An Empirical Study on Naive Bayes Classifier (January 2024)

Bishwambhar Dahal[1] and Sirjana Bhatta[1]

[1]Department of Electronics and Computer Engineering, IOE, Thapathali Campus, Kathmandu 44600, Nepal
Corresponding author: Bishwambhar Dahal(dahalbishwambhar@gmail.com) and Sirjana Bhatta(sirjanabhatta6@gmail.com)

Abstract This empirical study presents a comprehensive investigation into the performance of the Naive Bayes algorithm for classification tasks. Known for its simplicity and efficiency, the Naive Bayes algorithm is a popular probabilistic machine learning technique. The primary objective of this research is to assess the effectiveness of Naive Bayes in classsification task and gain practical insights into its applicability. To achieve this objective, we conduct the study on the widely used Pokemon dataset from the kaggle. We evaluate the performance of Gaussian Naive Bayes, Categorical Naive Bayes and Hybrid Naive Bayes classifiers, exploring their ability to handle diverse data types. To gain a deeper understanding of the dataset and the algorithm's behavior, we utilize a range of data visualization techniques, including graphs and tables, to analyze the features and their distributions. Moreover, we investigate the impact of different preprocessing techniques, such as Label encoding and discretization, on the algorithm's accuracy. The results of this study contribute to a better understanding of Naive Bayes' performance, shedding light on its strengths and limitations in various contexts.

Keywords *Naive Bayes, Classification, Machine Learning, Probabilistic.*

I Introduction

In recent years, there has been a growing interest in the field of classification algorithms, and one particular approach that has garnered significant attention is the Naive Bayes algorithm [1]. With its simplicity and effectiveness in various applications, Naive Bayes has become a popular choice for tackling classification tasks. The Naive Bayes algorithm is a simple yet powerful probabilistic machine learning [2] technique used for classification tasks. It is supervised learning algorithm based on Bayes' theorem and is particularly well-suited for problems involving classification tasks.

While other classification algorithms like Decision Trees, Support Vector Machines (SVM), Logistic Regression, and Neural Networks have gained prominence, Naive Bayes algorithm offers a unique probabilistic perspective on the classification task. Unlike many other traditional algorithms, Naive Bayes relies on an assumption of feature independence, making it computationally lightweight and easy to implement. This work delves into the principles of Naive Bayes and evaluates its accuracy in classification on Pokemon dataset available from kaggle.
In this work, we implement label encoding for class labels since it is necessary for Naive Bayes to convert categorical class labels into numerical representations, allowing the algorithm to process the target variable effectively and make accurate predictions in classification tasks.

In this report, our primary objective is to explore the principles of the Naive Bayes algorithm and evaluate its accuracy in classification tasks using the widely used Pokemon dataset obtained from the kaggle. The Pokemon dataset contains various attributes of Pokemon creatures, and the classification task involves predicting their respective types.

In this report, we explore the principles of Naive Bayes and explore its three prominent variants: Gaussian Naive Bayes [3], Categorical Naive Bayes and Hybrid Naive Bayes. We compare the strengths and limitations of these Naive Bayes variants and examine their performance in different scenarios. By understanding the nuances of each variant, we aim to provide insights into their applicability and potential benefits across a range of real-world classification problems. We trained both variants on Pokemon dataset and evaluated their accuracy in classification.

We delve into the mathematical foundations of Bayes' theorem, explain the naive assumption, and demonstrate how the algorithm is applied in practice. Ultimately, this

report aims to provide a comprehensive understanding of the Naive Bayes algorithm's strengths and weaknesses for classification tasks.

II Methodology

A Theory

In classification tasks, the goal is to assign a label or category to an input instance based on its features. Given a set of training examples with known labels, Naive Bayes learns the conditional probability of each feature given the class label. It then uses these probabilities to predict the most likely class label for new, unseen instances.

The algorithm's simplicity and effectiveness stem from the assumption of conditional independence among the features given the class label, which is why it is called "naive." Despite this seemingly unrealistic assumption, Naive Bayes has proven to be remarkably successful in many real-world applications. The term "naive" in the context of the Naive Bayes algorithm refers to a key simplifying assumption made by the algorithm during the classification process. It assumes that all features in the dataset are conditionally independent of each other given the class label.

In probability theory and statistics, two events A and B are considered independent if the occurrence of one event does not affect the probability of the other event happening. In the context of Naive Bayes, this assumption means that the presence or absence of a particular feature does not depend on the presence or absence of any other feature, given the class label. In practice, this assumption greatly simplifies the calculation of probabilities, as it allows the joint probability of multiple features to be expressed as a product of individual probabilities. By making this assumption, Naive Bayes can efficiently estimate the probability of a particular class given the observed features, even in cases where the dataset has a large number of features.

However, it is essential to acknowledge that the assumption of feature independence may not hold in all real-world datasets. In many cases, features may be correlated or dependent on each other, and Naive Bayes can be limited in handling such complex relationships. Despite this simplification, Naive Bayes has been found to have high efficiency and good performance in various applications.

Naive Bayes is a family of algorithms that are based on the Bayes theorem and make the "naive" assumption of conditional independence among the features given the class label. There are several variants of the Naive Bayes algorithm, two of which are the Gaussian Naive Bayes and Categorical Naive Bayes. Gaussian Naive Bayes is well-suited for continuous numeric features, assuming that each class's features follow a Gaussian (normal) distribution. If all the features are not Gaussian, Gaussian Naive Bayes may not be the best choice for the classification task. Since Gaussian Naive Bayes assumes that all features have a Gaussian (normal) distribution within each class, it may lead to suboptimal performance and inaccurate predictions when this assumption does not hold. On the other hand, Categorical Naive Bayesr, also known as Multinomial Naive Bayes, is tailored for discrete features, such as word counts in text data or binary variables, making it ideal for text classification tasks. If features are categorical (discrete) variables, you can use Categorical Naive Bayes. This variant of Naive Bayes is suitable for dealing with features that have discrete outcomes and do not follow a continuous distribution.

Binning continuous values is a process of dividing a continuous variable into a set of discrete intervals or bins. This can be helpful when we need to transform a continuous variable into a categorical one. If we want to apply Categorical Naive Bayes to a dataset with continuous features, we can convert those continuous values to discrete values using binning techniques. Since Categorical Naive Bayes only works with discrete (categorical) features, converting the continuous features into discrete bins allows you to utilize the Categorical Naive Bayes algorithm.

Equal-width binning and equal-frequency binning are two techniques used to convert continuous data into discrete intervals or bins. In equal-width binning, the data range is divided into a fixed number of bins, with each bin covering an equal range of values . This method is easy to implement but may lead to empty bins or be sensitive to outliers. On the other hand, equal-frequency binning divides the data into bins containing an equal number of data points, determined by quantiles. This approach is useful for handling skewed distributions and maintaining an even distribution of data across bins. It is less affected by outliers compared to equal-width binning. The choice between the two methods depends on the data distribution and specific analysis requirements. Both techniques can be employed to transform continuous features into categorical ones, enabling the use of algorithms like categorical Naive Bayes for classification tasks.

An essential advantage of the Hybrid Naive Bayes classifier is its ability to handle diverse datasets without the need for discretization of continuous features. By natively managing both feature types, it simplifies the

preprocessing steps and retains the original characteristics of the data.

In conclusion, the Hybrid Naive Bayes classifier is a powerful supervised machine learning algorithm for classification tasks that involve datasets with mixed attribute types. By combining the principles of Categorical Naive Bayes and Gaussian Naive Bayes, it offers an efficient and effective solution, making it a popular choice in real-world applications where data contains a combination of continuous and categorical features.

The Hybrid Naive Bayes classifier is advantageous when dealing with datasets that contain a mix of categorical and continuous features. By leveraging the strengths of both Categorical and Gaussian Naive Bayes, it offers a versatile and effective solution for classification tasks in real-world applications.

In conclusion, the Naive Bayes algorithm, including its variants like Gaussian Naive Bayes, Categorical Naive Bayes, and Hybrid Naive Bayes, is a widely used supervised machine learning algorithm for classification tasks. It is known for its simplicity, computational efficiency, and ability to scale well with large datasets. Despite the "naive" assumption of feature independence, Naive Bayes often performs remarkably well, making it a popular choice in various domains, particularly in text classification and natural language processing applications.

B Mathematical Formulae

The Naive Bayes algorithm makes the naive assumption of conditional independence among features given the class label. Formally, for a set of features X1, X2, ..., Xn, and a class label C, the conditional independence assumption can be expressed as: For a set of features X1, X2, ..., Xn, and a class label C:

$$P(X_1, X_2, \ldots, X_n \mid C) = P(X_1 \mid C) \times P(X_2 \mid C) \times \cdots \times P(X_n \mid C) \quad (1)$$

$$P(A|B) = \frac{P(B \cap A)}{P(B)} \quad (2)$$

Given an input instance with features X, the Naive Bayes algorithm predicts the class that maximizes the posterior probability given the observed features X.
For a binary classification problem with two classes, "positive" (P) and "negative" (N):

$$P(P|X) = \frac{P(X|P) \cdot P(P)}{P(X)} \quad (3)$$

$$P(N|X) = \frac{P(X|N) \cdot P(N)}{P(X)} \quad (4)$$

where,

- $P(P \mid X)$ is the posterior probability of class "positive" given the observed features X.
- $P(N \mid X)$ is the posterior probability of class "negative" given the observed features X.
- $P(X \mid P)$ is the likelihood of observing the features X given the class "positive."
- $P(X \mid N)$ is the likelihood of observing the features X given the class "negative."
- $P(P)$ and $P(N)$ are the prior probabilities of classes "positive" and "negative," respectively.
- $P(X)$ is the probability of observing the features X (a normalization factor).

To classify a new instance with features X_new, we can use the class with the highest posterior probability as the predicted class:

$$\text{Predicted Class } = \arg\max(P(C|X_{\text{new}})) \quad (5)$$

where, where argmax denotes the class with the highest probability.

We consider each feature x_i independently given the class C. For a continuous feature, the likelihood $P(x_i|C)$ is modeled as a Gaussian (normal) distribution with mean μ_i and standard deviation σ_i. The probability density function (PDF) of the Gaussian distribution is given by:

$$f(x_i|\mu_i, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} \cdot e^{-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}} \quad (6)$$

where,

- μ_i is the mean of feature x_i in class C.
- σ_i is the standard deviation of feature x_i in class C.

For a categorical feature, the likelihood $P(x_i|C)$ is modeled as a multinomial distribution with parameter $\theta_{i,C}$, which represents the probability of observing category 'i' in class 'C'. The formula for the likelihood $P(x_i|C)$ in Categorical Naive Bayes is:

$$P(x_i|C) = \theta_{i,C} \quad (7)$$

Where $\theta_{i,C}$ is the probability of observing category 'i' in class 'C'. These probabilities are estimated from the training data for each class.

The formula for calculating the probability of Hybrid classifier is given as:

$$\text{hybrid\_probs} = \text{wt\_cl1} \times \text{cl1\_probs} + \text{wt\_cl2} \times \text{cl2\_probs} + \ldots + \text{wt\_clN} \times \text{clN\_probs} \quad (8)$$

where,

- hybrid_probs: This represents the hybrid probabilities that are obtained by combining the probabilities from multiple classifiers.
- wt_classifier1,wt_classifier2,. . . ,wt_classifierN: These are the weights assigned to each classifier's probabilities in the ensemble.
- classifier1_probs,classifier2_probs,. . . ,classifierN_probs: These are the probabilities generated by each individual classifier in the ensemble for each instance in the dataset.

The formula for accuracy is used to evaluate the performance of a classification model, including k-Nearest Neighbors (KNN). The formula for accuracy is as follows:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (9)$$

Mathematically, if TP represents the number of true positives (correctly predicted positive instances), TN represents the number of true negatives (correctly predicted negative instances), FP represents the number of false positives (incorrectly predicted positive instances), and FN represents the number of false negatives (incorrectly predicted negative instances) [4], then the formula for accuracy, recall and precision can be expressed as:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (11)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

Also, the formula for f1-score is as follows:

$$\text{F1 Score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (13)$$

C System Block Diagram

The figure 1 presents the block diagrams for both the Gaussian Naive Bayes classifier and the Categorical Naive Bayes classifier. On the other hand, figure 2 showcases the block diagram specifically for the Hybrid Naive Bayes classifier. In the block diagrams, each classifier's architecture and workflow can be seen.

D Working Principle

The Naive Bayes algorithm is a probabilistic classification algorithm based on Bayes' theorem. It is considered "naive" because it makes a strong assumption of feature independence, which simplifies the calculations and allows it to scale well with large datasets. Despite this assumption, Naive Bayes often performs surprisingly well in many real-world applications, particularly in text classification tasks.

Naive Bayes algorithm is widely used algorithm for classification task. It is supervised machine learning algorithm. We have implemented this algorithm using python programming language. We first load pokemon dataset which was availabe on kaggle. The dataset contains 800 records having 13 features for each record. The whole dataset is divided into 2 classes.
Intially, we gather and preprocess the labeled dataset which consists features and corresponding class labels. The class labels 'True' and 'False' are label encoded to 0 and 1. We visualized the features using various graphs and tables.

We first implement Gaussian Naive Bayes Classifier. To build the Gaussian Naive Bayes model, we need to estimate the parameters of the Gaussian distribution for each feature in each class. This involves calculating the mean and standard deviation of the feature values for each class from the training data. The parameters are estimated using maximum likelihood estimation, which helps us make the best possible estimates based on the available data.

The classification process in Gaussian Naive Bayes involves calculating the posterior probability of each class given the input features. This is done using Bayes' theorem, where the likelihood of the features given the class and the prior probability of each class are combined to obtain the posterior probability. The class with the highest posterior probability is then assigned as the predicted class for the input. For this, we fit the Gaussian classifier on our training set which includes 70% of total data. Gaussian Naive Bayes assumes that all features are distributed normally. This assumption simplifies the calculations and allows us to estimate the parameters of the Gaussian distribution (mean and standard deviation) for each feature in each class. We predict the result using the trained Classifier on test set which is the remaining 30% of the data.

Classification report calculate different parameters like accuracy, precision, recall, f1 score, etc. to evaluate model performance. We draw classification report using

the Predicted values and the ground truth. We also draw the confusion matrix to visualize the accuracy of model which is a fundamental tool for evaluating the performance of a supervised machine learning algorithm, especially in classification tasks.

Furthermore we implement Categorical Naive Bayes classifier which is a probabilistic approach to solve classification tasks with categorical features. To apply Categorical Naive Bayes we converted the continuos features into discere features using equal width binning and equal frequency binning. The Categorical classifier is trained on the discrete features and the trained model is used to predict on test set. The classification report is then generated and the confusion matrix is drawn.

As our dataset consists of both continuous and discrete data, Hybrid Naive Bayes is the most suitable approach for this kind of data. For best result we implement Hybrid Naive Bayes on our dataset. During model training, two distinct processes are applied to handle these different feature types. We implement the Hybrid Naive Bayes classifier by combining the principles of both Categorical Naive Bayes and Gaussian Naive Bayes. For categorical features Generation, En_Type 1, we follow the same process as in Categorical Naive Bayes, estimating the prior probabilities and likelihoods. For continuous features HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, the Hybrid Naive Bayes classifier models them using Gaussian distributions, estimating the mean and standard deviation for each continuous feature in each class, akin to Gaussian Naive Bayes.

In the classification phase, the Hybrid Naive Bayes classifier calculates the conditional probability for each class using Bayes' theorem, combining probabilities from both categorical and continuous features. This allows the classifier to effectively leverage the strengths of both approaches when predicting the class label for a new instance.

To evaluate the performance of the Hybrid Naive Bayes classifier, predictions are made on the test set, and a comprehensive classification report is generated. This report includes various metrics, such as accuracy, precision, recall, and F1-score, providing valuable insights into the model's performance. Additionally, a confusion matrix is drawn to visualize the accuracy of the classifier in predicting the class labels.

During the classification phase, the Hybrid Naive Bayes classifier calculates the conditional probability for each class using Bayes' theorem, combining probabilities from both categorical and continuous features. To evaluate the performance of the Hybrid Naive Bayes classifier, we predict on the test set and generate a classification report, which includes various metrics such as accuracy, precision, recall, and F1-score. Additionally, we draw the confusion matrix to visualize the accuracy of the model in predicting the class labels.

E Instrumentation Details

In our study, we utilized the Python programming language to implement the NaiveBayes algorithm. The implementation was done in Jupyter Notebook, which is an interactive computing notebook environment. We imported various libraries for our study, with the most important one being the scikit-learn library [5]. To read the dataset present in csv file for our machine learning task, we used the `pd.read_csv` function from the pandas library.

In our data analysis workflow, we employ a series of well-defined steps to gain valuable insights from the dataset. Initially, we utilize the powerful pandas library to drop less significant columns, namely 'Type 2','Total' and 'Name' using the `drop()` function. Subsequently, we leverage the `groupby().size()` function to investigate the size of different groups within a specific column. To visualize the frequency distribution of data in a column, we turn to the seaborn library's `countplot()` function, which provides clear and informative visual representations. Additionally, we harness the `heatmap()` function from seaborn to create a visually appealing correlation representation of attributes. For histogram visualization, the `hist()` function proves valuable. To gain further insights, we once again rely on pandas, utilizing the `pivot()` function to construct a comprehensive pivot table, enabling us to analyze relationships between various attribute sets. Moreover, we effectively segment data within the 'Total' attribute by implementing the `cut()` function, resulting in two distinct bins: 180 to 600 and 600 to 780. We take advantage of the matplotlib.pyplot library to generate a variety of visually engaging graphs, such as horizontal pivot graphs using `plot.barh()` function and scatter plots using `scatter()` function. We have also used `label_encoder.fit_transform()` function of sklearn.preprocessing module to encode our textual catagorical data.

In the next step of our process, we employ the `shuffle()` function to randomly shuffle our dataset, ensuring a diverse and unbiased representation of the data. To split the dataset into training and test sets, we utilize the `train_test_split()` function from the `sklearn.model_selection` module, which takes

parameters like data, target, test_size, and random_state. To maintain a balanced distribution of less frequent class data, we include the stratify parameter. Using this function we split our dataset into 70% of train and 30% of test set. For evaluating the performance of our model, we utilized the `confusion_matrix()` function from the sklearn.metrics module to construct a confusion matrix. Furthermore, we generated a classification report using the `classification_report()` function providing valuable insights into the model's precision, recall, F1 score, and support for each class.In order to visualize the confusion matrix, we used the `heatmap()` function from the seaborn library.

To create an instance of the Naive Bayes classifier, we used the `GaussianNB()` function from the sklearn.naive_bayes. For the training and prediction tasks, we utilized the `.fit()` and `.predict()` functions of the `.naive_bayes()` class, respectively. The. `fit()` function takes the training data and the ground truth values, while the `.predict()` function takes the test data and returns the predicted class values.

We also use categorical Naive Bayes classifier. For this purpose deiscretize features having continuous value into bins of equal width and equal frequency. For this purpose we use `cut()` and `qcut()` functions respectively. We create instance of classifier using `CategoricalNB()` function from the sklearn.naive_bayes. For the training and prediction tasks, we utilized the `.fit()` and `.predict()` functions of the `.naive_bayes()` class, respectively. The. `fit()` function takes the training data and the ground truth values, while the `.predict()` function takes the test data and returns the predicted class values.

In last step, we perform Hybrid Naive Bayes. For this purpose `GaussianNB()` function is used for continuous features and `CategoricalNB()` function is used for categorical features. For the training, we utilized the `.fit()` function and for predicting probabilities, we utilize `.predict_proba` functions of the `.naive_bayes()` class. Class with highest probability is chosen using `.argmax()` function of numpy library.

Overall, these libraries and functions were employed to implement and evaluate the Naive Bayes algorithm, as well as to visualize the results in a clear and informative manner.

F Dataset Description

In our project, we utilized the Pokemon dataset, a popular and widely used dataset for classification tasks. This comprehensive dataset contains a diverse range of attributes for numerous Pokemon species. Each row represents a unique Pokemon, and the dataset consists of 800 instances featuring 721 distinct Pokemon species.

The dataset columns encompass essential information, including the Pokemon's name, primary and secondary types (Type 1 and Type 2), the total sum of its base stats (Base Total), and specific stats such as HP (Hit Points), Attack, Defense, Special Attack (Sp. Atk), Special Defense (Sp. Def), and Speed. Moreover, the dataset incorporates a crucial feature known as "Legendary," which indicates whether the given Pokemon is legendary or not. This feature serves as the target for prediction in our classification tasks.

Notably, the dataset presents an inherent class imbalance, with 735 instances corresponding to non-legendary Pokemon and only 65 instances representing legendary Pokemon. This unbalanced nature poses a challenge in developing accurate classification models, which we addressed through careful data preprocessing and model evaluation techniques to ensure reliable predictions.

Throughout our project, we leveraged this comprehensive Pokemon dataset to build and evaluate classification models aimed at predicting the legendary status of Pokemon based on their attributes. The insights gained from this dataset shed light on the distinguishing characteristics of legendary and non-legendary Pokemon, contributing to a deeper understanding of the fascinating Pokemon universe.

III EXPERIMENTAL RESULTS

From fig 3, we can observe a notable class imbalance between the true class and false class in the dataset. The true class appeared to be much less frequent compared to the false class.

Fig 4 shows the correlation between various features present in the dataset.

In fig 5 we present the probabilities of encountering Legendary Pokemon in different generations of the Pokemon series. The table displays the likelihood of encountering a Legendary Pokemon in each generation, expressed as a percentage.

In fig 6 we can see the pivot table between Generation, Attack and Legendary. The values in the table 6 represent the probabilities of encountering Legendary Pokemon within each combination of generation and attack category. These probabilities are given in percentage format.

Fig 7 displays the frequency of different generations of pokemons with their attack capability.

We also draw bar plot for the pivot table of fig 5 in fig 8 and for the pivot table of fig 6 in fig 9.

We also draw histogram with a kernel density plot for each features in our dataset. Fig 10, 11, 12, 13, 14 and 15 shows the Gaussian distribution of the continuous features 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def' and 'Speed' respectively. We can see that for 'Generation' and 'En_Type 1' histogram with a kernel density plot don't follow Gaussian distribution as shown in figures 16 and 17

Fig 18, 19, 20 and 21 are the confusion matrices of Gaussian Naive Bayes, Categorical Naive Bayes for equal width binning, Categorical Naive Bayes for equal frequency binning and Hybrid Naive Bayes respectively.

Similarly 22, 23, 24, 25 are their classification reports respectively. We found that the accuracy of Gaussian Naive Bayes is found to be 92.91%, accuracy of Categorical Naive Bayes with equal width and equal frequency are found to be 94.58% and 93.75% respectively and the accuracy of Hybrid Naive Bayes is found to be 95.00%. The Hybrid Naive Bayes achieved the highest accuracy outperforming the other two variants.

IV Discussion and Analysis

From data analysis, we observed the extreme class imbalance between the two classes. This imbalance can potentially lead to biased predictions and affect the overall performance of the classification model. The stratify parameter used plays a crucial role in mitigating the impact of class imbalance during the data splitting process, such as train-test split. By setting the stratify parameter to the target class variable, we ensure that the class distribution remains consistent in both the training and test datasets. This parameter helps prevent biased training, enables better generalization, and provides a more accurate assessment of the model's performance across all classes. Without stratification, the accuracy metric is found to be misleading, leading to potential under performance on minority class instances in the test set. If our dataset was balanced then the classifier surely would have performed better.

We also examined the correlation between different features in the dataset to gain insights into their relationships. The maximum correlation value observed was 0.51, indicating a moderate positive correlation between the two corresponding features. Identifying correlated features is important task for feature selection and dimensionality reduction. High correlations between features may suggest redundant information or multicollinearity, where one feature may be linearly predictable from another. Here, correlations between features are moderate and do not raise significant concerns about redundancy or multicollinearity.

According to the data from table 5, the probabilities vary across different generations. In Generation 1, the probability of encountering a Legendary Pokemon is approximately 3.61%, while in Generation 2, it increases slightly to about 4.72%. The likelihood further rises in Generation 3, reaching approximately 11.25%. In Generations 4 and 5, the probabilities are quite close, at around 10.74% and 9.09%, respectively. Finally, in Generation 6, the probability of encountering a Legendary Pokemon is approximately 9.76%. The trend suggests that there is a general increasing trend in the probability of encountering Legendary Pokemon as the generations progress.

In pivot table of fig 6, the rows represent the generation categories (gen_cat), which indicate the range of generations in which the Pokemon appeared. For example, "1 to 3" represents the first three generations, and "4 to 6" includes the fourth to the sixth generations. The columns represent the attack categories (attack_cat), which group Pokemon based on their attack strength. For instance, "5 to 50" represents Pokemon with attack strength between 5 and 50, "51 to 100" includes those with attack strength between 51 and 100, and so on.

Fig 10, 11, 12, 13, 14, 15, 16 and 17, shows the histogram with a kernel density plot. The continuous features 'HP', 'Attack', 'Defense', 'Sp. Atk', 'Sp. Def' and 'Speed' do not precisely adhere to a perfect Gaussian distribution, however, they exhibit a remarkably similar shape to the Gaussian curve. 'Generation' and 'En_Type 1' features of our dataset are categorical due to which they don't possess Gaussian distribution in histogram with a kernel density plot figures which can be seen in figures 16 and 17.

When applying Categorical Naive Bayes, we implement binning for discretization of continuous features. Equal width binning and equal frequency binning [6] are the two approaches used. During equal width binning, we sometimes encounter NaN values after binning. This is because it is possible to encounter sparse regions in the data which results in some bins containing no data points and leading to empty bins. To prevent this situation, we filled the NaN values with the mode of the respective feature. By filling the NaN values with the mode, we

effectively substituted the missing values with the most frequently occurring value in that feature. This approach is particularly useful for handling sparse regions since the mode represents the most common value and helps maintain the overall distribution of the data.

We found that the accuracy of Categorical Naive Bayes is more than that of Gaussian Naive Bayes even we have more continuous features in the dataset. This is because of the nature of the data and the assumptions made by the classifier. Gaussian Naive Bayes is specifically designed to handle continuous numerical data that follows a Gaussian (normal) distribution. But in our dataset there are some categorical features as well and also the continuous features don't follow perfect gaussian distribution due to which it's performance is lower compared to Categorical Naive Bayes. This result indicates that the assumptions made by Categorical Naive Bayes were more suitable for our dataset.

Since the data distribution in our dataset is not uniform and has varying data density across different ranges, equal width binning is seen to perform better at capturing the underlying patterns and thus its accuracy is higher than classifier with equal frequency binning.

From the classification reports in figures 22, 23, 24 and 25, we can see that the precision, recall and f1-score for class0 (False classes) are higher than that of class1(True classes). This is due to the class imbalance nature of our dataset. But also the weighted precision, weighted recall and weighted f1-score indicate that the overall performance of the model is quite good. The higher performance metrics for class0 can be attributed to the fact that it has more samples compared to class1, resulting in the model being more biased towards class0 during training.

V Conclusion

In this lab, we successfully performed classification on a 'Pokemon' dataset using various varients of Naive Bayes Classifier. By using various libraries and functions of python, we removed some of the unrequired features and label encoded some features to implement Naive Bayes. We also preprocessed the features as per the requirement of the classifier varient. The previous sections of the analysis yield several key conclusions. First and foremost, we observed a significant class imbalance between the True class (Legendary Pokemon) and the False class (Non-Legendary Pokemon). We found that Categorical Naive Bayes performed better than Gaussian Naive Bayes on our dataset, despite the presence of both continuous and categorical features. In case of Categorical Naive Bayes classifier, equal width binning demonstrated better performance compared to equal frequency binning when discretizing the continuous features.

Our experimentation demonstrated the importance of handling class imbalance, choosing appropriate classifiers based on data characteristics, and using hybrid approaches to leverage the strengths of different models. The Hybrid Naive Bayes classifier, combining Gaussian and Categorical Naive Bayes, emerged as the most accurate model for the Pokemon dataset.

REFERENCES

[1] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, no. 60, pp. 1–8, 2006.

[2] Z.-H. Zhou, *Machine learning*. Springer Nature, 2021.

[3] H. Kamel, D. Abdulah, and J. M. Al-Tuwaijari, "Cancer classification using gaussian naive bayes algorithm," in *2019 international engineering conference (IEC)*. IEEE, 2019, pp. 165–170.

[4] C. Schwenke and A. Schering, "True positives, true negatives, false positives, false negatives," *Wiley StatsRef: Statistics Reference Online*, 2014.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[6] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," *GESTS International Transactions on Computer Science and Engineering*, vol. 32, no. 1, pp. 47–58, 2006.

Bishwambhar Dahal is a fourth-year student of Electronics, Communication, and Information Engineering. With a deep fascination for Artificial Intelligence (AI), He is driven by the potential of AI to transform industries and tackle complex challenges. His academic journey has equipped him with a strong foundation in AI concepts, including machine learning and data analysis. He possess a relentless curiosity and is always eager to explore the latest advancements in AI. His goal is to apply his knowledge and skills in AI to make meaningful contributions to research and development, pushing the boundaries of what is possible with intelligent algorithms.(THA076BEI009)

Sirjana Bhatta is a fourth-year student of Electronics, Communication, and Information Engineering with a keen interest in the field of Artificial Intelligence (AI). She possess a strong academic foundation and practical skills in machine learning, deep learning, and data analysis. Her passion lies in leveraging AI to revolutionize industries and solve complex problems. She is a motivated learner, constantly staying updated with the latest advancements in AI. She is seeking opportunities to contribute to AI research and development and make a positive impact on society. (THA076BEI036)

Appendix

A Figures

1 Block Diagram

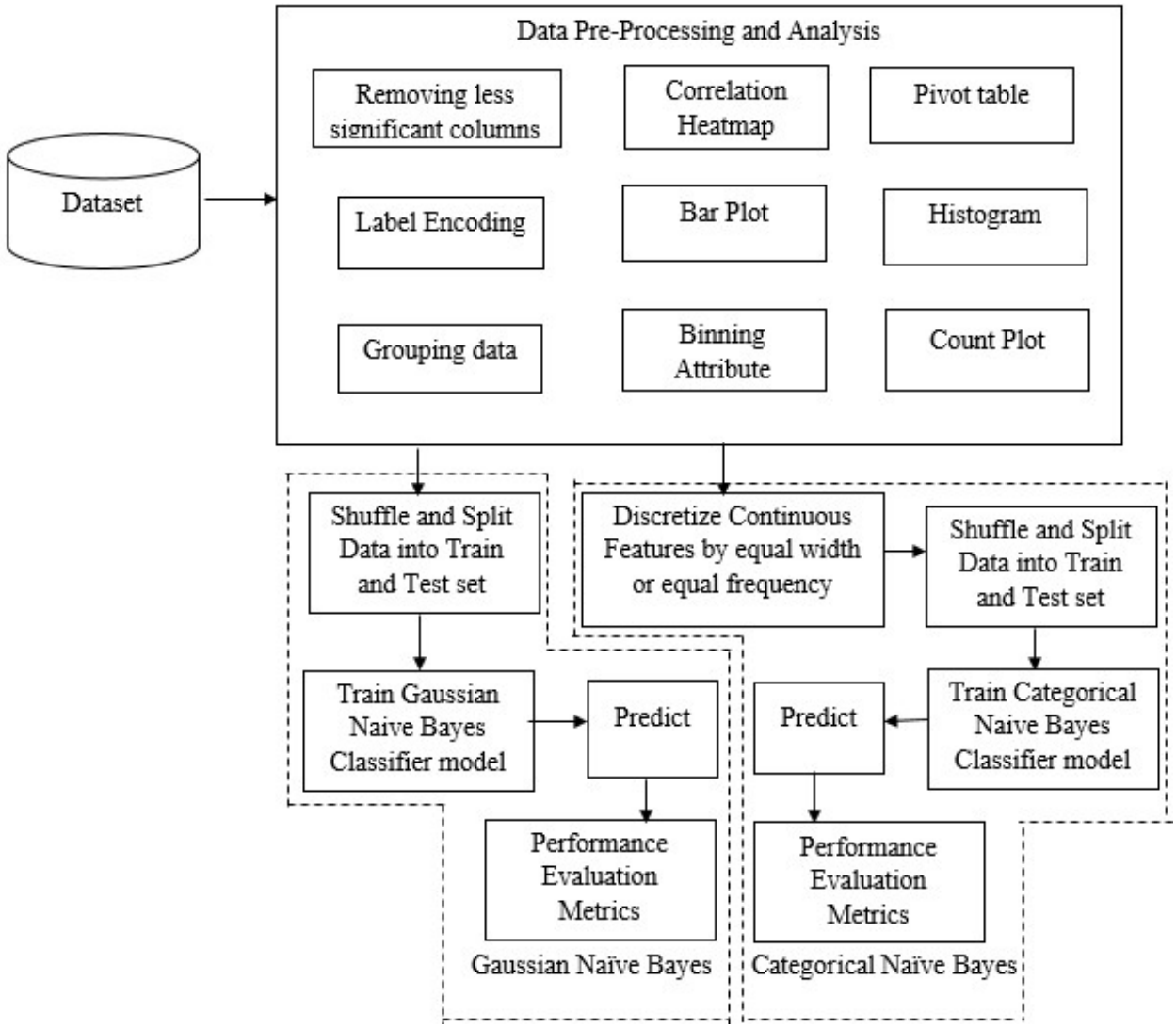

Figure 1: Block Diagram of Gaussain and Categorical Naive Bayes Classifier.

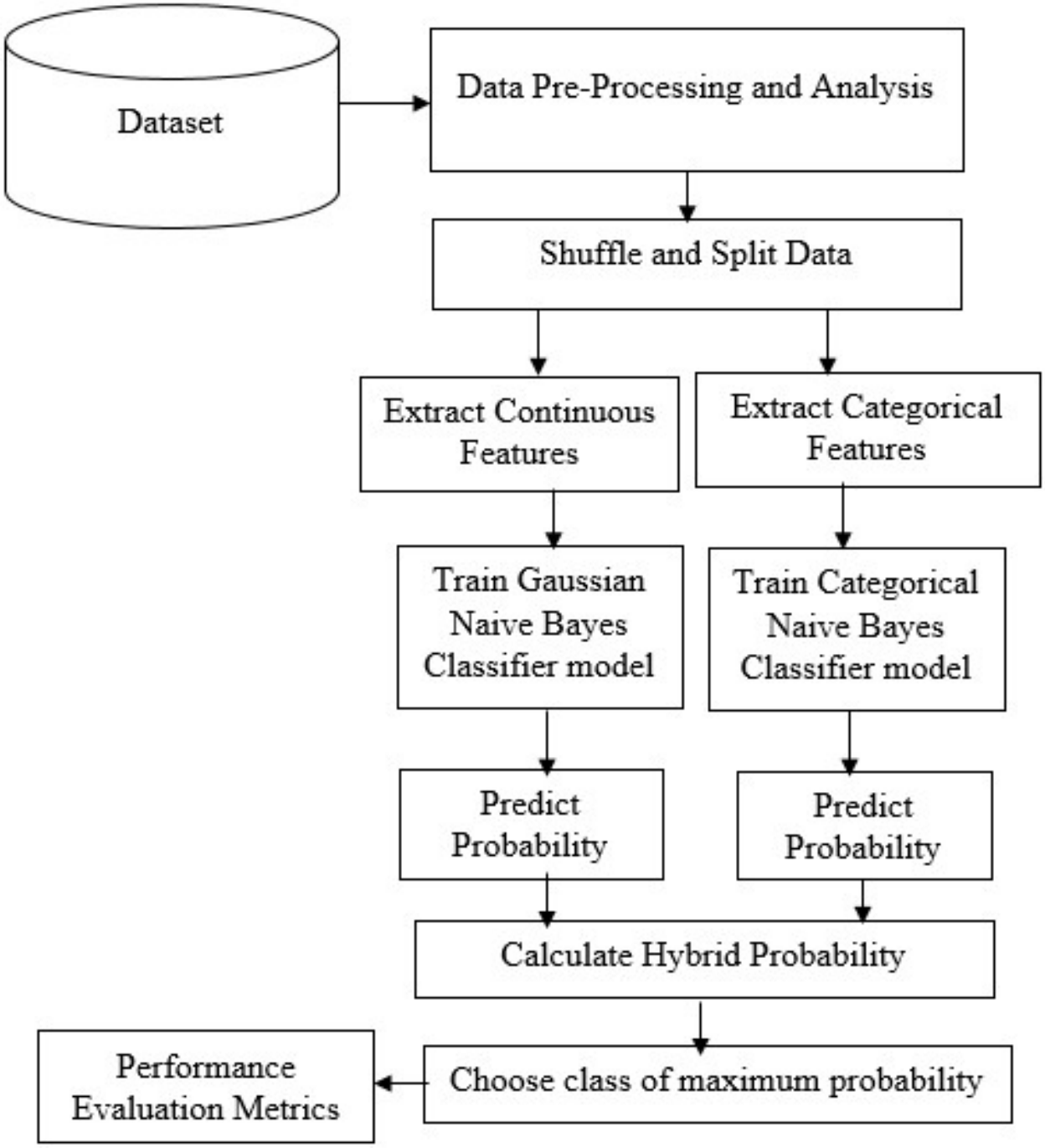

Figure 2: Block Diagram of Hybrid Naive Bayes Classifier.

2 Result

1. Dataset Analysis and Visualization

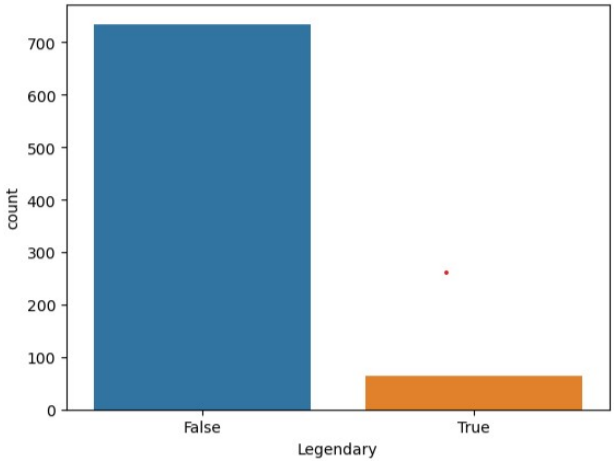

Figure 3: Dataset distribution

2. Result

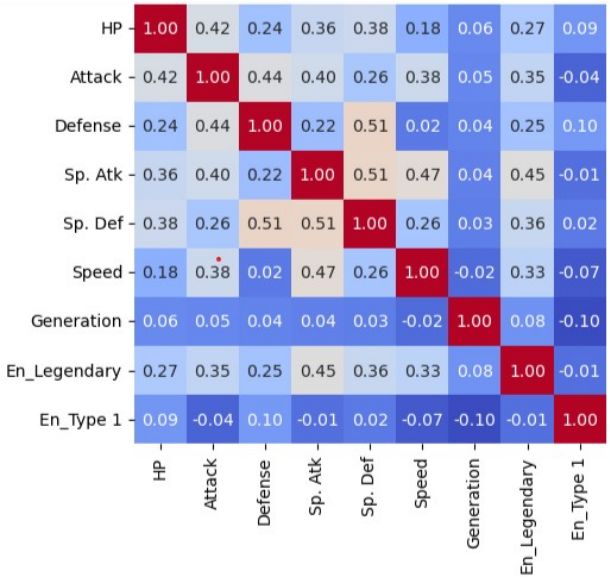

Figure 4: Heatmap of corelation between features.

Generation	1	2	3	4	5	6
Legendary	0.036145	0.04717	0.1125	0.107438	0.090909	0.097561

Figure 5: Pivot table between Generation and Legendary.

	Legendary			
attack_cat	5 to 50	51 to 100	101 to 150	191 to 190
gen_cat				
1 to 3	0.009259	0.062241	0.1250	0.363636
4 to 6	0.000000	0.050228	0.2625	0.571429

Figure 6: Pivot table between Generation, Attack and Legendary.

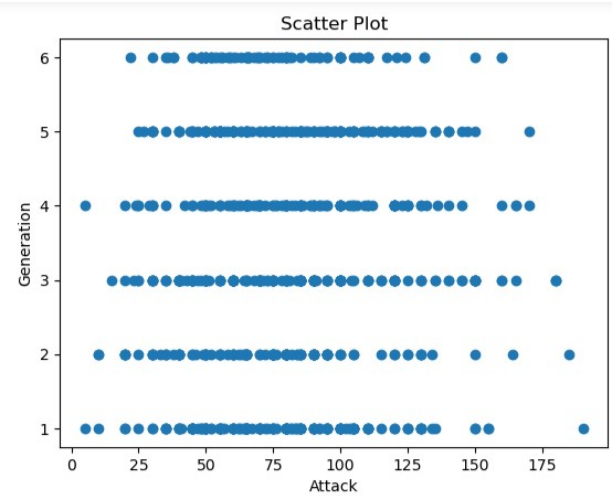

Figure 7: Scatter plot between Generation and Attack.

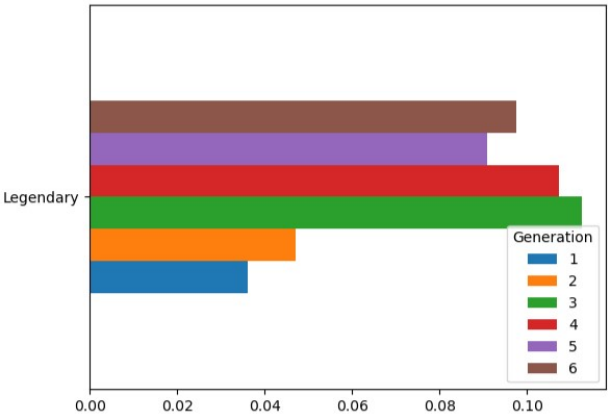

Figure 8: Bar graph of pivot table between Generation and Legendary.

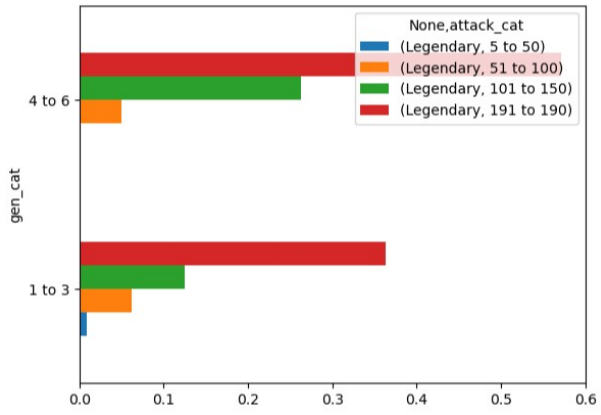

Figure 9: Bar garaph of Pivot table between Generation, Attack and Legendary.

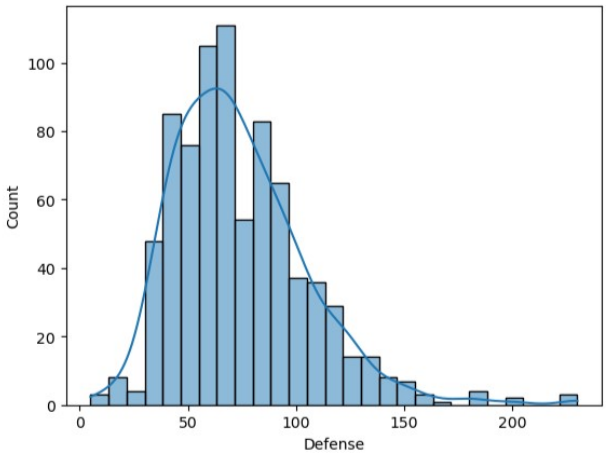

Figure 12: Histogram plot of Defense attribute.

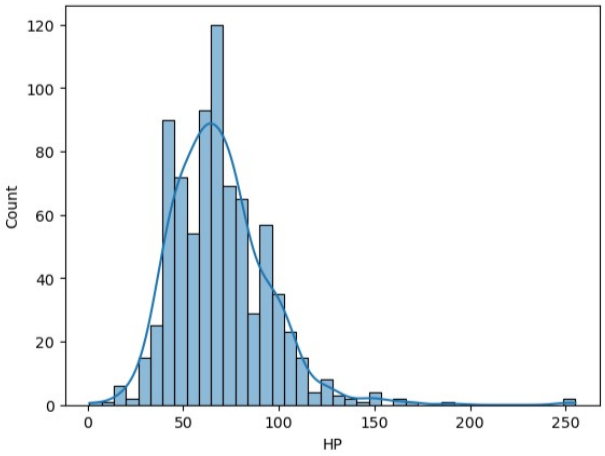

Figure 10: Histogram plot of HP attribute.

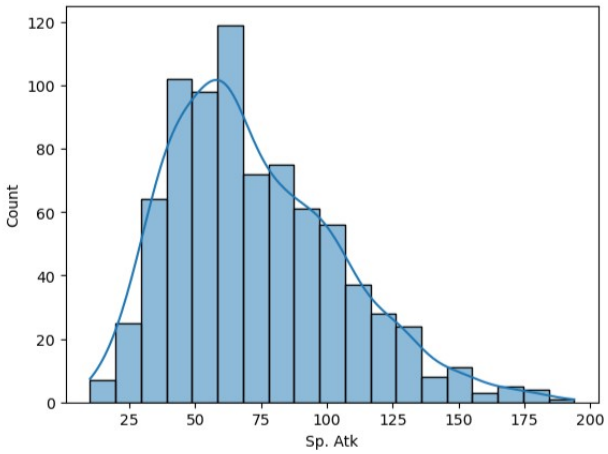

Figure 13: Histogram plot of Sp. Atk attribute.

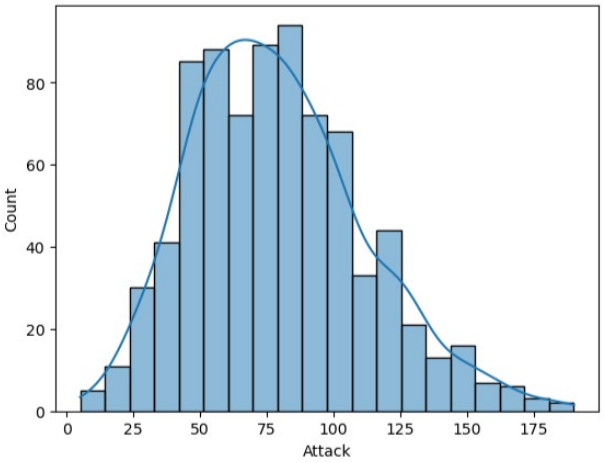

Figure 11: Histogram plot of Attack attribute.

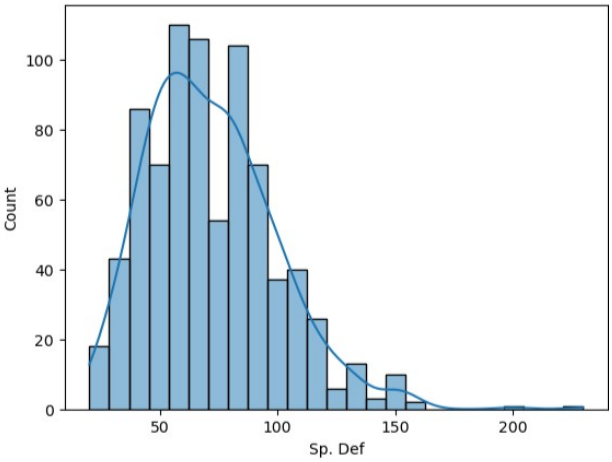

Figure 14: Histogram plot of Sp. Def attribute.

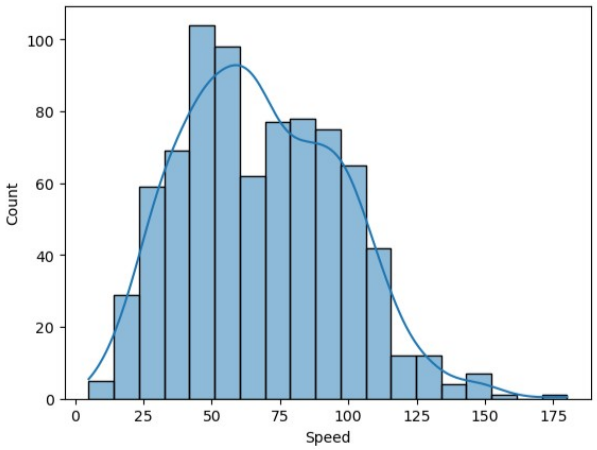

Figure 15: Histogram plot of Speed attribute.

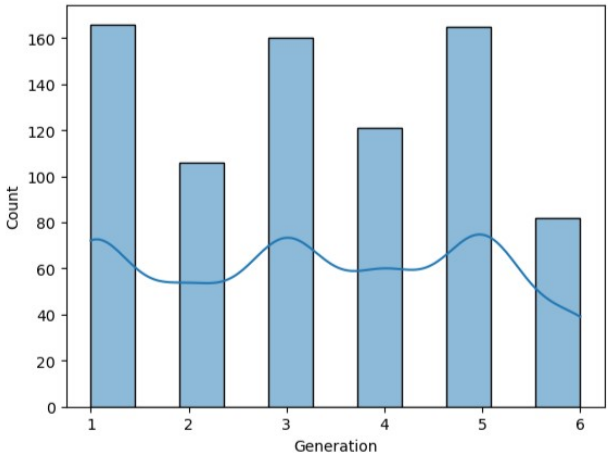

Figure 16: Histogram plot of Generation attribute.

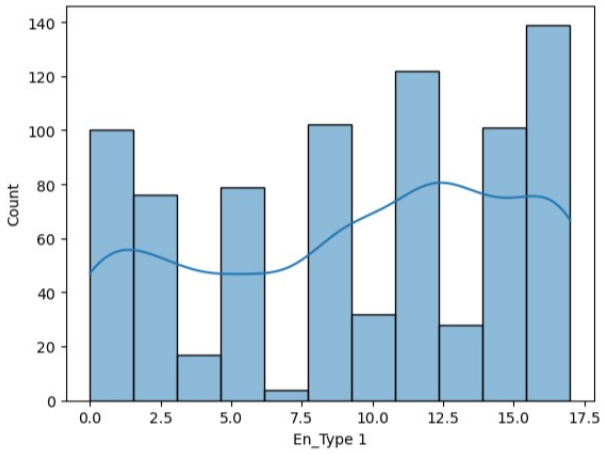

Figure 17: Histogram plot of En_Type 1 attribute.

3. Performance Metrics

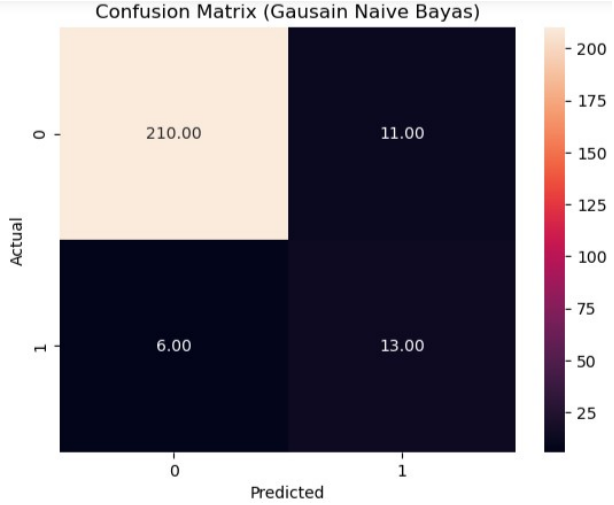

Figure 18: Confusion matrix of Gaussian Naive Bayes classifier.

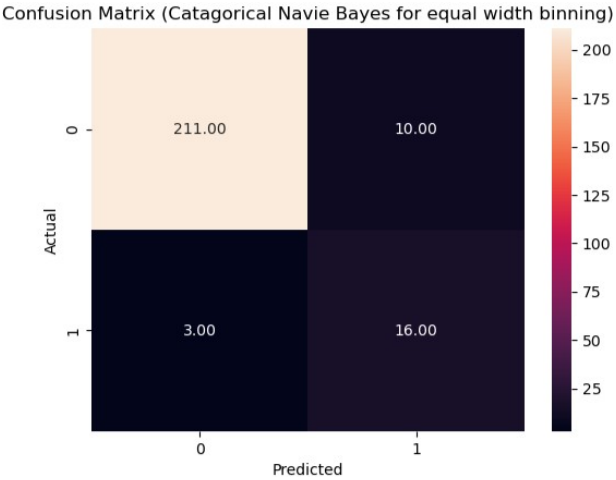

Figure 19: Confusion matrix of Categorical Naive Bayes classifier for equal width.

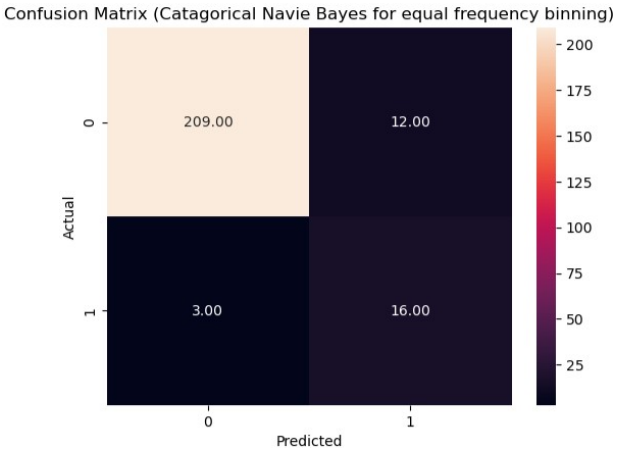

Figure 20: Confusion matrix of Categorical Naive Bayes classifier for equal frequency.

Classification Report (Catagorical Navie Bayes for equal width binning)

	precision	recall	f1-score	support
0	0.9860	0.9548	0.9701	221
1	0.6154	0.8421	0.7111	19
accuracy			0.9458	240
macro avg	0.8007	0.8984	0.8406	240
weighted avg	0.9566	0.9458	0.9496	240

Figure 23: Classification Report of Categorical Naive Bayes classifier for equal width.

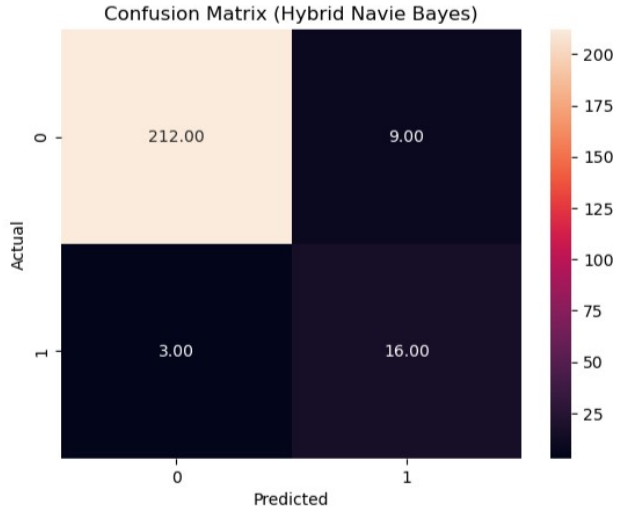

Figure 21: Confusion matrix for Hybrid Naive Bayes classifier

Classification Report (Catagorical Navie Bayes for equal frequency binning)

	precision	recall	f1-score	support
0	0.9858	0.9457	0.9654	221
1	0.5714	0.8421	0.6809	19
accuracy			0.9375	240
macro avg	0.7786	0.8939	0.8231	240
weighted avg	0.9530	0.9375	0.9428	240

Figure 24: Classification Report of Categorical Naive Bayes classifier for equal frequency.

Classification Report (Gausain Naive Bayas)

	precision	recall	f1-score	support
0	0.9722	0.9502	0.9611	221
1	0.5417	0.6842	0.6047	19
accuracy			0.9292	240
macro avg	0.7569	0.8172	0.7829	240
weighted avg	0.9381	0.9292	0.9329	240

Figure 22: classification report of Gaussian Naive Bayes classifier.

Classification Report (Hybrid Navie Bayes)

	precision	recall	f1-score	support
0	0.9860	0.9593	0.9725	221
1	0.6400	0.8421	0.7273	19
accuracy			0.9500	240
macro avg	0.8130	0.9007	0.8499	240
weighted avg	0.9587	0.9500	0.9531	240

Figure 25: Classification Report of Hybrid Naive Bayes classifier.