STABLE DISTRIBUTION NAIVE BAYES ACHIEVES HIGHER ACCURACY THAN TRADITIONAL NAIVE BAYES CLASSIFICATION

Xiaoying Zeng and Eugene Pinsky

Department of Computer Science, Metropolitan College, Boston University, Boston, MA, USA

ABSTRACT

This study evaluates the performance of the Stable Distribution Naive Bayes Classifier on the well-known IRIS dataset, comparing it against the traditional Naive Bayes Classifier. The Stable Distribution Classifier, well-suited for data with heavy tails and skewness, consistently achieves superior accuracy, especially when handling outliers and non-standard samples. This study conducted 18 feature combinations of Iris Versicolor and Iris Virginica across varying parameter configurations (□, □), demonstrating the stable model's robustness under constrained sample sizes. A significant technical contribution involves integrating R's specialized stable package into Python, enabling the direct application of professional fitting and PDF functions for precise analysis. Representative results from key feature combinations further illustrate its practical advantages. Additionally, five additional datasets—Wine, Social Network Ads, Diabetes, Electrical Grid Stability Simulated, and Vehicle Silhouettes—further demonstrate the Stable Distribution Classifier's broad applicability across diverse domains. This research further confirms that the Stable Distribution Naive Bayes Classifier is a robust and accessible alternative, offering enhanced predictive performance over models traditionally based on Gaussian distribution assumptions.

KEYWORDS

Stable Distribution, Naive Bayes Classifier, Heavy-tailed Distributions, Skewness, Model Robustness

1. INTRODUCTION

The Naive Bayes Classifier is a widely used probabilistic classification method due to its computational efficiency and strong performance in high-dimensional settings [1, 2]. It is capable of handling both continuous and categorical data and remains a fundamental approach in machine learning. However, a key limitation of traditional Naive Bayes classifiers is the assumption of feature independence, which is often unrealistic in real-world applications. When features exhibit strong correlations, classification performance may degrade significantly [3].

In addition to the independence assumption, traditional Gaussian Naive Bayes classifiers often assume that continuous features follow a normal distribution. This assumption is restrictive, as real-world data frequently displays skewness, heavy tails, or multimodal distributions, deviating significantly from the Gaussian model. Normal distributions are symmetric and have thin tails,

 DOI:10.5121/ijci.2025.140205

while many practical datasets exhibit long-range dependencies and extreme values that cannot be effectively modeled under this assumption.

To address these limitations, researchers have proposed alternative probabilistic models that relax the Gaussian assumption. One promising approach is the Stable Distribution-Based Naive Bayes Classifier, which replaces the normal distribution with the stable distribution—a more general class of probability distributions that allows for skewness and heavy tails. Theoretical foundations of stable distributions have been extensively studied in probability theory [4], and their practical applications have been explored in fields such as finance, signal processing, and machine learning [5]. Unlike traditional Gaussian-based methods, the Stable Distribution Naive Bayes model is better suited for real-world datasets exhibiting heavy tails and extreme values.

Our study makes the following key contributions:

1. **Customization of the Naive Bayes Classifier:** We extend the Naive Bayes framework by incorporating -stable distributions, allowing it to handle heavy-tailed and skewed data that the traditional Gaussian Naive Bayes struggles with.
2. **Integration of Statistical Tools:** By integrating R's specialized stable distribution functions into Python's machine learning framework, we enable efficient generation, fitting, and probability density estimation for stable distributions, improving computational accuracy and flexibility.
3. **Improved Accuracy for Heavy-Tailed and Skewed Data:** The Stable Naive Bayes Classifier consistently outperforms the Gaussian model, particularly when α is less than 2, demonstrating its robustness in modeling non-Gaussian data distributions.
4. **Practical Applications:** Beyond the IRIS dataset, we validate the model's effectiveness on five additional real-world datasets, confirming its superior adaptability and classification performance across diverse domains.

2. STABLE DISTRIBUTIONS

Stable distributions are a highly versatile family of probability distributions, widely used for modeling complex real-world phenomena, particularly in finance and physics. These distributions are characterized by four key parameters: alpha (α) determines tail behavior, beta (β) controls skewness, gamma (γ) represents scale, and delta (δ) defines location. This flexibility allows stable distributions to effectively capture heavy tailed and asymmetric data patterns, making them a powerful tool in probabilistic modeling.

Stability parameter $\alpha \in (0, 2]$, skewness parameter $\beta \in [-1, 1]$, scale parameter $c \in (0, \infty)$, location parameter $\mu \in (-\infty, \infty)$.

Examining the six plots in Figure 1, we can explore the distinctive characteristics of stable distributions in greater detail.

1. Alpha (α) Parameter: Alpha determines the "tail heaviness" of the distribution. When α decreases, the distribution exhibits heavier tails, indicating a greater likelihood of extreme values deviating significantly from the mean. This is crucial for modeling financial returns that can have heavy tails, unlike the normal distribution that underestimates the probabilities of extreme events.
2. Beta (β) Parameter: Beta controls the skewness of the distribution. A β value of 0 results in a symmetric distribution, whereas positive β values create right-skewness, and negative β values lead to left-skewness. In the context of the plots, the first and fourth

plots show how varying α affects symmetric distributions (β=0), indicating that as α decreases, the distribution flattens and the tails become heavier.

3. Gamma (γ) and Delta (δ) Parameters: While gamma (often represented as c for scale) and delta (μ for location) are not the focus of these plots, they are nonetheless important. Gamma stretches or shrinks the distribution, affecting the dispersion of data points, while delta shifts it left or right along the x-axis. For our plots, gamma (c) is set to 1, indicating a standard scale, and delta (μ) is set to 0, placing the center of the distribution at the origin.

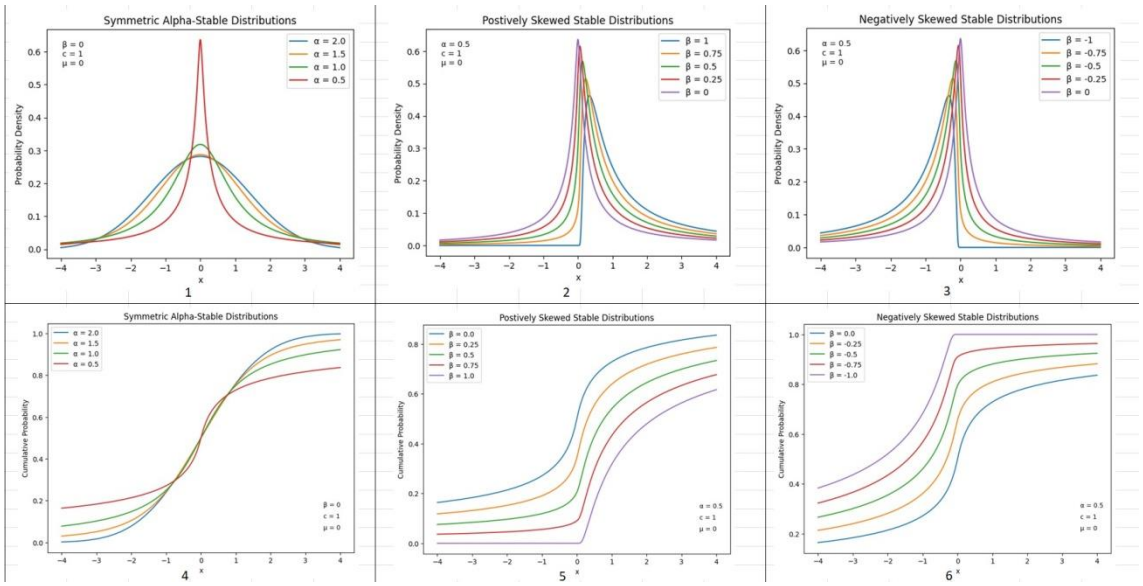

Figure 1. Probability density function and Cumulative distribution function, adapted from [6].

Now, the second and fifth plots show the effect of positive β values with a fixed α of 0.5, illustrating rightskewed distributions. The more positive the β, the more pronounced the skew to the right. This skewness can model data that has a tendency to produce values significantly larger than the mode more frequently than significantly smaller ones.

Conversely, the third and sixth plots showcase negative β values with a fixed α of 0.5, demonstrating leftskewed distributions. In this case, negative β values produce an extended left tail, making them suitable for modeling data where extreme negative values occur more frequently.

Therefore, stable distributions, with their flexibility to model skew and heavy tails, are incredibly useful. They enable us to capture the characteristics of real-world data that are not well described by the normal distribution, particularly in fields where outliers and extreme values are common.

3. WHY DATASETS CONFORM TO STABLE DISTRIBUTIONS?

3.1. Iris Dataset

In Figure 2, Iris-versicolor (red) and Iris-virginica (green) exhibit significant distribution overlap, particularly in sepal length and sepal width, making class separation more challenging. Additionally, both classes display fat tails and skewness, especially in petal length and petal width, where the distributions extend further on one side, indicating non-Gaussian characteristics.

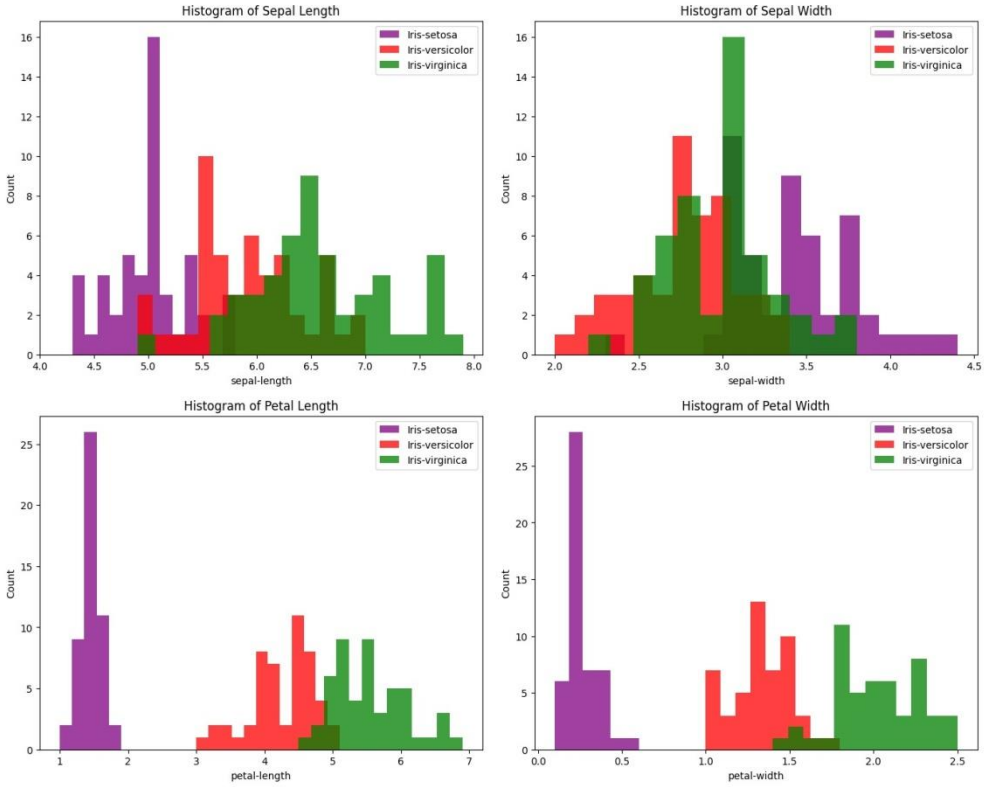

Figure 2. Histograms of 4 features in IRIS Dataset

The probability formulation of the Naive Bayes classifier is derived based on the conditional independence assumption [7]. We can now apply this approach to specific features, such as sepal width and petal length, to estimate the most likely class for a given observation.

In classification, the term $P(\mathbf{x})$ acts as a normalization constant and does not influence class comparisons. As a result, the classification decision is based on evaluating $P(\mathbf{x} \mid Ck) \cdot P(Ck)$. Given that the prior probabilities $P(Ck)$ are equal across classes, the final decision is primarily determined by the likelihood values $P(\mathbf{x} \mid Ck)$.

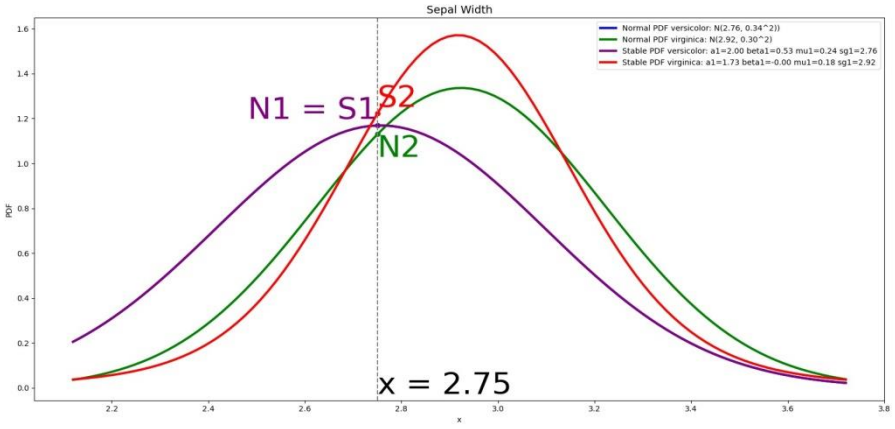

Figure 3. Stable Distribution Naive Bayes classifier vs. Normal Distribution Naive Bayes classifier on Sepal Width (Blue and purple distribution overlapped)

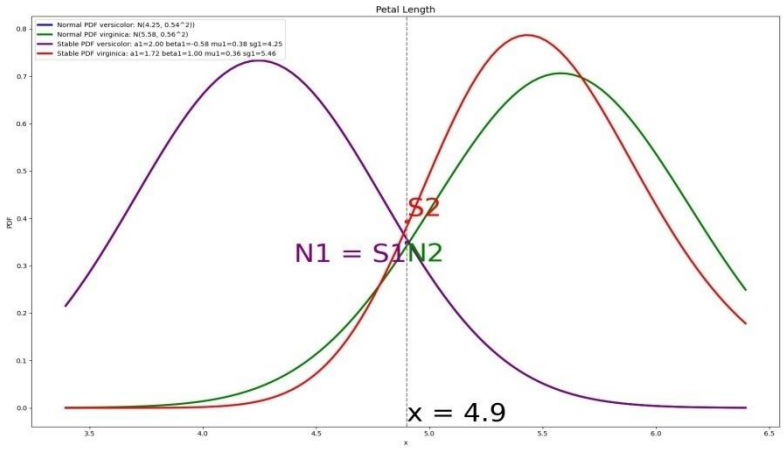

Figure 4. Stable Distribution Naive Bayes classifier vs Normal Distribution Naive Bayes classifier on Petal Length (Blue and purple distribution overlapped)

For SepalWidth = 2.75, under the Normal Distribution Naive Bayes classifier, the PDF value for Versicolor (point N1) is larger than that for Virginica (point N2), meaning $P(\mathbf{x} \mid C1)$ is greater than $P(\mathbf{x} \mid C2)$. Thus, the Naive Bayes classifier is more inclined to classify the data point as Versicolor. This is illustrated in Figure 3.

In contrast, under the Stable Distribution Naive Bayes assumption, the PDF value for Versicolor (point S1) is smaller than that for Virginica (point S2), meaning $P(\mathbf{x} \mid C1)$ is less than $P(\mathbf{x} \mid C2)$. As a result, the

Stable Distribution Naive Bayes classifier tends to classify the data point as Virginica.

For Petal Length = 4.9, under the Normal Distribution Naive Bayes classifier, the PDF value for Versicolor (point N1) is the same as that for Virginica (point N2), meaning $P(\mathbf{x} \mid C1)$ is equal to $P(\mathbf{x} \mid C2)$. Thus the Normal Distribution Naive Bayes classifier shows equal confidence in classifying the data point as either Versicolor or Virginica. This is illustrated in Figure 4.

In contrast, under the Stable Distribution Naive Bayes assumption, the PDF value for Versicolor (point S1) is smaller than that for Virginica (point S2), meaning $P(\mathbf{x} \mid C1)$ is less than $P(\mathbf{x} \mid C2)$. This suggests that, according to the stable distribution model, a petal length of 4.9 is more likely to belong to the Virginica.

3.2. Other Datasets

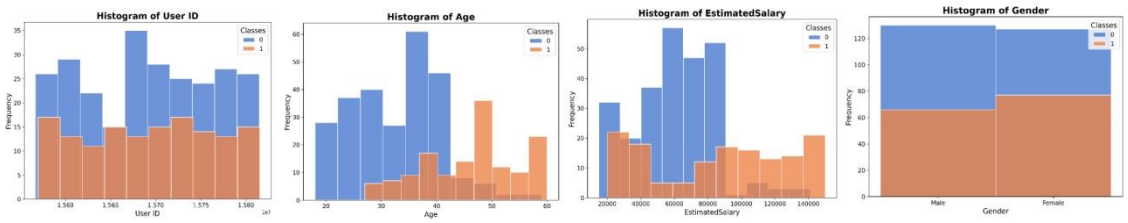

Figure 5. Social Network Ads Dataset

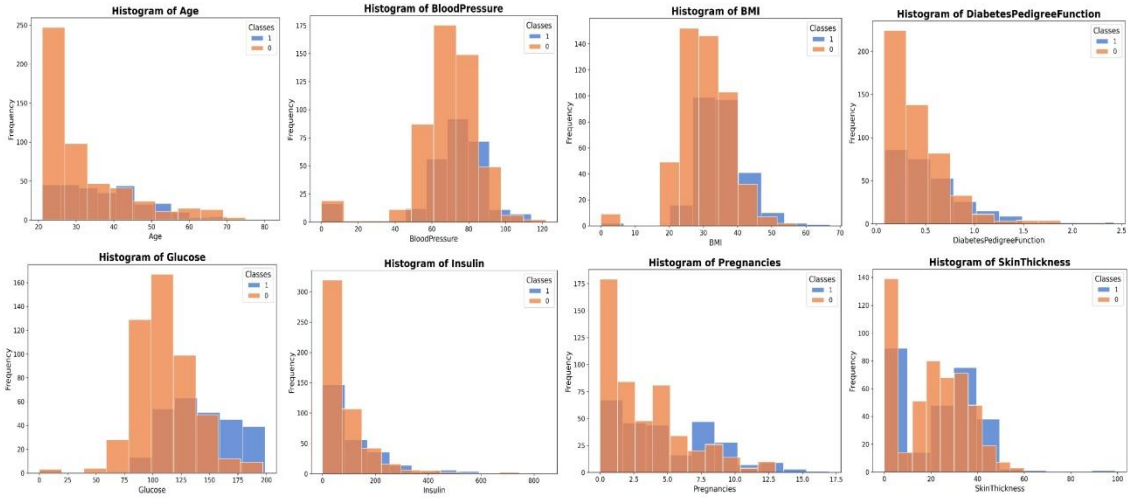

Figure 6. Diabetes Dataset

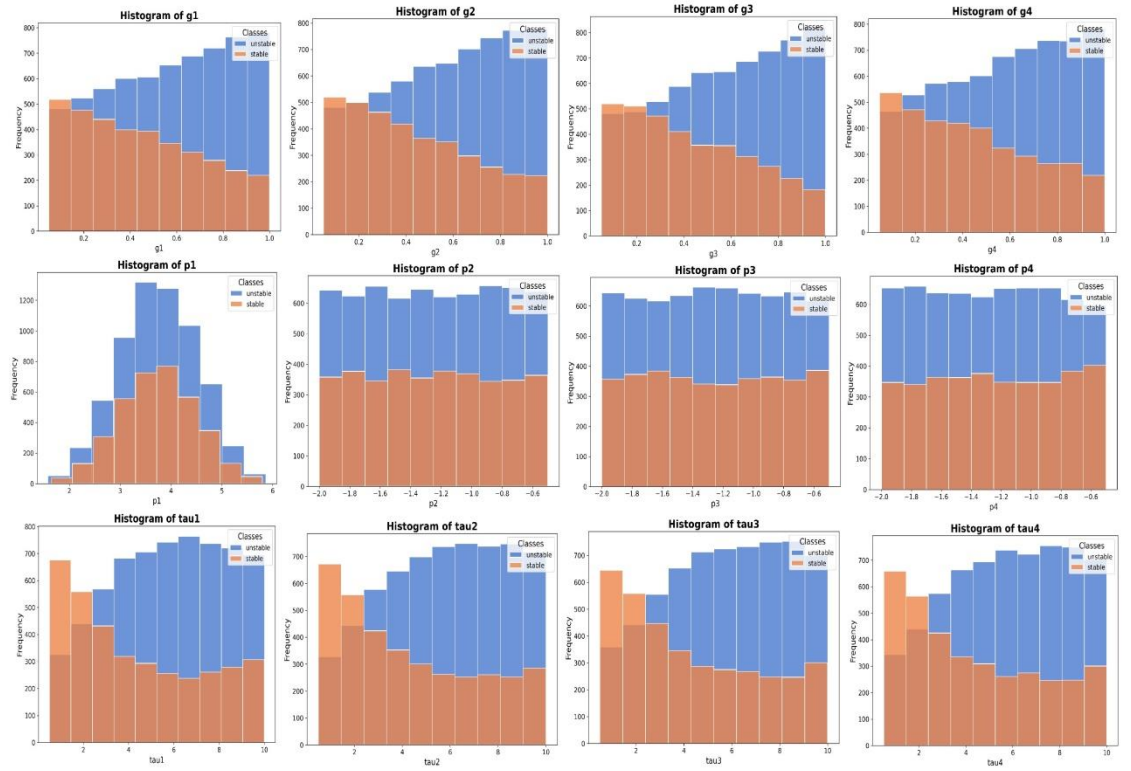

Figure 7. Electrical Grid Stability Simulated Data

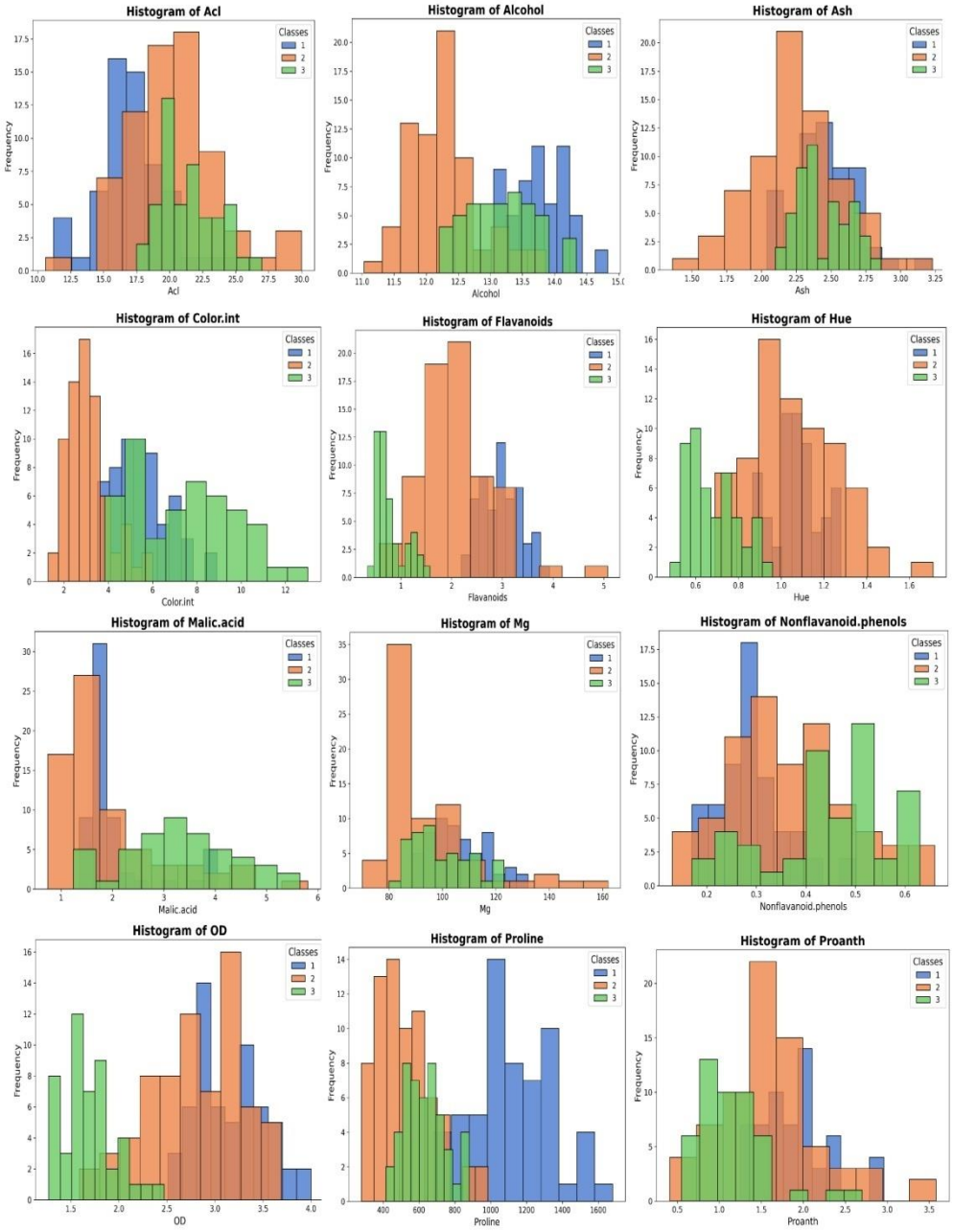

Figure 8. Wine Dataset

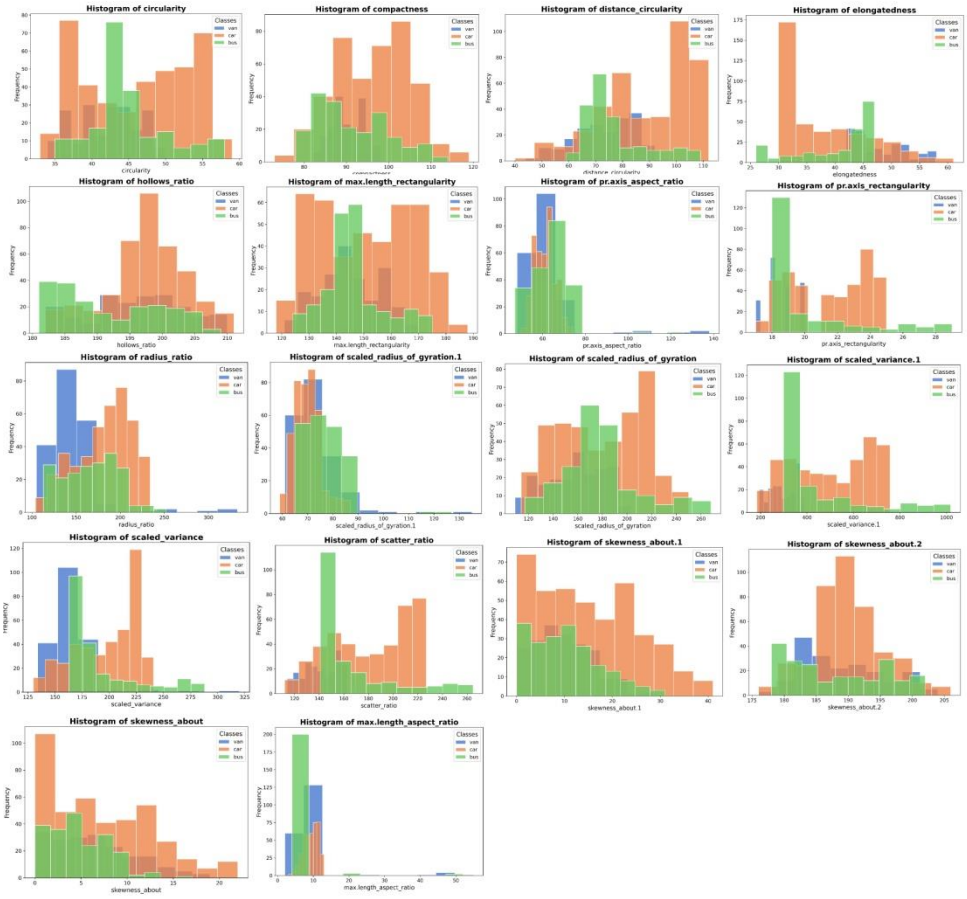

Figure 9. Vehicle Silhouettes Dataset

The datasets exhibit substantial overlap between class distributions, making simple probabilistic separation challenging. This suggests that Naive Bayes classifiers relying on Gaussian assumptions may perform poorly, and Stable Distribution Naive Bayes could be beneficial in improving classification accuracy.

4. METHODOLOGY

4.1. Iris Dataset Generation and Modeling

To evaluate the Stable Distribution Naive Bayes Classifier on non-normal data, a synthetic dataset was generated using stable distributions. This process ensures that the modeled data retains the statistical characteristics of the original Iris dataset while introducing heavy-tailed and skewed properties commonly observed in real-world datasets.

Taking the generation of Sepal Width as an example, to maintain the overlap between classes while adhering to the characteristics of stable distributions, we referenced the statistical properties of Sepal Width from the original dataset. Specifically, the mean and variance for Iris-versicolor were 5.94 and 0.516, respectively, while those for Iris-virginica were 6.59 and 0.636. These statistics were used to determine the delta (mean) and gamma (scale, calculated as the square root of 2 times the variance) for the generated dataset. Based on these parameters, we derive the following distributions.

- **Distribution 1:** (α, β, γ = 0.73, δ = 5.94)
- **Distribution 2:** (α, β, γ = 0.90, δ = 6.59)

This ensures that the generated data accurately reflects the distinct characteristics of the two classes.

To create a balanced dataset, 50 data points were generated for each class (Iris-versicolor and Iris-virginica), and the two datasets were concatenated. The data were then randomly shuffled to ensure independence and eliminate potential bias. Finally, the corresponding class labels (0 for Iris-versicolor and 1 for Iris-virginica) were added to form the final training data set.

The decision to use a sample size of 50 data points per class was based on extensive testing with varying sample sizes. We evaluated accuracy across sample sizes of 200, 400, 800, and 1000 in a total of 288 experiments. The results demonstrated that, while larger sample sizes improved overall accuracy for stable distribution models, they also increased the performance gap between stable and Gaussian-based models.

The MATLAB stblrnd function was specifically chosen for its ability to accurately simulate stable distributions while remaining independent of Python and R. This independence ensures that the data generation process is not influenced by the tools used for subsequent analysis, thus avoiding any potential dependencies. By separating data generation from model training, we effectively ensured the integrity and objectivity of the experimental results.

4.2. Other Dataset Preparation

Wine Dataset[12] contains 13 chemical attributes for classifying three types of wine (3 classes).
Social Network Ads Dataset[13] contains features such as age and gender to classify whether a user has made a purchase (binary classification).

Diabetes Dataset[14] comprises 8 health indicators to predict whether a person has diabetes (2 classes).
Electrical Grid Stability Simulated Dataset[15] provides 12 features to predict whether the power grid is stable (2 classes).

Vehicle Silhouettes Dataset[16] has 18 features for classifying four types of vehicles (4 classes).

4.3. Model Implementation

The Stable Distribution Naive Bayes Naive Bayes Classifier is also as user-friendly and accessible as the Normal Distribution Naive Bayes Classifier. It integrates seamlessly into analytical workflows, providing a familiar interface for those accustomed to traditional statistical models. It is easy to use.

In terms of availability, many statistical and machine learning packages across various programming languages have implemented the Stable Distribution Naive Bayes Classifier. For example: In Python, we can use the ”scipy.stats“ module which provides a variety of distributions, including stable distributions, that can be leveraged to implement a Stable Naive Bayes Classifier. Packages like ”scikit-learn“ offer a framework for creating custom classifiers, which can be extended to use stable distributions.

In the realm of R, we have a wealth of packages at your disposal for implementing Stable Distribution Naive Bayes Classifiers, each adept in handling the intricacies of stable distributions. The ”fBasics“ and ”stabledist“ packages offer comprehensive tools for calculating probabilities, and densities, and for simulating data. For more advanced statistical modeling and fitting, ”StableEstim“ and stable provide robust methods for parameter estimation and inference. We use STABLE 5.3 toolkit developed by Robust Analysis, Inc. This toolkit is designed for handling stable distributions, making it ideal for analyzing heavy-tailed or skewed data, especially in finance. It provides functions to calculate densities, and distribution functions, and to simulate and fit data to stable distributions. This allows for a robust analysis of datasets that do not conform to the normal distribution, making it a valuable tool in statistical modeling and inference. Whether you’re estimating parameters with ”stablefit“, generating random samples with ”rstable“, or checking the fit of your model with diagnostic functions, the stable package equips you with everything you need to harness the power of stable distributions in R.

Please see details in the appendix.

5. IRIS DATASET RESULTS

This study compares the accuracy of two classifiers in predicting class membership based on four distinct features.

Table 1. Comparative Accuracy of Generated Dataset using Different Values of α and β for four features

Sepal Length & Sepal Width & Petal Width & Petal Length						
Generated Alpha	Accuracy (= 0.5)		Accuracy (= 0)		Accuracy (= -0.5)	
	Stable	Normal	Stable	Normal	Stable	Normal
1.5	0.73	0.43	0.87	0.76	0.73	0.73
1.6	0.80	0.70	0.80	0.77	0.83	0.70
1.7	0.83	0.60	0.83	0.80	0.75	0.70
1.8	0.67	0.70	0.83	0.80	0.78	0.73
1.9	0.93	0.93	0.83	0.80	0.85	0.85
2.0	0.87	0.87	0.73	0.73	0.85	0.85

In Table 1, four features are selected for predicting classifications at different locations where β is separately set to 0.5, 0, and -0.5.The overall performance of Stable Distribution Naive Bayes Classifier is better than the Normal Distribution Naive Bayes Classifier given different alpha values.

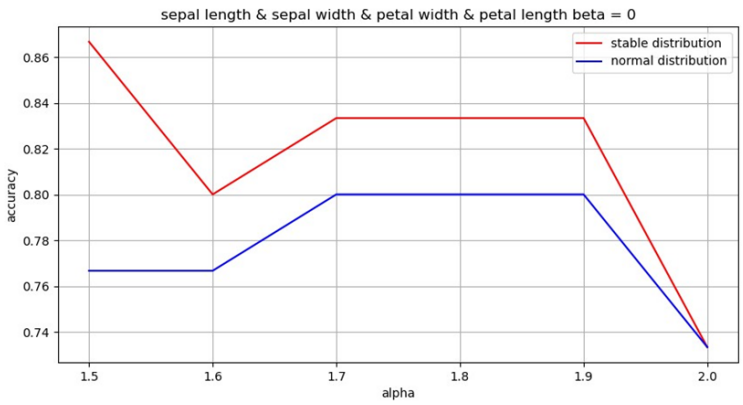

Figure 10. β = 0 (Symmetric)

In Figure 10, while β = 0, both algorithms are performing well with overall accuracy over 70% but there is a clear performance difference between Stable Distribution Naive Bayes Classifier and Normal Distribution Naive Bayes Classifier.

From the experimental results, the accuracy of classification for Stable Distribution Naive Bayes Classifier at different α is greater than or equal to those of Normal Distribution Naive Bayes Classifier. Furthermore, where α is between 1.6 and 1.9, the accuracy of the Stable Distribution Naive Bayes Classifier is 3% higher than the Normal Distribution Naive Bayes Classifier. A maximum difference of 11% occurs when α =1.5, where accuracy of classification for Stable Distribution Naive Bayes Classifier is 87%, comparing to 76% for the Normal Distribution Naive Bayes Classifier.

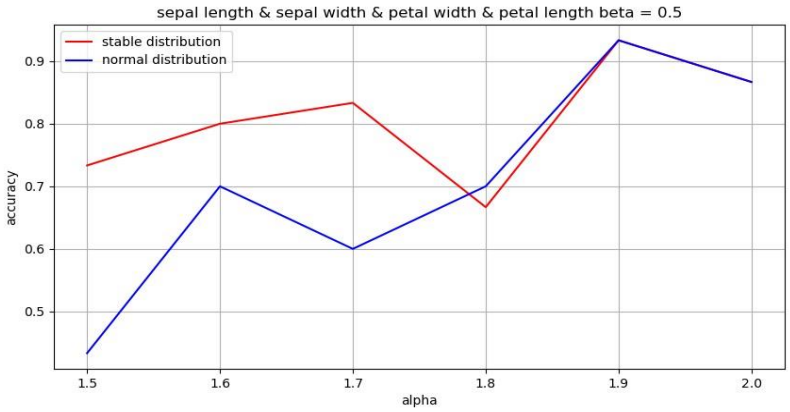

Figure 11. β = 0.5 (Right Skew)

In Figure 11, while β = 0.5, the overall performance fluctuates, especially for Normal Distribution Naive Bayes Classifier accuracy ranges from 43% to 93% based on different alpha values used. At the same time, Stable Distribution Naive Bayes Classifier turns to be more stable in classification in range from 67% to 93% while maintaining a high level of accuracy.

Only exception happens when α =1.8, where the accuracy of the Stable Distribution Naive Bayes Classifier is 67%, which is slightly lower than the 70%, accuracy of Normal Distribution Naive Bayes Classifier, with 3% difference, the overall performance of Stable Distribution Naive Bayes Classifier is much better. The maximum difference of 30% occurs at α =1.5, where the classification accuracy for Stable Distribution Naive Bayes Classifier is 73%, compared to 43% for Normal Distribution Naive Bayes Classifier.

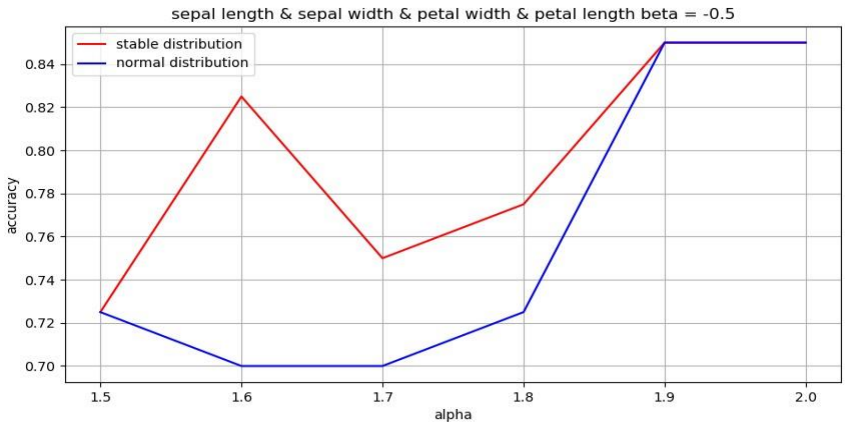

Figure 12. β = -0.5 (Left Skew)

In Figure 12, while $\beta = -0.5$, both algorithms are performing well with overall accuracy over 70%. From figure we can still see the leading performance in Stable Distribution Naive Bayes Classifier. Accuracy of classification for Stable Distribution Naive Bayes Classifier at different is greater than or equal to those of Normal Distribution Naive Bayes Classifier. A maximum difference of 13% occurs when α =1.6, giving Stable Distribution Naive Bayes Classifier accuracy of 83%, higher than Normal Distribution Naive Bayes Classifier of 76%.

Given the experimental results on the Iris dataset, the best accuracy is 93% for both Stable and Normal Distribution Naive Bayes Classifiers at α =1.9 and β =0.5, However, overall, the Normal Distribution Naive Bayes Classifier shows greater fluctuation in accuracy, with the lowest accuracy dropping to 43%, which is even below random chance (50%).

Out of 18 varying combinations of alpha and beta, 11 combinations exhibit higher accuracy with the Stable Distribution Naive Bayes Classifier compared to the Normal Distribution Naive Bayes Classifier.

6 combinations exhibit the same performance for the Stable Distribution Naive Bayes Classifier and the Normal Distribution Naive Bayes Classifier, and only 1 combination shows lower accuracy with the Stable Distribution Naive Bayes Classifier.

The result implies that in 94% of the parameter combinations, the Stable Distribution Naive Bayes Classifier either matches or outperforms the Normal Distribution Naive Bayes Classifier in terms of accuracy, and in 61% of them, the Stable Distribution Naive Bayes Classifier outperforms the Normal Distribution Naive Bayes Classifier in terms of accuracy. This result indicates that Stable Distribution Naive Bayes Classifier deals well with various parameter distributions that are often seen in real-world datasets where data points are not normally distributed.

So far, we have analyzed the Stable Distribution Naive Bayes Classifier, which demonstrates overall superior performance with higher accuracy of classification and ita stability. The stable model's adaptability through α, β, γ, δ makes it highly effective for real-world datasets, especially when data deviates from normality.

6. CROSS-DATASET PERFORMANCE EVALUATION

To further validate the effectiveness of the Stable Distribution Naive Bayes Classifier, we conducted additional experiments on several publicly available datasets: Wine, Social Network Ads, Diabetes, Electrical Grid Stability Simulated Data, and Vehicle Silhouettes. These datasets represent a diverse range of data distributions and classification challenges, testing the robustness and generalizability of the classifier.

6.1. Summary of Results

The results, summarized in Table 2, demonstrate that the Stable Distribution Naive Bayes Classifier outperforms the Gaussian Naive Bayes Classifier across these datasets.

Table 2. Comparison of Stable and Normal Naive Bayes Classifier Accuracy Across Datasets

Dataset	Stable Accuracy	Normal Accuracy
Wine	0.98	0.98
Social Network Ads	0.86	0.85
Diabetes	0.74	0.73
Electrical Grid Stability Simulated Data	0.84	0.84
Vehicle Silhouettes	0.66	0.58

6.2. Analysis of Key Findings

The **Wine Dataset** [12] features well-structured class separations with relatively low variability. The *Stable Naive Bayes Classifier* achieved an average accuracy of **98.46%**, slightly outperforming the *Gaussian model's 97.69%*. This result highlights the Stable model's ability to maintain high accuracy even in datasets with a simple structure.

The **Social Network AdsDataset** [13] involves userbehavior classification, where feature distributions often exhibit significant class overlap. The *Stable Naive Bayes Classifier* achieved **86.25%** accuracy, surpassing the *Gaussian model's 85.00%*. The results demonstrate the Stable model's advantage in capturing subtle distinctions in overlapping data distributions.

The **Diabetes Dataset** [14] presents a classification challenge due to skewed and heavy-tailed feature distributions. The *Stable Naive Bayes Classifier* achieved **73.59%** accuracy, showing a marginal improvement over the *Gaussian model's 72.73%*. The results underscore the Stable model's ability to handle complex, non-Gaussian distributions effectively.

The **Electrical Grid Stability Simulated Dataset** [15] consists of well-balanced feature distributions designed to assess system stability. Both the *Stable and Gaussian Naive Bayes models* achieved an accuracy of **83.93%**, indicating that in datasets with minimal skewness and heavy tails, the Stable model remains competitively accurate without a distinct advantage.

The **Vehicle Silhouettes Dataset** [16] is characterized by high variability and complex feature distributions, making classification particularly challenging. The *Stable Naive Bayes Classifier*

significantly outperformed the *Gaussian model, achieving 65.64% accuracy compared to 57.67%*. The dataset contains features with pronounced skewness and long tails, where the Stable model's robustness in handling non-Gaussian distributions contributes to its superior performance.

6.3. Implications

These findings confirm the Stable Distribution Naive Bayes Classifier's ability to generalize across diverse datasets, particularly in cases where the data deviates from the normality assumption. Its enhanced performance on datasets with heavy tails or skewness further underscores its utility as a robust and versatile classification tool for real-world applications.

7. CONCLUSION

In this paper, we have demonstrated howto use a Stable Distribution in aNaive Bayes Classifier by integrating the "rstable" R package functions into Python. This package is more specialized than packages currently available in Python for stable distributions. The presented approach shows how such integration can be accomplished. The Stable Distribution in a Naive Bayes Classifier extends the standard naive Bayesian classifier and could result in higher accuracy in predicting complicated and various real-world datasets.

DECLARATIONS

Conflict of Interest: We declare that there are no conflicts of interest regarding the publication of this paper.

Author Contributions: All authors contributed equally to the effort.

Funding: This research was conducted without any external funding. All aspects of the study, including design, data collection, analysis, and interpretation, were carried out using the resources available within the authors' institution.

Data Availability (including Appendices): All the relevant data, Python code for analysis, detailed annual tables and graphs are available via: https://drive.google.com/file/d/1Ml6v8WHuk_RHFV3yat5mOnGacymZKgl_ /view?usp=sharing

REFERENCES

[1] R. O. Duda and P. E. Hart, Pattern Classification and Scene Analysis. Wiley, New York, NY, USA, 1973.

[2] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," in Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI), Menlo Park, CA, USA: AAAI Press, 1992, pp. 223–228.

[3] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., Springer, New York, NY, USA, 2009.

[4] G. Samorodnitsky and M. S. Taqqu, Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance. Chapman and Hall/CRC, New York, NY, USA, 1994.

[5] J. P. Nolan, Stable Distributions - Models for Heavy Tailed Data. Birkhauser, Basel, Switzerland, 2013.¨

[6] Wikipediacontributors, "Stabledistribution,"Wikipedia, TheFreeEncyclopedia, 2024.[Online].Available: https://en.wikipedia.org/wiki/Stable_distribution. Accessed: 2024-02-25.

[7] Wikipedia contributors, "Naive Bayes classifier," Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Naive_Bayes_classifier. Accessed: 2024-0225.

[8] J. P. Nolan, ”Stable distributions: Models for heavy tailed data,” Math/Stat Department, American University, 2014. [Online]. Available: https://prac.im.pwr.edu.pl/˜burnecki/Materials/ chap1.pdf. Accessed: 2024-01-25.
[9] J. P. Nolan, ”Multivariate elliptically contoured stable distributions: theory and estimation,” Computational Statistics, vol. 28, no. 5, pp. 2067–2089, 2013. doi:10.1007/s00180-013-0396-7.
[10] K. P. Murphy, Machine Learning: A Probabilistic Perspective. MIT Press, 2012.
[11] Wikipedia contributors, ”Mixture model,” Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Mixture_model#Gaussian_mixture_model. Accessed: 2024-08-14.
[12] UCI Machine Learning Repository, ”Wine Dataset,” [Online]. Available: https://archive.ics.uci.edu/dataset/109/wine. Accessed: 2024-02-25.
[13] Kaggle contributors, ”Social Network Ads Dataset,” [Online]. Available: https://www.kaggle.com/datasets/akram24/social-network-ads. Accessed: 2024-02-25.
[14] Kaggle contributors, ”Diabetes Dataset,” [Online]. Available: https://www.kaggle.com/datasets/mathchi/diabetes-data-set. Accessed: 2024-02-25.
[15] Kaggle contributors, ”Electrical Grid Stability Simulated Data,” [Online]. Available: https://www.kaggle.com/datasets/sowlarn/ucis-electrical-grid-stability-simulated-data. Accessed: 2024-02-25.
[16] Kaggle contributors, ”Vehicle Silhouettes Dataset,” [Online]. Available: https://www.kaggle.com/datasets/pritech/vehicle-silhouettes. Accessed: 2024-02-25.

APPENDIX

```
class StableNaiveBayes:
    def __init__(self):
        self.params = {}
        self.class_prior = {}

    def fit(self, X, Y):
        self.classes = np.unique(Y)
        for cls in self.classes:
            X_cls = X[Y == cls]
            param_list = [fit_using_stable(X_cls[:, i]) for i in range(X.shape[1])]
            self.params[cls] = param_list
            self.class_prior[cls] = float(len(X_cls)) / len(X)

    def predict(self, X):
        preds = []
        for x in X:
            probs = []
            for cls in self.classes:
                prob = np.log(self.class_prior[cls])
                for i, val in enumerate(x):
                    alpha, beta, gamma, delta = self.params[cls][i]
                    stable_prob = predict_using_stable(val, alpha, beta, gamma,
                        ↪ delta)
                    prob += np.log(stable_prob)
                probs.append(prob)
            preds.append(self.classes[np.argmax(probs)])
        return np.array(preds)
```

```
class NormalNaiveBayes:
def __init__(self): self.params = {} self.class_prior = {}
def fit(self, X, Y):
self.classes = np.unique(Y) for cls in self.classes: X_cls = X[Y == cls] self.params[cls] =
[norm.fit(X_cls[:, i]) for i in range(X.shape[1])] self.class_prior[cls] = float(len(X_cls)) / len(X)
def predict(self, X):
preds = [] for x in X: probs = []

for cls in self.classes:
prob = np.log(self.class_prior[cls]) for i, val in enumerate(x):
loc, scale = self.params[cls][i] prob += np.log(norm.pdf(val, loc, scale))
probs.append(prob)
preds.append(self.classes[np.argmax(probs)]) return np.array(preds)
```

The ”StableNaiveBayes” and ”NormalNaiveBayes” classes demonstrate the Naive Bayes framework’s flexibility and simplicity, each tailored to different data distribution assumptions. ”NormalNaiveBayes” uses normal distribution parameters for prediction, suitable for symmetric data, while ”StableNaïveBayes” is designed for data with skewness or heavy tails, employing stable distribution parameters. Despite these methodological differences, bothclassesmaintainaneasy-to-followstructurewithmethodsforlearningfrom data (fit) and making predictions (predict). Their parallel design underscores the Naive Bayes classifier’s adaptability to various data types, making advanced statistical modeling accessible and straightforward.

We also need to write an additional custom calling R functions within Python. First, we need to import the crucial library ”rpy2” to call R functions in Python. Then the ”fit calling R” function we define converts data from Python to R vectors and passes them to ”stable.fit” function of R to fit the stable distribution. Similarly, the ”predict calling_R” function also uses the ”dstable” function to calculate the probability density of the stable distribution given the parameters. In this way, we have achieved interaction between Python and R.

```
def fit_calling_R(y, method_num):
    y_R = FloatVector(y)
    robjects.globalenv["x"] = y_R
    r_code = """
    library(stable)
    ret <- stable.fit(x, method = {method_num})
    """
    robjects.r(r_code)
    alpha, beta, gamma, delta = [float(robjects.r(f'ret["{param}"]')[0]) for
        ↪ param in ["alpha", "beta", "gamma", "delta"]]
    return alpha, beta, gamma, delta
```

```
def predict_calling_R(y, alpha, beta, gamma, delta):
    robjects.globalenv["x"] = y
    robjects.globalenv.update({"alpha": alpha, "beta": beta, "gamma": gamma, "
        ↪ delta": delta})
    r_code = """
    library(stable)
    prob <- dstable(x, alpha=alpha, beta=beta, gamma=gamma, delta=delta)
    """
    robjects.r(r_code)
    prob = np.array(robjects.r('prob'))
    return prob
```