bayesian models such as Gaussian Naïve Bayes or Categorical Naïve Bayes models are used, whose performance can be compared. The classifier's performance is evaluated using various metrics, including accuracy, precision, recall, and F1-score. Hyperparameter tuning can be applied to optimize the algorithm's performance. The trained model can then be used to make predictions on the testing data, classifying objects into predefined classes based on their features. The model's performance is visualized and compared through the use of confusion matrix visualization or through the classification report summary.

In summary, the Naive Bayes Classifier system architecture follows a series of steps, including data preprocessing, model training, evaluation, and prediction. Through effective handling of textual, numeric, and categorical data, the

TABLE 1: Algorithm: Naïve Bayes Classifier

Training Phase

Given a labeled training dataset with N instances and M attributes, calculate the prior probabilities $P(C_i)$ for each class C_i and the conditional probabilities $P(X_a|C_i)$ for each attribute X_a, given the class C_i.

For each class C_i:

- Count the number of instances belonging to class C_i, denoted as $N(C_i)$.
- Calculate the prior probability $P(C_i)$ as the ratio of $N(C_i)$ to the total number of instances N.

For each attribute X_a and each class C_i:

- If X_a is a categorical attribute:
  - Count the number of instances in class C_i having attribute value x, denoted as $N(X_a = x, C_i)$.
  - Calculate the conditional probability $P(X_a = x|C_i)$ using M-estimate or Laplacian smoothing.
- If X_a is a continuous attribute:
  - Assume a probability distribution for X_a (e.g., Gaussian) and estimate its parameters (mean and variance) from the instances in class C_i.
  - Calculate the conditional probability $P(X_a|C_i)$ using the assumed distribution and the estimated parameters.

Prediction Phase

Given a test instance X' with attribute values $\{X'a, X'b, ..., X'n\}$, predict its class label based on the calculated posterior probabilities for each class C_i.

For each class C_i:

- Calculate the joint probability $P(X'a, X'b, ..., X'n|C_i)$ as the product of the conditional probabilities $P(X_a|C_i)$ for each attribute X_a.
- Calculate the posterior probability $P(C_i|X')$ using Bayes' theorem:

$$P(C_i|X') = \frac{P(X'a, X'b, ..., X'n|C_i) \cdot P(C_i)}{P(X')}.$$

classifier demonstrates its capability in solving classification problems. The architecture ensures a systematic approach to accurately classify objects and enables informed decision-making in various applications like sentimental analysis, spam filtration, and customer churn prediction in banking.

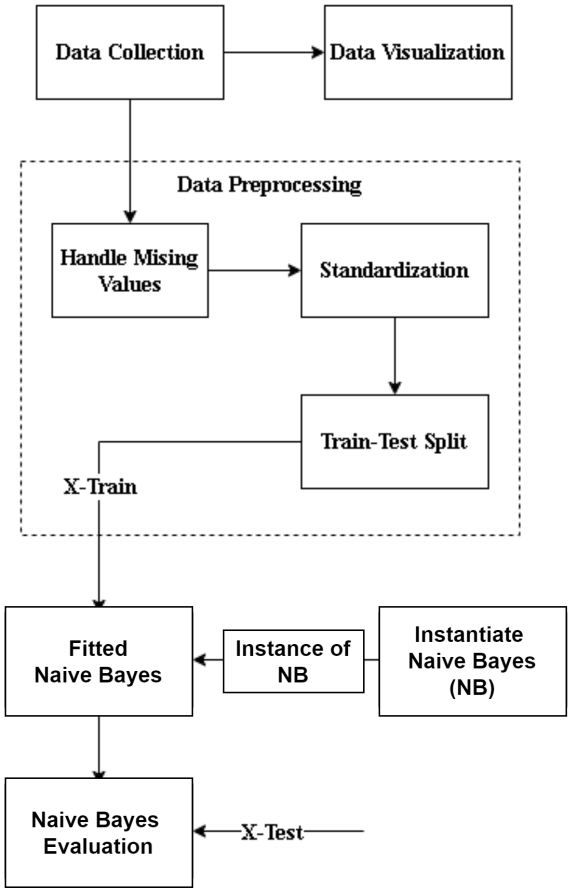

FIGURE 1: System Architecture of Naive Bayesian Classifier

C. INSTRUMENTATION

The implementation of Naïve Bayes Classifier for "Bankchurners" predicton in our study involved the use of several libraries and tools to facilitate data visualization and mathematical operations.

1) Pandas

Pandas is a powerful Python library used for data manipulation. Pandas played a crucial role in data preprocessing and exploration for the Naive Bayes classifier. We utilized its powerful DataFrame data structure to load, manipulate, and analyze the dataset. Pandas' read_csv() function allowed us to efficiently read the CSV file and create a DataFrame containing the data. Throughout the project, Pandas' selection and filtering capabilities enabled us to focus on relevant attributes and observations during preprocessing. Handling missing values was an essential step in the data preprocessing pipeline. Pandas' fillna() method was used to replace missing data with appropriate values or imputation techniques. Additionally, Pandas' get_dummies() function facilitated one-hot encoding for categorical variables, ensuring they were correctly represented for the Naive Bayes classifier.

2) Matplotlib and Seaborn

Matplotlib and Seaborn are visualization libraries in Python, which were instrumental in presenting our data in a graphical format. Matplotlib's functions, such as scatter(), hist(), and imshow(), were utilized for these visualizations. For instance, scatter() was used to create scatter plots, visualizing the relationship between two numerical features. Histograms were generated using hist(), providing insights into the distribution of data across specific numerical ranges. We also employed imshow() to display heatmaps, offering a visual representation of correlation matrices. Seaborn, on the other hand, enhanced the aesthetics of our plots and provided a high-level interface for creating complex visualizations with fewer lines of code. With the help of these libraries, we were able to create informative visualizations, gaining insights into the characteristics of our datasets and the performance of our classifiers.

3) NumPy

Numpy is a fundamental library in Python that provides support for large, multi-dimensional arrays and matrices, along with an extensive collection of mathematical functions to operate on these arrays. In this project, Numpy played a vital role in handling numerical data. We utilized Numpy's array data structure to efficiently store and manipulate our datasets. For instance, Numpy's mean() and std() functions were applied to compute the mean and standard deviation of numerical features in the dataset, aiding in data preprocessing and descriptive statistics. Numpy's slicing and indexing capabilities allowed us to extract specific columns and rows, enabling focused data exploration and feature selection. Moreover, Numpy's reshape() function was employed to transform data into appropriate dimensions for model training and evaluation.

4) Scikit-learn

Scikit-learn, or sklearn, was a fundamental component in implementing the Naive Bayes classifier in this project. It provided essential functionalities for data preprocessing, model training, and evaluation. To begin with, we utilized sklearn's preprocessing module to handle categorical data and numerical scaling. The LabelEncoder class from sklearn.preprocessing enabled us to convert categorical variables into numerical format, allowing compatibility with the Naive Bayes classifier. Additionally, the StandardScaler class was employed to standardize numerical features, ensuring all attributes had a similar scale, which is essential for the

Naive Bayes algorithm to make accurate predictions. Furthermore, scikit-learn offered a variety of Naive Bayes implementations, such as GaussianNB for continuous features and MultinomialNB for discrete features. We utilized the GaussianNB class to fit the Naive Bayes classifier to our dataset, given that it can handle numeric features. The ease of implementation and consistent API across different scikit-learn models streamlined the process of training and fitting the Naive Bayes classifier.

By utilizing these libraries, we were able to seamlessly integrate data visualization and mathematical operations into our Naïve Bayes implementation. This instrumentation allowed us to effectively analyse the datasets, visualize the results, and gain valuable insights from the prediction process.

D. WORKING PRINCIPLE

The Naive Bayes algorithm operates on the principle of Bayes' theorem and is designed for text classification and probabilistic predictions. The core idea behind Naive Bayes is to calculate the probability of an object belonging to a specific class based on its features and their associations with the target classes. In our implementation, we applied Naive Bayes to classify instances into different genres based on their textual, numeric, and categorical attributes.

1) Data Pre-processing

Pre-processing the dataset is a critical step in preparing it for Naive Bayes classification. As the algorithm primarily deals with numerical features, categorical and textual data require specific pre-processing techniques. For textual data, we utilized methods such as tokenization. If we were to handle other lengthy text handling techniques, we would have used preprocessing techniques like removing stopwords, and converting text to numerical vectors using techniques like TF-IDF (Term Frequency-Inverse Document Frequency). For numerical features, scaling or standardization was applied to ensure all attributes contribute equally to the classification process. Additionally, handling missing values was addressed using methods such as imputation, replacing the missing values with the mean or median, to avoid any bias in the model's performance. Categorical features were encoded into numerical format using techniques like Label Encoding or One-Hot Encoding. This transformation enables the Naive Bayes algorithm to work seamlessly with categorical data and make probabilistic predictions for classifying instances. By pre-processing the data, we guarantee that the Naive Bayes algorithm can effectively learn from the features and make accurate probabilistic predictions. The visualization of the pre-processed data provides insights into the transformed dataset, aiding in understanding its structure and the impact of pre-processing techniques on the input features. Overall, data pre-processing is an essential step in preparing the dataset for Naive Bayes classification, ensuring the algorithm's effectiveness in text classification tasks and probabilistic predictions.

2) Test-Train Split

This step involves partitioning the dataset into two separate sets: the training set and the testing set. The training set is used to train the classifiers on labeled instances, while the testing set is used to evaluate the classifiers' performance on unseen data. In our implementation, we use the train_test_split function from the scikit-learn library to split the pre-processed dataset into training and testing sets. The train_test_split function randomly shuffles the dataset and divides it into the specified proportions for training and testing. Commonly, the dataset is divided into a training set, which comprises around 70% to 80% of the data, and a testing set, which makes up the remaining 20% to 30%. The training set is used to fit the GaussianNB and CategoricalNB classifiers, allowing them to learn the underlying distributions and probabilities of features given class labels. On the other hand, the testing set is used to evaluate the classifiers' performance. By predicting the class labels of instances in the testing set and comparing them to the true labels, we can calculate evaluation metrics such as accuracy, precision, recall, and F1-score to assess how well the classifiers perform on unseen data.

3) Model Instantiation

In the working principle pipeline, after data pre-processing, we instantiate two types of Naive Bayes classifiers: Gaussian Naive Bayes (GaussianNB) and Categorical Naive Bayes (CategoricalNB). These classifiers are based on different probability distributions to handle numerical and categorical features, respectively. GaussianNB is well-suited for datasets with numerical features that follow a Gaussian (normal) distribution. In our implementation, we instantiate the GaussianNB classifier using the scikit-learn library. The GaussianNB model assumes that each class is associated with a Gaussian distribution of numerical feature values. It estimates the mean and standard deviation for each class and uses these parameters to calculate the conditional probability of a feature given the class label using the Gaussian probability density function. The fitting process for GaussianNB involves calculating the class priors (the probability of each class occurrence in the dataset) and the mean and standard deviation of each numerical feature for each class. During prediction, the classifier calculates the posterior probability of each class for a given instance using Bayes' theorem and then selects the class with the highest probability as the predicted class label. CategoricalNB is designed for datasets with categorical features. In our implementation, we use the CategoricalNB classifier from scikit-learn. This model assumes that each categorical feature follows a categorical distribution and estimates the probabilities of the feature's categories given each class label. The fitting process for CategoricalNB involves calculating the class priors and the probabilities of each category for each categorical feature in each class. During prediction, the classifier calculates the posterior probability of each class for a given instance using Bayes' theorem and then selects the class with the highest

probability as the predicted class label.

4) Fitting and Prediction

In this step, we use the training set to fit the GaussianNB and CategoricalNB classifiers to the data. This process involves estimating the probability distributions of features given class labels, which are crucial for making predictions on new, unseen data. For the GaussianNB classifier, which is suitable for continuous numeric data, we estimate the mean and variance of each feature for each class label. These parameters are then used to model the probability distributions of features given the class labels as Gaussian distributions. This enables the classifier to calculate the likelihood of an instance belonging to a particular class based on the observed feature values. For the CategoricalNB classifier, which is ideal for categorical data, we estimate the probability of each feature's categories given the class labels. This involves calculating the frequency of each category for each class label and then converting these frequencies into probabilities. The classifier then uses these probabilities to calculate the likelihood of an instance belonging to a particular class based on the observed categorical feature values. Once the classifiers are fitted to the training data, we proceed with the prediction step. In this step, we use the testing set, which the classifiers have never seen before, to make predictions. For each instance in the testing set, the classifiers use the learned probability distributions to calculate the posterior probability of each class label. The instance is then assigned to the class label with the highest posterior probability, which is the predicted class label. The fitting and prediction step is the heart of the Naive Bayes classifiers' functionality. By estimating probability distributions from the training data and making predictions based on these distributions, the classifiers can efficiently classify new instances into the appropriate class labels. This step is computationally efficient and well-suited for large datasets, making Naive Bayes classifiers a popular choice for various classification tasks in machine learning.

5) Model Evaluation

In the final step of the working principle pipeline, the results of the Naive Bayes classifiers are visualized and evaluated using various performance metrics. Confusion matrices are commonly employed to visualize the classification results, showing the true positive, true negative, false positive, and false negative counts for each class. Classification reports provide a comprehensive summary of performance metrics such as accuracy, precision, recall, and F1-score for each class. These visualizations and metrics allow us to assess the classifier's overall performance and identify areas for improvement. Other techniques that can be employed for performance evaluation include ROC curves and precision-recall curves. ROC curves visualize the trade-off between the true positive rate and the false positive rate, providing insights into the classifier's sensitivity to different threshold values. Precision-recall curves illustrate the precision and recall relationship, which is valuable for imbalanced datasets where the class distribution is skewed. Employing these additional techniques can further enhance the understanding of the classifier's performance and provide more nuanced insights into its behavior.

III. RESULTS

A. DISTRIBUTION OF DATA

The 'Bankchurners' dataset is helpful to draw useful insights to find out the reason behind the customer attrition by predicting the customers who are more likely to drop off. There are twenty three columns in the dataset where the target column is the 'Attrition Flag', where the value 1 signifies that the account is closed and the value 0 signifies if the account is not yet closed. It is seen that the existing customer is 84% and the attrited custometr is 16% out of the total data available, which is 8500 and 1627 respectively.

The remaining columns are the attribute features. One such feature is 'Clientnum' which refers to the unique identifier of the customer. This feature is dropped out from the dataframe. Another feature named 'Customer Age' signifies the age in years of the customers. There are no missing or mismatched values in this feature. The mean age is found to be 46.3 and the standard deviation is 8.02. Likewise, the minimum age is 26 and the maximum age is 73. The third attribute 'Gender' shows that there are 53% females and 47% males, with no mismatched or missing values. Another such feature is 'Dependent_count' which refers to the number of dependent customers in the available dataset. It is observed that the mean and standard deviation for this feature is 2.35 and 1.3 respectively. The feature 'Months_on_book' signifies the period of relationship with bank. It is seen that the mean and standard deviation is 35.9 and 7..99 respectively. Similarly, the minimum value is seen to be 13 and maximum value is 56. In terms of quartiles, the number 31,36,40 describes the 1st, 2nd and 3rd quartile respectively. The 'Total_Relationship_Count' is the total number of products held by the customer, where the mean and standard deviation value is seen to be 3.81 and 1.55 respectively. Likewise, the minimum and the maximum value are 1 and 6 respectively. For the feature 'Months_Inactive_12_mon' means the number of months inactive in the last 12 months. It has 0 as minimum value and 6 as maximum value. The 'Contacts_Count_12_mon' means the number of contacts in the last 12 months which have the mean and standard deviation value as 2.46 and 1.11 respectively, Another feature named 'Credit_limit' is a self-explanatory feature name where all five quartiles are obtained to be 1.44k, 2.56k,4.55k,11.1k, and 34.5k respectively. The feature 'Total_Revolving_Bal' means the total revolving balance on the credit card. All five quartiles on this features are obtained as 0,357,1276,1784 and 2517 respecctively. Likewise, the feature 'Avg_Open_To_Buy' is for the last 12 months, where the mean and standard deviation values are obtained to be 7.47k and 9.09k respectively. Another feature 'Total_Amt_Chng_Q4_Q1' means the change in transaction amount which is Q4 over Q1. All the five

quartiles are obtained to be 0, 0.63, 0.74, 0.86, 3.4. Further, 'Total_Trans_Amt' have the mean and standard deviation values 4.4k and 3.4k respectively. Another such feature 'Total_Trans_Ct' refers to the Total Transaction count spanning the last 12 months, which has the mean and standard deviation value 64.9 and 23.5 respectively. Similarly, the minimum and maximum value is obtained to be 10 and 139 respectively. The 'Total_Ct_Chng_Q4_Q1' feature refers to the change in the transaction count which is reffered to as Q4 over Q1. Another feature is 'Avg_Utilization_Ratio' which refers to average card utilzation ratio where the mean and standard deviation are obtained to be 0.27 and 0.28 respectively.

B. MODEL PERFORMANCE

The training on the Gaussian Naive Bayes Model yeilded 88.45% accuracy on the test set. A confusion matrix was plotted to better visualize the performance of the model. It was seen that the True Negative (TN) value was 0.94. Likewise, the True Positive (TP) was obtained to be 0.58, False Negative(FN) was obtained to be 0.06, and False Positive (FP) was seen to be 0.42. The classification report was printed to obtain in-depth analysis of the model's inference. The accuracy was seen to be 0.88 with the support of 2026. For Class 0 the precision, recall, F1 score, and support was seen to be 0.92,0.94,0.93 and 1699 respectively. F1 score give the harmonic mean of precision and recall, which are calculated using their respective formula. The scores for macro avg and weighted avg was seen to be 0.78 and 0.88.

The training on the Categorical Naive Bayes model required the discretization of 'Age' and 'Credit_limit' column. For that binning was done by two ways: equal width and equal frequency. For equal width binning, each age bins had labels from 0-10, 11-20 and so on. Likewise, the credit_bins has labels 0-1000, 1001-2000 and so on. It was trained on one instance of this model. From the confusion matrix, TP,TN, FP, and FN can also be implied. It was seen that TN value was 0.98, TP was 0.38, FP was 0.62 and FN was 0.02. The classification report was printed to obtain in-depth analysis of the model's inference. The accuracy was seen to be 0.88 with the support of 2023. For Class 0 the precision, recall, F1 score, and support was seen to be 0.89,0.98,0.93 and 1696 respectively. The scores for macro avg and weighted avg was seen to be 0.72 and 0.87 respectively. For the equal frequency binning, the values were divided into 5 bins. From the confusion matrix, it can be inferred that the TP, TN, FP and FN values were obtained to be 0.40, 0.98, 0.60 and 0.02. Similarly from the classification report, accuracy of 0.89 was obtained with the support of 2023. Likewise, for class 0, the precisioon, recall, F1 score and support values were 0.89, 0.98, 0.94 and 1696 respectively. Similarly, for the class 1, the preision, recall, F1 score and support were obtained to be 0.79, 0.40,0.53 and 327 respectively. Thus by dropping some irrelevant features, and utilizing some features for the training on the Naive Bayes Model, the customers that are more likely to undergo attrition can be predicted so that the the institution can prepare in advance.

TABLE 2: Classification Report for Gaussian Naive Bayes

	Precision	Recall	F1-score	Support
0	0.92	0.94	0.93	1699
1	0.66	0.58	0.62	327
Accuracy			0.88	2026
Macro avg	0.79	0.76	0.78	2026
Weighted avg	0.88	0.88	0.88	2026

TABLE 3: Classification Results

Model	Precision	Recall	F1-score	Support
genericNB	0.88	0.88	0.88	2026
categoricalNB (equal width)	0.87	0.88	0.87	2023
CategoricalNB (equal freq)	0.88	0.89	0.87	2023

IV. DISCUSSION AND ANALYSIS

The results obtained from the implementation of the Gaussian Naive Bayes classifier on the "BankChurners.csv" dataset revealed promising performance metrics. For the first classifier, True Positive (TP) value was relatively lower at 0.58, suggesting that the classifier had some difficulty in correctly identifying churned customers. It was also seen that for two classes the precision and recall value were very different. Such high values for Class 0 indicates that the model was highly accurate in identifying customers who are likely to stay with the bank. However, for Class 1 (churned customers), the precision (0.66) and recall (0.58) were relatively lower, resulting in a lower F1-score of 0.62. This suggests that the model had some difficulty in accurately identifying customers who are likely to churn. One possible reason for the observed difference in performance between the two classes could be the class imbalance present in the dataset, where the number of non-churned customers (Class 0) is significantly higher than the number of churned customers (Class 1). This class imbalance may have affected the model's ability to correctly identify churned customers, leading to a lower recall value for Class 1. To address this issue, further exploration of techniques such as oversampling or undersampling of the data could be considered. Additionally, it is essential to consider feature importance and the impact of each attribute on the classification results. Analyzing the relationships between features and the target variable can provide valuable insights into the customer attributes that significantly contribute to churn prediction.

The negative correlation between 'Attrition_Flag' and 'Total_Trans_Ct' indicates a strong association between customer activity and customer retention. Customers who are highly engaged and conduct a larger number of transactions are more likely to stay with the bank and not drop out. Active customers are likely to be satisfied with the bank's products and services, which motivates them to conduct more transactions. Their positive experiences with the bank may contribute to a stronger sense of loyalty and reduce the likelihood of switching to another institution. Many banks also offer rewards and incentives to customers who conduct

frequent transactions. These loyalty programs can further encourage customers to remain with the bank and benefit from the rewards associated with their activity. Also, high level of engagement indicates that the bank's offerings meet their financial needs effectively, which indicates Bank's good work with the customers. A higher "Total_Trans_Ct" may indicate that a customer has developed a sense of loyalty and trust towards the bank. Trust is a critical factor in customer retention, and customers who have built trust are less likely to consider alternatives. To retain more customers and reduce attrition rates, the bank should focus on strategies to increase customer engagement and promote active usage of its services. Offering personalized incentives, enhancing customer support, and continuously improving the overall customer experience can play a vital role in nurturing customer loyalty and reducing the likelihood of dropouts. By understanding and leveraging the relationship between customer activity and churn, the bank can make informed decisions and take proactive measures to ensure long-term customer retention and success. On the other hand, there are few irrelavant features such as 'Avg_Open_To_Buy' which are better left dropped from the dataframe so that the model can generalize better. It's because such features have approximately zero correlation with the 'Attrition_Flag'

It was seen that middle-aged people were most likely to drop out than other age groups. Likewise, women were most likely to drop out than men. Thus, middle-aged women had the highest attrition rate. The age range of 50+ often corresponds to a phase of life where individuals experience significant life events such as retirement, children moving out, or lifestyle changes. These life events could influence their banking needs and preferences, leading to churn if their current bank doesn't meet their evolving requirements. The quality of customer service provided by the bank can significantly impact customer loyalty. It's possible that middle-aged women have specific expectations and preferences for customer service, and if those expectations are not met, they might be more likely to switch banks. The bank's marketing and communication strategies may inadvertently influence the observed pattern. If marketing efforts are not well-targeted or resonate with middle-aged women, they might not feel connected to the brand and consider switching.

The higher accuracy observed in both categorical Naive Bayes models (equal width binning and equal frequency binning) compared to the Gaussian Naive Bayes model can be attributed to the nature of the data and the assumptions made by each model. Since the dataset contains a mix of categorical and numeric features, the Categorical Naive Bayes models can better capture the relationships between the categorical variables, leading to improved accuracy. On the other hand, the Gaussian Naive Bayes model assumes that the features follow a Gaussian (normal) distribution. This assumption may not hold well for some of the features in the dataset, which could lead to a decrease in accuracy. For instance, the 'Customer_Age' the feature is continuous, and its distribution might not be entirely Gaussian, affecting the model's performance. Likewise, it was seen that equal frequency binning for categorical models gave more accuracy than equal width binning. Equal frequency binning ensures that each bin has the same number of instances, which can be beneficial for capturing the distribution of data evenly. This approach helps in reducing the impact of outliers and skewed data, which could lead to a more robust model with better generalization performance. Equal-width binning, on the other hand, splits the data into bins of the same width. This method might not handle imbalanced data well, as some bins could have a significantly higher number of instances than others, leading to unequal representation in the training process. This imbalance might affect the model's ability to generalize to new, unseen data and could result in slightly lower accuracy compared to equal frequency binning. In summary, the higher accuracy observed in both categorical Naive Bayes models can be attributed to their ability to handle the specific data types present in the dataset more effectively.

V. CONCLUSION

In conclusion, the application of Gaussian Naive Bayes classification on the "BankChurners.csv" dataset has proven to be effective in tackling the churn prediction problem. Utilizing both Gaussian Naive Bayes and Categorical Naive Bayes classifiers, we obtained commendable accuracy scores, precision, recall, and F1 scores on the test data, showcasing the algorithm's robustness in handling diverse data types. In case of GaussianNB, the algorithm demonstrated notable accuracy with an accuracy score of 88.45%, indicating its proficiency in distinguishing between churned and non-churned customers based on the given features. By accurately classifying a significant proportion of non-churned customers, it enables businesses to focus on customer retention strategies and enhance customer loyalty. Moreover, the classifier's ability to handle diverse data types, such as textual and numeric attributes, makes it a valuable tool for churn prediction in various domains, including the banking industry. However, in cases where a more balanced classification performance is crucial, further fine-tuning of hyperparameters or exploration of alternative classification algorithms may be warranted.

REFERENCES

[1] Han, J., Kamber, M., & Pei, J. (2012). Data mining concepts and techniques, third edition. Morgan Kaufmann Publishers, Waltham, Mass. ISBN: 0123814790. http://www.amazon.de/Data-Mining-Concepts-Techniques-Management/dp/0123814790/ref=tmm_hrd_title_0?ie=UTF8&qid=1366039033&sr=1-1.

[2] Tan, P.-N., Steinbach, M., Karpatne, A., & Kumar, V. (2018). Introduction to Data Mining (2nd Edition). Pearson. ISBN: 0133128903.

[3] KDNuggets. (2020). Naive Bayes Algorithm: Everything you need to know. Retrieved from https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html

[4] Turing. (n.d.). An Introduction to Naive Bayes Algorithm for Beginners. Retrieved from https://www.turing.com/kb/an-introduction-to-naive-bayes-algorithm-for-beginners

PRABIN BOHARA is an enthusiastic individual currently pursuing a Bachelor's degree in Computer Engineering, set to graduate in 2024. He has a genuine passion for coding, web technologies, and machine learning. Prabin's primary focus revolves around AI research and exploring the various applications of data science in general. He actively contributes to open-source projects and continuously expands his knowledge through academic endeavors. Prabin actively participates in webinars, both nationally and internationally, as well as competing in diverse technology competitions. He also shares his expertise by conducting workshops on various technology topics. With a strong drive for excellence, Prabin aims to make valuable contributions in the field of AI while keeping up with the latest advancements.

PRABIN SHARMA POUDEL , an undergraduate senior at IOE's Thapathali Campus, has forever been in pursuit of knowledge and wisdom. His involvement extends beyond technical confines, with involvement in non-technical organizations like AYON. With a vision to become a scholarly figure, he aims to bridge multiple disciplines of knowledge. His current field of interest encompasses AI in conflict resolution, Computer vision, and AI in creative industries Beyond academics, Prabin finds joy in playing chess, composing music, and journaling his thoughts through poetry.

SANTOSH PANDEY is a fourth-year Bachelor's student at IOE, Thapathali Campus, with a passion for coding and technology. Currently working as an intern at NAAMII organization, he is driven by the goal of applying his skills to earn a living through coding. With a keen interest in full-stack development, graphic designing, mobile app development, and computer vision, Santosh is constantly exploring new technologies and honing his skills in these areas. In his free time, he actively participates in coding competitions and enjoys participating in marathons. Santosh's dedication, enthusiasm, and continuous learning make him a promising professional in the field of technology and coding.

APPENDIX A: FIGURES AND PLOTS

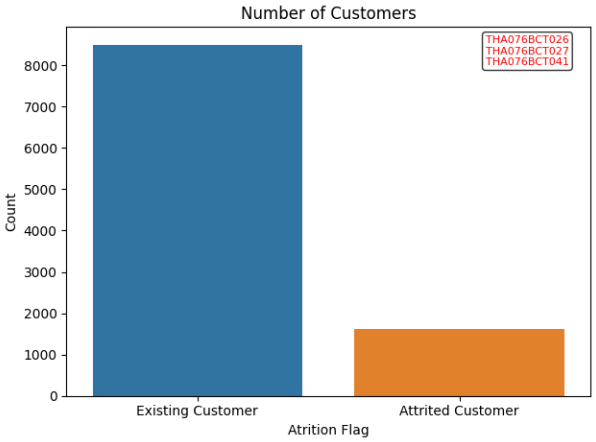

FIGURE 2: Bar Graph of Number of Labels

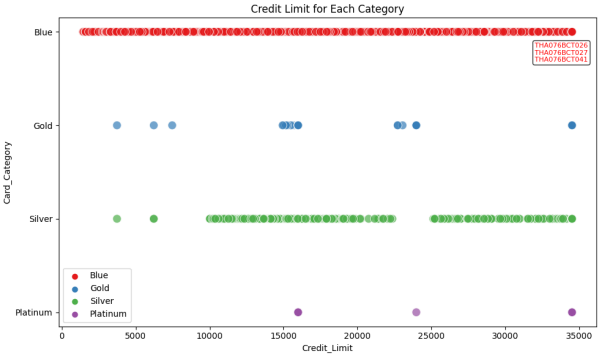

FIGURE 3: Credit Limit for Each Category

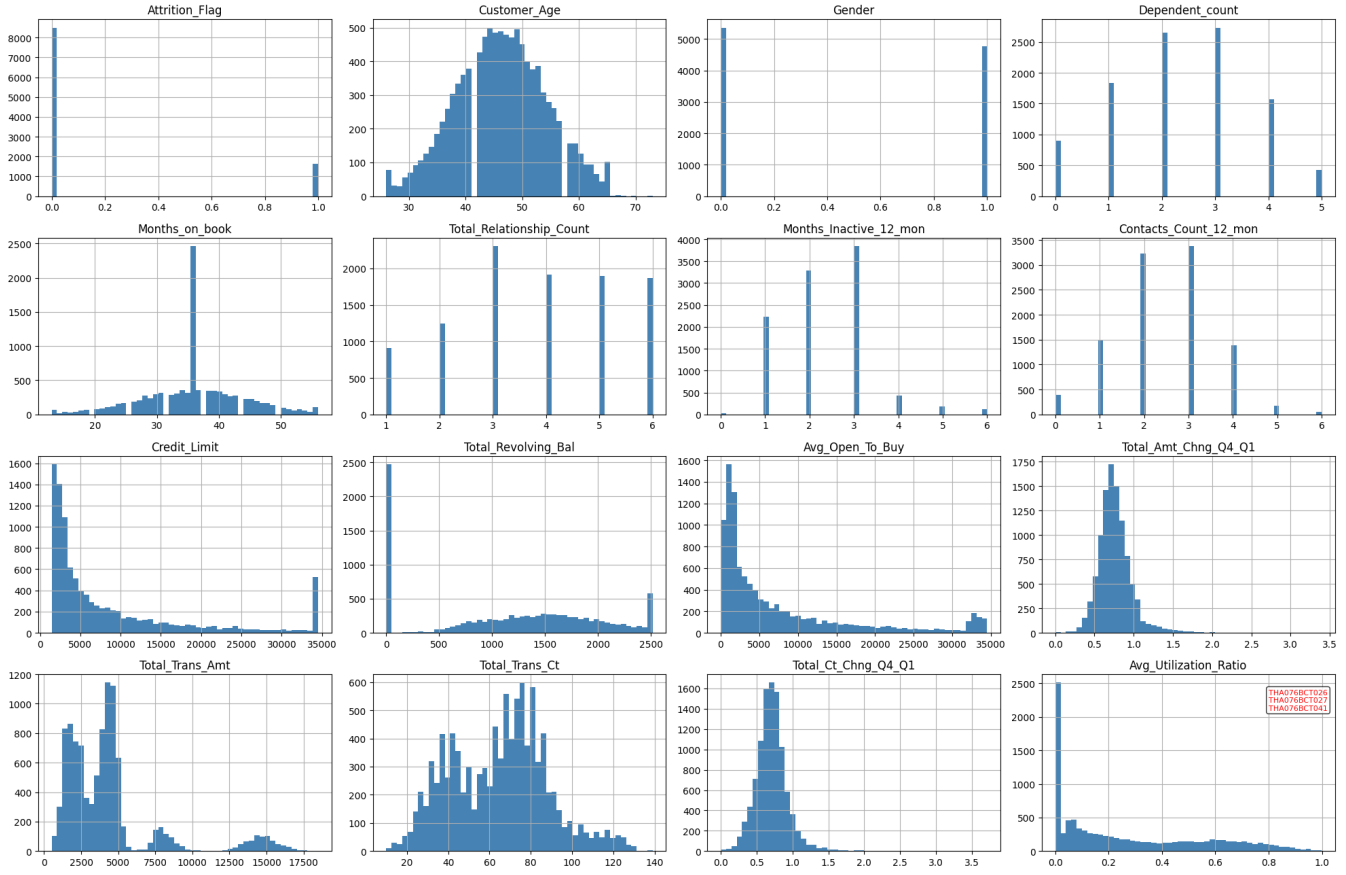

FIGURE 4: Histogram of features

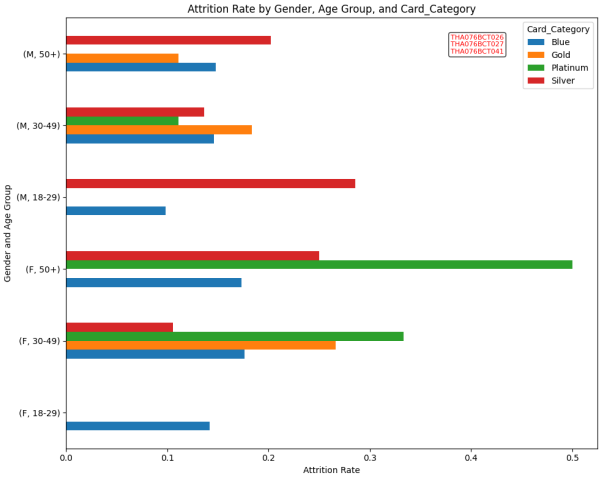

FIGURE 5: Attrition Rate by Gender, Age Group, and Card_Category

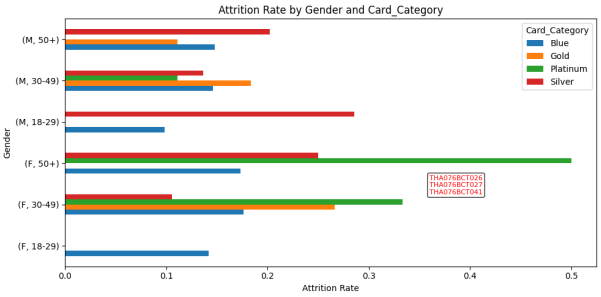

FIGURE 6: Attrition Rate by Gender and Card_Category

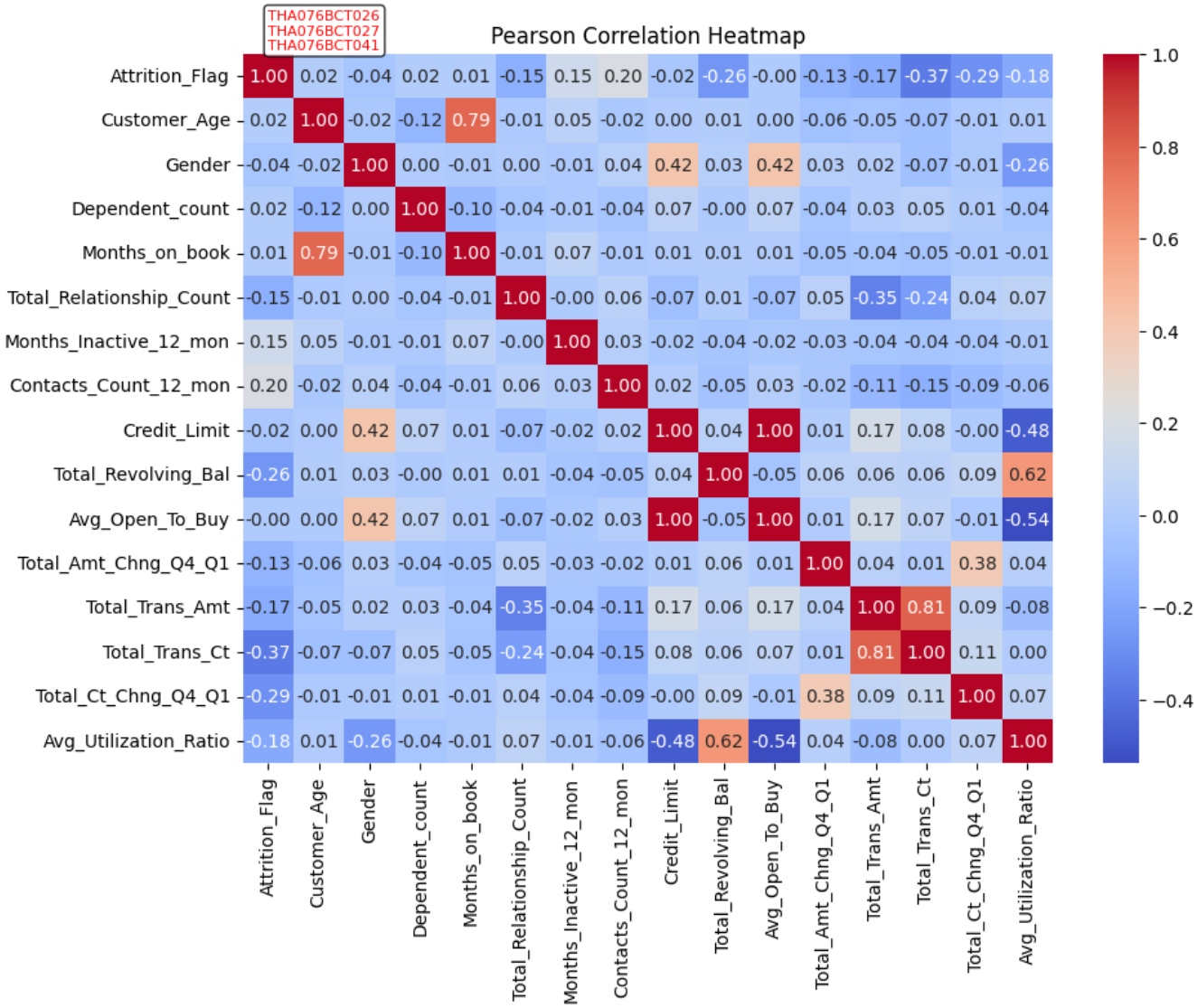

FIGURE 7: Pearson Correlation Heatmap

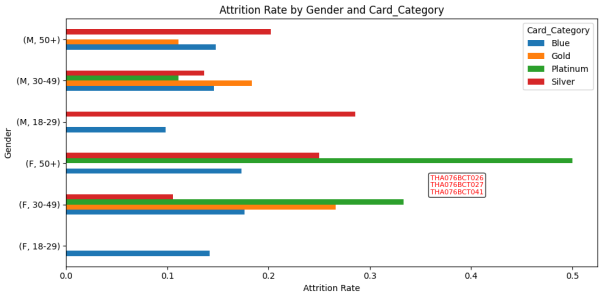

FIGURE 8: Attrition Rate by Gender and Card_Category

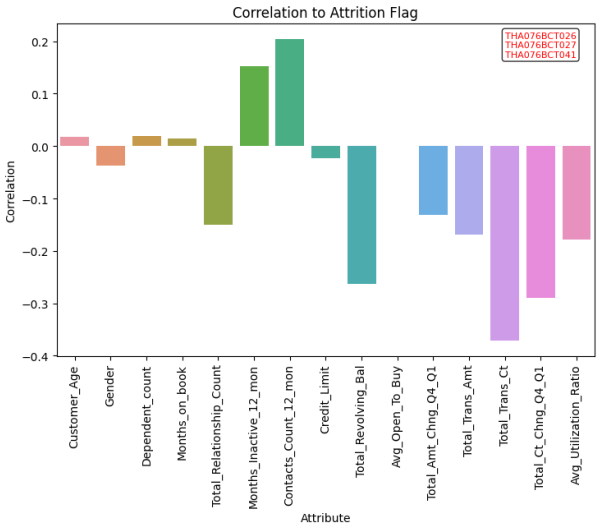

FIGURE 9: Correlation to Attrition Flag

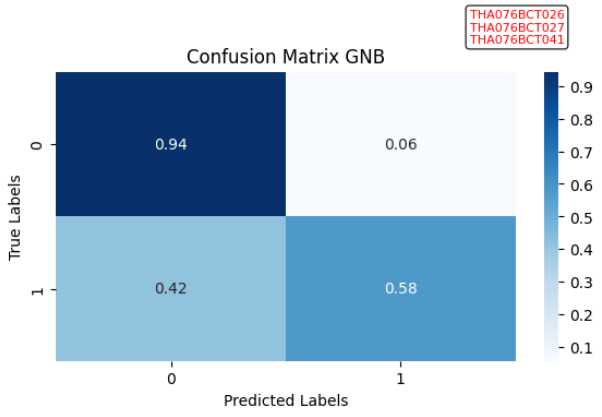

FIGURE 10: Confusion Matrix for Gaussian Naive Bayes

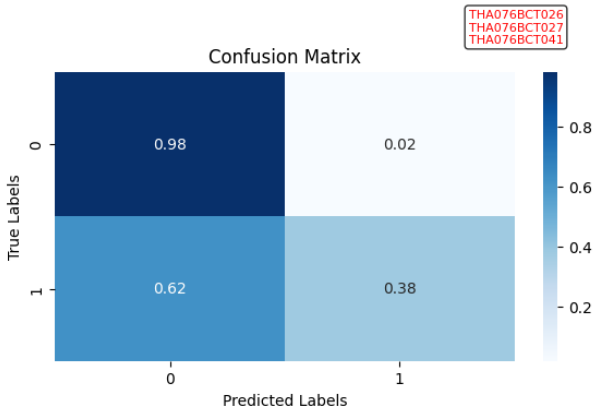

FIGURE 11: Confusion Matrix for Classification Naive Bayes with equal width

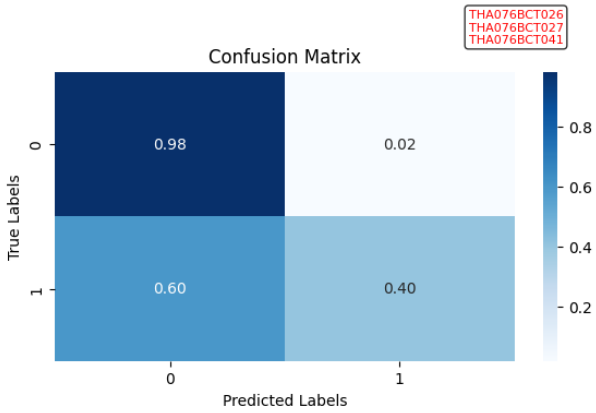

FIGURE 12: Confusion Matrix for Classification Naive Bayes with equal frequency

APPENDIX B: CODE

```
import pandas as pd

# Load the dataset into a DataFrame
df=pd.read_csv('dataset/BankChurners.csv')

df.head(5)

new_column_names = {
    'Naive_Bayes_Classifier_Attrition_Flag_Car': 'mon_2',
    'Naive_Bayes_Classifier_Attrition_Flag_Car': 'mon_1',
}

df.rename(columns=new_column_names, inplace=True)
df.head()

df.drop(columns=['CLIENTNUM','mon_1','mon_2'],inplace=True)

print(df.info())
print(df.head(5))
print(df.isna().sum())
print(df.duplicated().sum())

# Display count of survivors (0: Not Survived, 1: Survived)
label_counts = df['Attrition_Flag'].value_counts()
print(label_counts)

type(label_counts)

import seaborn as sns
import matplotlib.pyplot as plt
# Create the count plot
sns.countplot(x='Attrition_Flag', data=df)

text_box = {
    'boxstyle': 'round',
    'facecolor': 'white',
    'alpha': 0.8
}

plt.text(1.1, 8000, 'THA076BCT026\nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red')
# Set plot labels
plt.xlabel('Atrition Flag')
plt.ylabel('Count')
plt.title('Number of Customers')
plt.tight_layout()
plt.savefig('Number of Label.png', bbox_inches='tight')
# Show the plot
plt.show()

df.replace({'Attrition_Flag':{'Existing Customer':0,'Attrited Customer':1}},
    inplace=True)
# Group the data by 'Sex' and calculate the mean of 'Attrition_Flag' for each group
Atrition_rate_by_gender = df.groupby('Gender')['Attrition_Flag'].mean()

# Print the Atrition rates
print(Atrition_rate_by_gender)

# Create a pivot table to calculate the mean of 'Attrition_Flag' for each combination of 'Sex' and '                _count_E
    Card_Category'
Attrition_rate_pivot = df.pivot_table(values='Attrition_Flag', index='      _count_E
    Gender', columns='Card_Category', aggfunc='mean')

# Print the survival rates
print(Attrition_rate_pivot)

# Plot the survival rates using bar function
Attrition_rate_pivot.plot(kind='barh', figsize=(10, 5))
plt.xlabel('Attrition Rate')
plt.ylabel('Gender')
plt.title('Attrition Rate by Gender and Card_Category')
plt.legend(title='Card_Category')
plt.text(0.36, 1.25, 'THA076BCT026\nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red')
plt.tight_layout()
plt.savefig('Attrition Rate by Gender and Card_Category.png',bbox_inches='tight')
plt.show()

# Discretize the 'Age' column into age groups using the cut function
age_bins = [0, 18, 30, 50, 100]
age_labels = ['0-17', '18-29', '30-49', '50+']
df['Age_Group'] = pd.cut(df['Customer_Age'], bins=age_bins, labels=age_labels)

df.head(5)

# Create a pivot table to calculate the mean of 'Attrition_Flag' for each combination of 'Sex', 'Age_Group', and 'Card_Category'
Attrition_rate_pivot = df.pivot_table(values='Attrition_Flag', index=['Gender', 'Age_Group'], columns='Card_Category', aggfunc='mean')

# Plot the Attrition rates using barh function
Attrition_rate_pivot.plot(kind='barh', figsize=(10, 8))
plt.xlabel('Attrition Rate')
plt.ylabel('Gender and Age Group')
plt.title('Attrition Rate by Gender, Age
```

```
    Group, and Card_Category')
plt.legend(title='Card_Category', loc='
    upper right')
plt.text(0.38, 5, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.savefig('Attrition Rate by Gender,
    Age Group, and Card_Category.png',
    bbox_inches='tight')
plt.show()

# Create a scatter plot for each
    passenger class
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Credit_Limit'
    , y='Card_Category', hue='
    Card_Category', palette='Set1', s
    =100, alpha=0.7)

plt.title('Credit Limit for Each Category
    ')
plt.legend()
plt.text(32000, 0.32, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.savefig('Credit Limit for Each
    Category.png',bbox_inches='tight')

# Show the plot
plt.show()

# Create a new DataFrame without the '
    Class' variable
df1 = df.drop('Card_Category', axis=1)
df1.head(5)
numeric_df = df1.replace({'M': 1, 'F':
    0})
numeric_df.head()

numeric_df.describe()

new_df=numeric_df.drop(columns=['
    Marital_Status','Education_Level','
    Income_Category','Age_Group'])
# Assuming 'df' is the DataFrame
    containing the data
correlation_matrix = new_df.corr(method='
    pearson')

# Create a heatmap using Seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=
    True, cmap='coolwarm', fmt='.2f')
plt.title('Pearson Correlation Heatmap')
plt.text(.5, -0.1, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.savefig('Pearson Correlation Heatmap.
    png',bbox_inches='tight')
plt.show()
correlation_with_target=
    correlation_matrix.iloc[0, 1:]

# Plot Correlation with Target
plt.figure(figsize=(8, 5))
sns.barplot(x=correlation_with_target.
    index, y=correlation_with_target.
    values,)
plt.title('Correlation to Attrition Flag'
    )
plt.xlabel('Attribute')
plt.ylabel('Correlation')
plt.text(12, 0.17, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.xticks(rotation=90)
plt.savefig('Correlation to Attrition
    Flag.png',bbox_inches='tight')
plt.show()

# Get the list of column names
column_names = new_df.columns.tolist()
print(column_names)

new_df.hist(bins=50, figsize=(20, 13),
    color='steelblue')
plt.text(0.8, 2100, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.savefig('Histogram of features.png',
    bbox_inches='tight')
plt.show()


from sklearn.model_selection import
    train_test_split
X= new_df.drop(columns=['Attrition_Flag'
    ])
Y= new_df['Attrition_Flag']
# X_train, X_test, y_train, y_test will
    be the resulting datasets

# Split the dataset into training and
    test sets (80% training and 20% test)
X_train, X_test, y_train, y_test =
    train_test_split(X, Y, test_size=0.2,
     random_state=42)


from sklearn.naive_bayes import
    GaussianNB

# Create an instance of the Gaussian
    Naive Bayes model
gnb = GaussianNB()

from sklearn.impute import SimpleImputer

# Create an imputer object with strategy
    as 'mean'
imputer = SimpleImputer(strategy='mean')

# Fit and transform the imputer on
    X_train
```

```
X_train = imputer.fit_transform(X_train)

# Transform X_test using the same imputer
X_test = imputer.transform(X_test)

# Train the Gaussian Naive Bayes model
gnb.fit(X_train, y_train)

# Make predictions on the test data
y_pred = gnb.predict(X_test)

print(y_pred[:10].tolist())
print(y_test[:10].tolist())
probs = gnb.predict_proba(X_test)
print(probs[:10])

from sklearn.metrics import confusion_matrix

# Assuming gnb is your trained Gaussian Naive Bayes model and X_test, y_test are your test set and corresponding labels
y_pred = gnb.predict(X_test)
confusion_mat = confusion_matrix(y_test, y_pred)
print(confusion_mat)

import numpy as np
# Normalize the confusion matrix
normalized_confusion_mat = confusion_mat.astype('float') / confusion_mat.sum(axis=1)[:, np.newaxis]
# Create a heatmap using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(normalized_confusion_mat, annot=True, fmt=".2f", cmap="Blues")

# Add labels and title
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix GNB")

plt.text(1.8, -0.2, 'THA076BCT026\nTHA076BCT027\nTHA076BCT041', fontsize=8, bbox=text_box, color='red')
plt.tight_layout()
plt.savefig('Confusion Matrix GNB.png', bbox_inches='tight')
plt.show()

from sklearn.metrics import classification_report

report=classification_report(y_test, y_pred)

# Print the classification report
print(report)

from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train3 = scaler.fit_transform(X_train)
X_test3=scaler.transform(X_test)
gnb3 = GaussianNB()
gnb3.fit(X_train3, y_train)
y_pred3 = gnb3.predict(X_test3)

report=classification_report(y_test, y_pred3)
print(report)

print(numeric_df.head())
print(numeric_df.describe())

# Binning settings
age_bins = [0, 11, 21, 31, 41, 51, 61, 71, 81]
age_labels = ['0-10', '11-20', '21-30', '31-40', '41-50', '51-60', '61-70', '71-80']
credit_bins = [0, 1001, 2001, 3001, 6001, 12001, 35001]
credit_labels = ['0-1000', '1001-2000', '2001-3000', '3001-6000', '6001-12000', '12001-35000']

categorical_df = numeric_df.drop(columns=['Customer_Age', 'Credit_Limit'])

# Discretize 'Age' column
categorical_df['Age_Category'] = pd.cut(numeric_df['Customer_Age'], bins=age_bins, labels=age_labels)

# Discretize 'Credit_Limit' column
categorical_df['Credit_Category'] = pd.cut(numeric_df['Credit_Limit'], bins=credit_bins, labels=credit_labels)
categorical_df.drop(columns=['Age_Group'],inplace=True)

categorical_df.head()

categorical_df=new_df.copy()
# Discretize 'Age' column using equal width method with 5 bins
categorical_df['Age_EqualWidth'] = pd.cut(new_df['Customer_Age'], bins=5, labels=False)

# Discretize 'Fare' column using equal width method with 5 bins
categorical_df['Credit_EqualWidth'] = pd.cut(new_df['Credit_Limit'], bins=5, labels=False)
# categorical_df.drop(columns=['Age_Group'],inplace=True)

categorical_df.head()

from sklearn.naive_bayes import CategoricalNB
# Initialize the Categorical Naive Bayes model
cnb = CategoricalNB()
# Access the attributes of the classifier
```

```
    using the get_params() method
classifier_attributes = cnb.get_params()

# Display the attributes
print(classifier_attributes)
# Drop the rows containing 'UNKNOWN' in any column
# categorical_df =categorical_df = categorical_df.drop(index=categorical_df[categorical_df.eq('UNKNOWN').any(axis=1)].index)
X1= categorical_df.drop(columns=['Attrition_Flag'])
Y1= categorical_df['Attrition_Flag']

# Split the dataset into training and test sets (80% training and 20% test)
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, Y1, test_size=0.2, random_state=42)


# In[49]:


X_train1.isnull().sum()


# In[50]:


## Create an imputer object with strategy as 'mean'
imputer = SimpleImputer(strategy='mean')

# Fit and transform the imputer on X_train
# X_train1 = imputer.fit_transform(X_train1)
print(type(X_test1))
# Transform X_test using the same imputer
# X_test1 = imputer.transform(X_test1)
print(type(X_test1))
# Train the model on the training data
print(X_train1.shape,X_test1.shape)
classifier_attributes = cnb.get_params()

# Display the attributes
print(classifier_attributes)
# Make predictions on the test data
# Train the model on the training data
cnb.fit(X_train1, y_train1)
length=len(X_test1)
print(length)
# Make predictions on the test data
(pd.concat([X_test1[:216], X_test1[217:]])).shape
new_test = pd.concat([X_test1[:216], X_test1[217:827], X_test1[828:1934], X_test1[1935:]])
new_y_test = pd.concat([y_test1[:216], y_test1[217:827], y_test1[828:1934], y_test1[1935:]])

print(len(X_test1))
y_pred1 = cnb.predict(new_test)

print(f"Accuracy Score: {accuracy_score(y_true = new_y_test, y_pred = y_pred1) * 100:.2f} %")

confusion_mat = confusion_matrix(new_y_test, y_pred1)
print(confusion_mat)

# Normalize the confusion matrix
normalized_confusion_mat = confusion_mat.astype('float') / confusion_mat.sum(axis=1)[:, np.newaxis]
# Create a heatmap using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(normalized_confusion_mat, annot=True, fmt=".2f", cmap="Blues")

# Add labels and title
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.text(1.8, -0.2, 'THA076BCT026\nTHA076BCT027\nTHA076BCT041', fontsize=8, bbox=text_box, color='red')
plt.tight_layout()
plt.savefig('Confusion Matrix CNB_eql_width.png',bbox_inches='tight')

plt.show()
report=classification_report(new_y_test, y_pred1)

# Print the classification report
print(report)
categorical_df1=new_df.copy()

categorical_df1['Age_EqualFrequency'] = pd.qcut(numeric_df['Customer_Age'], q=5, labels=False)

# Discretize 'Age' column using equal frequency method with 5 bins
categorical_df1['Credit_EqualFrequency'] = pd.qcut(numeric_df['Credit_Limit'], q=5, labels=False)
categorical_df1.head()

X2= categorical_df1.drop(columns=['Attrition_Flag'])
Y2= categorical_df1['Attrition_Flag']
# Split the dataset into training and test sets (80% training and 20% test)
X_train2, X_test2, y_train2, y_test2 = train_test_split(X2, Y2, test_size=0.2, random_state=42)
# Fit and transform the imputer on X_train
X_train2 = imputer.fit_transform(X_train2)

# Transform X_test using the same imputer
X_test2 = imputer.transform(X_test2)
# Train the model on the training data
X_test2= np.concatenate([X_test2[:216], X_test2[217:827], X_test2[828:1934],
```

```
    X_test2[1935:]])
y_test2= np.concatenate([y_test2[:216],
    y_test2[217:827], y_test2[828:1934],
    y_test2[1935:]])
cnb.fit(X_train2, y_train2)

# Make predictions on the test data
y_pred2 = cnb.predict(X_test2)
print(y_pred2.shape)
print(y_test2.shape)
print(X_test2.shape)

confusion_mat = confusion_matrix(y_test2,
     y_pred2)
print(confusion_mat)

# Normalize the confusion matrix
normalized_confusion_mat = confusion_mat.
    astype('float') / confusion_mat.sum(
    axis=1)[:, np.newaxis]
# Create a heatmap using seaborn
plt.figure(figsize=(6, 4))
sns.heatmap(normalized_confusion_mat,
    annot=True, fmt=".2f", cmap="Blues")

# Add labels and title
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix")
plt.text(1.8, -0.2, 'THA076BCT026\
    nTHA076BCT027\nTHA076BCT041',
    fontsize=8, bbox=text_box, color='red
    ')
plt.tight_layout()
plt.savefig('Confusion Matrix
    GNB_eql_freq.png',bbox_inches='tight'
    )

plt.show()
report=classification_report(y_test2,
    y_pred2)

# Print the classification report
print(report)
```