ABSTRACT
Learning from limited and imbalanced data is a challenging problem in the Artificial Intelligence community. Real-time scenarios demand decision-making from rare events wherein the data are typically imbalanced. These situations commonly arise in medical applications, cybersecurity, catastrophic predictions etc. This motivates development of learning algorithms capable of learning from imbalanced data. Human brain effortlessly learns from imbalanced data. Inspired by the chaotic neuronal firing in the human brain, a novel learning algorithm namely Neurochaos Learning (NL) was recently proposed. NL is categorized in three blocks: Feature Transformation, Neurochaos Feature Extraction (CFX), and Classification. In this work, the efficacy of neurochaos feature transformation and extraction for classification in imbalanced learning is studied. We propose a unique combination of neurochaos based feature transformation and extraction with traditional ML algorithms. The explored datasets in this study revolve around medical diagnosis, banknote fraud detection, environmental applications and spoken-digit classification. In this study, experiments are performed in both high and low training sample regime. In the former, five out of nine datasets have shown a performance boost in terms of macro F1-score after using CFX features. The highest performance boost obtained is 25.97% for Statlog (Heart) dataset using CFX+Decision Tree. In the low training sample regime (from just one to nine training samples per class), the highest performance boost of 144.38% is obtained for Haberman’s Survival dataset using CFX+Random Forest. NL offers enormous flexibility of combining CFX with any ML classifier to boost its performance, especially for learning tasks with limited and imbalanced data.

1 Introduction
Technological advancements have made a paradigm shift in the evolution of science. A driving force behind this shift is the high storage and computational capacity available in this era. This has given rise to computational techniques for analysis and pattern discovery from data, popularly known as data-driven science. Data can be structured or
Another common pre-processing technique involves feature selection and feature extraction. Feature selection provides AI have seen a remarkable progress in recent past. On the other hand, these approaches are limited when it comes to non-parametric methods like SVM, especially in the low training sample regime. Recently, has demonstrated Coronavirus Genome classification (NL), proposed in [17, 18]. NL draws its inspiration from the chaos-based-features in [17, 21]. These features are passed to a classifier for decision making. For the first time, in [17], the authors employ the rich properties of chaos in learning algorithms for classification. Also, a performance boost in classification in the few-shot learning regime has been shown using chaos-based-features in [17][21]. The effectiveness of NL is shown in the classification of various datasets such as MNIST, Iris, Exoplanet [17], Coronavirus Genome classification [21], especially in the low training sample regime. Recently, [22] has demonstrated the efficacy of NL in the classification and preservation of cause-effect for 1D coupled chaotic maps and coupled AR processes.

There is a need to rigorously test NL, and in particular the nonlinear chaotic transformation of features, to more datasets in the context of imbalanced learning. This study addresses this gap. We combine NL features with classical machine learning algorithms such as Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Support Vector Machine (SVM), k-Nearest Neighbors (k-NN) and Gaussian Naïve Bayes (GNB). A comparison of NL: chaos-based-hybrid ML architectures with stand-alone ML algorithms is brought out for the following balanced and imbalanced datasets: Iris, MNIST, Iris, Exoplanet
**NEURCHAOS FEATURE TRANSFORMATION AND CLASSIFICATION FOR IMBALANCED LEARNING**

*Ionosphere, Wine, Bank Note Authentication, Haberman’s Survival, Breast Cancer Wisconsin, Statlog (Heart), Seeds, Free Spoken Digit Dataset.* A detailed analysis of this comparative study (NL: chaos-based-hybrid ML vs. stand-alone ML) is carried out for both the high and low training sample regime. Learning from limited data/imbalanced data is a challenging problem in the ML community. This research highlights the performance comparison of NL and classical ML algorithms in learning from limited training instances.

The organization of this paper is as follows: section 2 explains the proposed architecture and methods being investigated in this paper. The description of all datasets used for the experiments is provided in section 3. The experiments and their corresponding results are available in section 4. A detailed discussion on the inferences is provided in section 5. The concluding remarks of this study are in section 6.

### 2 Proposed Method

![Figure 1: The three steps proposed in this study. (a) feature transformation, (b) neurochaos feature extraction, and (c) classification. The classifier chosen could either be ChaosNet or any other ML classifier.](image)

In this study, a modified form of the recently proposed Neurochaos Learning architecture [18] is put forward. This modified NL architecture is depicted in Figure 1. NL has mainly three blocks - (a) feature transformation, (b) neurochaos feature extraction and (c) classification. A detailed description of each block is given below.

- **Input:** Input is the first and most significant step of any learning process. It contains input attributes obtained from the dataset $(x_1, x_2, \ldots, x_n)$. These input attributes (after suitable normalization) are further passed to the feature transformation block.

- **Feature Transformation:** The feature transformation block consists of an input layer of 1D Generalized Lüroth Series (GLS) neurons. The number of GLS neurons $(G_1, G_2, \ldots, G_n)$ in the input layer is equal to the number of input attributes $(n)$ in the dataset. Each neuron $G_1, G_2, \ldots, G_n$ has an initial neural activity of $q$ units. Upon arrival of the input attributes (also known as stimuli) $x_1, x_2, \ldots, x_n$, each of the 1D GLS neurons $(G_1, G_2, \ldots, G_n)$ starts independently firing with an initial neural activity of $q$ units. The neural trace of these chaotic neurons halts when it reaches the $\epsilon$ neighbourhood of the stimulus (at which point we say that it has successfully recognized the stimulus). The halting of the neural trace is mathematically guaranteed by the topological transitivity property of chaos [17]. The transformed input attributes are further passed to the neurochaos feature extraction block.

- **Neurochaos Feature Extraction:** The following features are extracted from the chaotic neural trace:
1. **Firing time** \((N)\): The time taken by the chaotic neural trace to recognize the stimulus \([23]\).

2. **Firing rate** \((R)\): Fraction of time for which the chaotic neural trace exceeds the discrimination threshold \(b\) so as to recognize the stimulus \([23]\).

3. **Energy** \((E)\): A chaotic neural trace \(z(t)\) with a firing time \(N\) has an energy \((E)\) defined as:

   \[
   E = \sum_{t=1}^{N} z(t)^2.
   \]

4. **Entropy** \((H)\): The entropy of a chaotic neural trace \(z(t)\) is computed using the symbolic sequence \(SS(t)\) of \(z(t)\). \(SS(t)\) is defined as follows:

   \[
   SS(t_i) = \begin{cases} 
   0, & z(t_i) < b, \\
   1, & b \leq z(t_i) < 1,
   \end{cases}
   \]

   where \(i = 1\) to \(N\) (firing time). From \(SS(t)\), Shannon Entropy \(H(SS)\) is computed as follows:

   \[
   H(SS) = -\sum_{i=1}^{2} p_i \log_2(p_i) \text{ bits},
   \]

   where, \(p_1\) and \(p_2\) are the probabilities of occurrence of symbols 0 and 1 in \(SS(t)\) respectively.

An input stimulus \(x_k\) of a data instance visiting the \(k\)-th GLS neuron \((G_k)\) is transformed to a 4D vector \([N_{x_k}, R_{x_k}, E_{x_k}, H_{x_k}]\). The CFX feature space contains a collection of all the 4D vectors after feature transformation. These chaos based features are passed to the third block of the NL architecture i.e, classification.

- **Classification**: There are mainly two kinds of NL architecture: (a) ChaosNet, (b) ChaosFEX (CFX) + ML. ChaosNet architecture computes the mean representation vector for each class. The mean representation vector of class-\(k\) contains the mean firing time, mean firing rate, mean energy and mean entropy of the \(k\)-th class. ChaosNet uses a simplistic decision rule, namely, the cosine similarity of test data instances with the mean representation vectors. The test data instance is assigned a label \(l\), if the cosine similarity of that test data instance with \(l\)-th mean representation vector is the highest.

Alternatively, we have the flexibility of choosing any other ML classifier instead of ChaosNet. In this kind of NL architecture, the ChaosFEX features are fed directly to the ML classifier (CFX+ML). In this study, we have tested with the following ML algorithms -- Decision Tree (DT), Random Forest (RF), AdaBoost (AB), Support Vector Machine (SVM), \(k\)-Nearest Neighbors (\(k\)-NN) and Gaussian Naive Bayes (GNB). CFX+ML is a hybrid NL architecture that combines the best of chaos and machine learning.

- **Output**: The output obtained from the classification block serves as output for that respective NL architecture. On using ChaosNet in the classification block, the output is an outcome of classification based on the mean representation vectors. The Machine Learning implementation in the classification block produces an output dependent on the choice of the ML classifier.

In \([21]\), it has been shown that ChaosNet satisfies the *Universal Approximation Theorem* with a bound on the number of chaotic neurons to approximate the finite support discrete time function. This is due to the uncountably infinite number of dense orbits and the *topological transitivity* property of chaos. Another important feature of ChaosNet and CFX+ML is the natural presence of Stochastic Resonance (noise enhanced signal processing) \([18]\) found in the architecture. An optimal performance of ChaosNet and CFX+ML is obtained for an intermediate value of noise intensity (\(\epsilon\)). This has been thoroughly shown in \([18]\).

### 3 Dataset Description

The ChaosFEX (CFX) feature extraction algorithm requires normalization of the dataset and numeric codes for labels. Therefore, to maintain uniformity, all datasets are normalized\(^1\) for both stand-alone algorithms and their integration with CFX. The labels are renamed to begin from zero in each dataset to ensure compatibility with CFX feature extraction. The rules followed for the numeric coding for the labels of all the datasets are provided in section 8 (Table 16 - 24).

\[^1\] \(X_{\text{norm}} = \frac{X - \min(X)}{\max(X) - \min(X)}\).
3.1 Iris

Iris [24, 25] aids classification of three iris plant variants: Iris Setosa, Iris Versicolour, and Iris Virginica. There are 150 data instances in this dataset with four attributes in each data instance: sepal length, sepal width, petal length, and petal width. All attributes are in cms. The specified class distribution provided in Table 1 is in the following order: (Setosa, Versicolour, Virginica).

3.2 Ionosphere

The Ionosphere [26, 27] dataset enables a binary classification problem. The classes represent the status of returning a radar signal from the Ionosphere. Label ‘g’ (Good) denotes the return of the radar signal, and label ‘b’ (Bad) indicates no trace of return of the radar signal. The goal of this experiment is to identify the structure of the Ionosphere using radar signals. This dataset has 351 data instances and 34 attributes. The specified class distribution provided in Table 1 is as follows: (Bad, Good).

3.3 Wine

Wine [26, 28] dataset aims to identify the origin of different wines using chemical analysis. The classes are labeled ‘1’, ‘2’, and ‘3’. It has 178 data instances and 13 attributes ranging from alcohol, malic acid to hue and proline for the collected samples. The specified class distribution provided in Table 1 is as follows: (1, 2, 3).

3.4 Bank Note Authentication

Bank-note Authentication [26, 29] is a binary classification dataset. The classes involve Genuine and Forgery. A Genuine class refers to an authentic banknote denoted by ‘0’, while a Forgery class refers to a forged banknote denoted by ‘1’. The obtained dataset is from images of banknotes belonging to both classes, taken from an industrial camera. It contains 1372 total data instances. The dataset has four attributes retrieved from the images using wavelet transformation. The specified class distribution provided in Table 1 is as follows: (Genuine, Forgery).

3.5 Haberman’s Survival

Haberman’s Survival [26, 30] is a compilation of sections of a study investigating the lifespan of a patient after undergoing a breast cancer surgery. This is a binary classification problem and the dataset provides information for the prediction of the survival of patients beyond five years. Class ‘1’ denotes survival of the patient for five years or longer after the surgery. Class ‘2’ denotes the death of a patient within five years of the surgery. It contains 306 total data instances and three attributes. The specified class distribution provided in Table 1 is as follows: (1, 2).

3.6 Breast Cancer Wisconsin

Breast Cancer Wisconsin [26, 31] dataset deals with the classification of the intensity of the breast cancer. Class ‘M’ refers to a malignant level of infection and class ‘B’ refers to a benign level of infection. It contains a total of 569 data instances and 31 attributes such as radius, perimeter, texture, smoothness, etc. for each cell nucleus. The specified class distribution provided in Table 1 is as follows: (Malignant - M, Benign - B).

3.7 Statlog (Heart)

Statlog (Heart) [26] enables differentiation between presence and absence of a heart disease in a patient. Class ‘1’ denotes the absence while class ‘2’ denotes the presence of a heart disease. It contains 270 total data instances and 13 attributes including resting blood pressure, chest pain type, exercise induced angina and so on. The specified class distribution provided in Table 1 is as follows: (Absence , Presence).

3.8 Seeds

Seeds [26] dataset examines three classes of wheat: Kama, Rosa and Canadian using soft X-ray on the wheat kernels to retrieve relevant properties. It contains 210 total data instances and seven attributes namely compactness, length, width etc. of each wheat kernel. The specified class distribution provided in Table 1 is as follows: (Kama, Rosa, Canadian).
3.9 Free Spoken Digit Dataset

The Free Spoken Digit Dataset \(^{[32]}\) is a time-series dataset comprising recordings of six speakers. Each speaker recites numbers from one to nine. For each number, every speaker makes 50 recordings. The speaker chosen for all experiments is Jackson. The dataset undergoes preprocessing using a Fast Fourier Transform (FFT) technique. The dataset for speaker Jackson has 500 data instances, and only instances above a threshold of 3000 samples are considered to tackle the varying data length through the dataset. In these data instances, only the first 3005 data samples are examined. Finally, 480 data instances are filtered to feed into the algorithm. The specified class distribution provided in Table 1 is as follows: (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).

| Dataset              | Classes | Features | Training samples /class | Testing samples /class | Imbalanced data (Y/N) |
|----------------------|---------|----------|--------------------------|------------------------|-----------------------|
| Iris                 | 3       | 4        | (40, 41, 39)             | (10, 9, 11)            | N                     |
| Ionosphere           | 2       | 34       | (98, 182)                | (28, 43)               | Y                     |
| Wine                 | 3       | 13       | (45, 57, 40)             | (14, 14, 8)            | Y                     |
| Bank Note Authentication | 2     | 4        | (614, 483)               | (148, 127)             | Y                     |
| Haberman’s Survival  | 2       | 3        | (181, 63)                | (44, 18)               | Y                     |
| Breast Cancer Wisconsin | 2    | 31       | (169, 286)               | (43, 71)               | Y                     |
| Statlog (Heart)      | 2       | 13       | (117, 99)                | (33, 21)               | Y                     |
| Seeds                | 3       | 7        | (59, 56, 53)             | (11, 14, 17)           | N                     |
| FSDD                 | 10      | 3005     | (40, 35, 44, 42, 38, 34, 37, 44, 33, 37) | (10, 15, 6, 8, 8, 7, 13, 6, 10, 13) | Y                     |

4 Experiments & Results

In this section, the authors evaluate the efficacy of ChaosFEX (CFX) feature engineering on various classification tasks. For this, hybrid models are developed by combining CFX features extracted from the input data with the classical ML algorithms such as Decision Tree, Random Forest, AdaBoost, Support Vector Machine, \(k\)-Nearest Neighbors and Gaussian Naive Bayes. The experiments are performed for both low and high training sample regime. The train-test distribution (80% - 20%) for each dataset in the high training sample regime is available in Table 1. In low training sample regime, 150 random trials for training with 1, 2, \ldots, 9 data instances per class are considered. The trends of all algorithms in the stand-alone form and their implementation using CFX features are conveyed in this section. For all experiments in the paper, the following software: Python 3.8 and scikit-learn \(^{[24]}\), CFX \(^{[17]}\) are used.

4.1 Hyperparameter Tuning

Every ML algorithm has a set of optimal hyperparameters to be found by hyperparameter tuning. The hyperparameters for all the algorithms with respect to each dataset were tuned using five-fold cross-validation. Table 2 provides the set of hyperparameters tuned for all algorithms in this research.

In the case of CFX+ML for Iris, Ionosphere and Wine, both CFX \((q, b, \epsilon)\) and ML hyperparameters were tuned. For the remaining datasets, the hyperparameters tuned for ChaosNet \((q, b, \epsilon)\) were retained and only the ML hyperparameters were tuned.

4.2 Performance Metric

The most common performance metric used in ML is \textit{Accuracy} \(^{[33]}\). In the case of imbalanced datasets, this metric will lead to misinterpretation of results. Thus, the performance metric used for all experiments in this study is \textit{Macro F1-Score}. This metric is computed from the confusion matrix. Figure 2 shows a confusion matrix for a binary classification problem. Prediction values represent the predictions made by the classifier. Actual values stand for the
Table 2: References for the tuned hyperparameters for all algorithms. Owing to space constraints, most of the tables are moved to the Supplementary Information section (section 8).

| Algorithm         | Acronyms | Hyperparameters Tuned           | Reference                      |
|-------------------|----------|---------------------------------|--------------------------------|
| ChaosNet          | -        | $q, b, \epsilon$                | Tables 4, 5, 6, 7, 8, 9, 10, 11, 12 |
| Decision Tree     | DT       | $\text{min\_samples\_leaf}$, $\text{max\_depth}$, $\text{ccp\_alpha}$ | Tables 25 and 26 in Supplementary Information section |
| Random Forest     | RF       | $n\_estimators$, $\text{max\_depth}$ | Tables 27 and 28 in Supplementary Information section |
| AdaBoost          | AB       | $n\_estimators$                 | Table 29 in Supplementary Information section |
| Support Vector Machine | SVM   | $C, \text{kernel}$             | Table 30 in Supplementary Information section |
| $k$-Nearest Neighbors | $k$-NN, KNN | $k$                           | Table 31 in Supplementary Information section |
| Gaussian Naive Bayes | GNB    | -                               | -                             |

true/ target values for the predictions being made by the classifier. They both have two categories: Positive and Negative. A true positive stands for a target value equal to positive, also correctly classified by the classifier as positive. True negatives are target values equal to negative, also correctly classified by the classifier as negative. False negative implies a target value equal to positive which was incorrectly classified as negative by the classifier. Similarly, false positive means a target value equal to negative that was incorrectly classified as positive by the classifier.

Figure 2: Pictorial representation of a confusion matrix corresponding to a binary classification problem.

F1-Score depends on two parameters:

1. Precision - The ratio of the number positives correctly classified as positive by the algorithm to the total number of instances classified as positive, given by -

   \[
   \text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}
   \]  

   \[ (4) \]

2. Recall - The ratio of the number positives correctly classified as positive by the algorithm to the total number of actual positives, given by -

   \[
   \text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}
   \]

   \[ (5) \]

Thus, F1-Score which is the harmonic mean of Precision and Recall is given by,

\[
F1\text{-Score}(F1) = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}
\]

\[ (6) \]
Macro F1-Score is obtained by averaging the F1 scores for all classes in the dataset, given by

$$\text{Macro F1-Score} = \frac{F_{1_{\text{Class-1}}} + F_{1_{\text{Class-2}}} + \ldots + F_{1_{\text{Class-n}}}}{n},$$

(7)

where, $n$ stands for the number of distinct classes in the dataset. F1-score of $\text{Class}_k$ considers the instances of $\text{Class}_k$ as positive and all remaining instances of classes as negative. Thus, the task is transformed to a binary classification problem. All instances of $\text{Class}_k$ correctly classified as positive or negative by the classifier are termed as true positives or true negatives respectively. All instances that are incorrectly classified by the classifier are placed under the false positive and false negative category accordingly. Hence the associated equations of precision and recall for this example are given by,

$$\text{Precision}_{\text{Class}_k} (P_k) = \frac{\text{True Positive}_{\text{Class}_k}}{\text{True Positive}_{\text{Class}_k} + \text{False Positive}_{\text{Class}_k}},$$

(8)

$$\text{Recall}_{\text{Class}_k} (R_k) = \frac{\text{True Positive}_{\text{Class}_k}}{\text{True Positive}_{\text{Class}_k} + \text{False Negative}_{\text{Class}_k}}.$$  

(9)

The F1-score for $\text{Class}_k$ is computed by:

$$F_{1\text{-Score}_{\text{Class}_k}} = \frac{2 \cdot P_k \cdot R_k}{P_k + R_k}.$$  

(10)

Similarly, the F1-scores for all classes in the classification problem are computed and applied in Eq(7).

### 4.3 Performance of ChaosNet

ChaosNet has three hyperparameters – initial neural activity ($q$), discrimination threshold ($b$), and noise intensity ($\epsilon$) [18]. Specifics of the same are provided in Table 2.

| Dataset                          | Macro-F1 Score (Test Data) |
|---------------------------------|-----------------------------|
| Iris                            | 1.000                       |
| Ionosphere                      | 0.860                       |
| Wine                            | 0.976                       |
| Bank Note Authentication         | 0.845                       |
| Haberman’s Survival             | 0.560                       |
| Breast Cancer Wisconsin         | 0.927                       |
| Statlog (Heart)                 | 0.738                       |
| Seeds                           | 0.845                       |
| FSDD                            | 0.897                       |

### 4.4 Comparative Performance Evaluation

In this section, the authors represent the results in two formats (a) bar graph and (b) line graph. The comparative results for ChaosNet, CFX+ML and stand-alone ML in the high training sample regime are depicted using bar graph. All values plotted in the bar graphs for each dataset are provided in the Supplementary Information section (section 8) from Table[12] - [17]. On the other hand, the line graph depicts the comparative performance of ChaosNet, CFX+ML and stand-alone ML in the low training sample regime.

#### 4.4.1 Results for Iris

The tuned hyperparameters used and all experiment results for the Iris dataset are available in Table[4] and Figure[5] respectively.
Table 4: Hyperparameters used for Iris dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.141       |
| b              | 0.499       |
| ϵ              | 0.147       |

Figure 3: Iris. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the low training sample regime. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.

4.4.2 Results for Ionosphere

The tuned hyperparameters used and all experiment results for the Ionosphere dataset are available in Table 5 and Figure 4, respectively.

Table 5: Hyperparameters used for Ionosphere dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.680       |
| b              | 0.969       |
| ϵ              | 0.164       |
Figure 4: Ionosphere. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the low training sample regime. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.

4.4.3 Results for Wine

The tuned hyperparameters used and all experiment results for the Wine dataset are available in Table 6 and Figure 5 respectively.

Table 6: Hyperparameters used for Wine dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.790       |
| b              | 0.499       |
| ε              | 0.262       |
4.4.4 Results for Bank Note Authentication

The tuned hyperparameters used and all experiment results for the Bank Note Authentication dataset are available in Table 7 and Figure 6 respectively.

Table 7: Hyperparameters used for Bank Note Authentication dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.080       |
| b              | 0.250       |
| c              | 0.233       |
4.4.5 Results for Haberman’s Survival

The tuned hyperparameters used and all experiment results for the Haberman’s Survival dataset are available in Table 8 and Figure 7 respectively.

Table 8: Hyperparameters used for Haberman’s Survival dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.810       |
| b              | 0.140       |
| c              | 0.003       |
4.4.6 Results for Breast Cancer Wisconsin

The tuned hyperparameters used and all experiment results for the Breast Cancer Wisconsin dataset are available in Table 9 and Figure 8 respectively.

Table 9: Hyperparameters used for Breast Cancer Wisconsin dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.930       |
| b              | 0.490       |
| c              | 0.159       |

Figure 7: Haberman’s Survival. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the low training sample regime. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.
4.4.7 Results for Statlog (Heart)

The tuned hyperparameters used and all experiment results for the Statlog (Heart) dataset are available in Table 10 and Figure 9 respectively.

Table 10: Hyperparameters used for Statlog (Heart) dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.080       |
| b              | 0.060       |
| c              | 0.170       |
4.4.8 Results for Seeds

The tuned hyperparameters used and all experiment results for the Seeds dataset are available in Table 11 and Figure 10 respectively.

Table 11: Hyperparameters used for Seeds dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.020       |
| b              | 0.070       |
| $\epsilon$    | 0.238       |
4.4.9 Results for Free Spoken Digit Dataset (FSDD)

The tuned hyperparameters used and all experiment results for the FSDD dataset are available in Table 12 and Figure 11 respectively.

Table 12: Hyperparameters used for FSDD dataset for high and low training sample regime experiments.

| Hyperparameter | Tuned Value |
|----------------|-------------|
| q              | 0.340       |
| b              | 0.499       |
| c              | 0.178       |

Figure 10: Seeds. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the low training sample regime. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.
5 Discussion

5.1 High Training Sample Regime

The overall comparative performance of different algorithms is provided in Table 13. In the high training sample regime, the efficacy of using CFX features is evident from Table 13 for Iris, Ionosphere, Haberman’s Survival, Statlog (Heart) and FSDD. While Iris is a balanced dataset, Ionosphere, Haberman’s Survival and Statlog (Heart) are imbalanced. Through this, a versatility in the algorithm’s ability to perform with both balanced and imbalanced datasets can be established.

The performance boost after using CFX features is calculated as follows:

\[
Boost = \left( \frac{F_{1_{\text{CFX+ML}}} - F_{1_{\text{ML}}}}{F_{1_{\text{ML}}}} \right) \cdot 100\% ,
\]

where \( F_{1_{\text{ML}}} \) and \( F_{1_{\text{CFX+ML}}} \) refers to the macro F1-score for the stand-alone ML algorithm and the hybrid NL (CFX+ML) algorithm respectively.

5.2 Low Training Sample Regime

In the low training sample regime, the comparative performance of stand-alone and CFX+ML algorithms is depicted in Figure 11. It is observed that CFX+ML algorithms outperform stand-alone algorithms in terms of F1-score. The versatility in handling both balanced and imbalanced datasets is again demonstrated.

![Figure 11: Free Spoken Digit Dataset. (a) High training sample regime. (b) Comparative performance of stand-alone algorithms in the low training sample regime. (c) Comparative performance of CFX+ML algorithms in the low training sample regime.](image)
Table 13: Overall comparative performance of different algorithms in the high training sample regime. Percentages in parenthesis indicate the Boost computed using Eq[11].

| Dataset                  | ChaosNet (Macro F1-Score) | Best Algorithm                 | Best Macro F1-Score |
|--------------------------|--------------------------|--------------------------------|---------------------|
| Iris                     | 1.0                      | ChaosNet, RF, KNN CFx+DT (3.41%), CFx+RF (0%) | 1.0                |
| Ionosphere               | 0.860                    | CFx+KNN (14.37%)               | 0.939               |
| Wine                     | 0.976                    | GNB                            | 1.0                 |
| Bank Note Authentication  | 0.845                    | SVM, KNN                       | 0.993               |
| Haberman’s Survival      | 0.560                    | CFx+AB (20.59%)                | 0.609               |
| Breast Cancer Wisconsin  | 0.927                    | k-NN                           | 0.954               |
| Statlog (Heart)          | 0.738                    | CFx+DT (25.97%)                | 0.878               |
| Seeds                    | 0.845                    | KNN                            | 0.924               |
| FSDD                     | 0.897                    | CFx+SVM (2.73%)                | 0.978               |

5.2 Low Training Sample Regime

Table 14 demonstrates the overall performance of all datasets in the low training sample regime after employing the CFX features. A ✓ refers to a performance boost of ML algorithms after using CFX features. CFX features have shown an evident improvement in performance for Ionosphere, Haberman’s Survival, Statlog (Heart), Seeds and FSDD. 150 random trials of training in the low training sample regime with 1, 2, . . ., 9 samples per class are performed. The 150 random trials of training ensures the model does not overfit.

Table 14: Overall performance boost using CFX features in the low training sample regime. A ✓ refers to a performance boost of ML algorithms after using CFX features. We report (Minimum, Maximum) of Boost values only in these instances.

| Dataset                  | DT     | RF     | SVM    | KNN    | GNB    |
|--------------------------|--------|--------|--------|--------|--------|
| Iris                     | ✓      | ✓      | ✓      | ✓      | ✓      |
| (0.88%, 17.63%)          | (0.15%, 18.01%) | (0.83%, 10.71%) | (1.15%, 17.60%) | (0.80%, 17.60%) |
| Ionosphere               | ✓      | ✓      | ✓      | ✓      | ✓      |
| (0.0%, 21.50%)           | (8.43%, 144.38%) | (2.19%, 6.43%) | (3.16%, 8.85%) |
| Breast Cancer Wisconsin  | ✓      | ✓      | ✓      | ✓      | ✓      |
| (0.0%, 1.05%)            | (0.69%, 5.58%) |
| Statlog (Heart)          | ✓      | ✓      | ✓      | ✓      | ✓      |
| (0.0%, 9.08%)            | (0.01%, 0.84%) | (1.56%, 19.02%) | (0.04%, 0.43%) |
| Seeds                    | ✓      | ✓      | ✓      | ✓      | ✓      |
| (0.76%, 17.63%)          | (2.76%, 17.63%) |
5.3 Inferences based on consistency of algorithms across all datasets

The consistency of an algorithm can be inferred by evaluating the range of minimum and maximum macro F1-scores produced by it in the high training sample regime. Table 15 shows the ranges of macro F1-scores of different algorithms, measured across all nine datasets used in the research. The provided format for the range of macro F1-scores in Table 15 is [Minimum, Maximum]. ChaosNet ranks second in the least difference between the maximum and minimum macro F1-scores. This observation owes to the consistency and tolerance towards dataset diversity in ChaosNet. In algorithms such as AdaBoost (AB) and Support Vector Machine (SVM), a relatively low difference between F1-scores is realised after the usage of CFX features. Gaussian Naive Bayes (GNB) shows the least difference between F1-scores. Along datasets of different domains considered in this study, the performance of ChaosNet can be seen as comparable with GNB.

5.4 Limitations

With the current implementation of the ChaosFEX algorithm, computation of image datasets is a costly process. This may limit the use of NL architectures in practical situations involving images.

Furthermore, NL architectures have been based on certain assumptions. One assumption revolves around the separability of data. NL assumes that applying a nonlinear chaotic transformation will result in separable data suitable for classification, which may not be true in all cases. As it stands, the input layer of NL treats input attributes as independent of each other. It establishes no connection between the neurons for each input attribute. This limitation can be addressed by using coupled chaotic neurons in the input layer. Currently, we have not considered multi-layered NL (Deep-NL) which could significantly enhance performance (by careful choice of coupling between adjacent layers). Another limitation of the NL architectures is the lack of a principled approach to tune the best hyperparameters for a classification task. Currently, cross-validation experiments are used to tune the hyperparameters. The connection between the degree of chaos as measured by lyapunov exponent and learnability is also worth exploring for future research.

6 Conclusion

Decision making under the presence of rare events is a challenging problem in the ML community. This is because rare events have limited data instances, and this problem boils down to imbalanced learning. In this work, we have evaluated the effectiveness of ChaosFEX (CFX) feature transformation used in Neurochaos Learning (NL) architectures for imbalanced learning. Nine benchmark datasets were used in this study to bring out this evaluation. Seven out of nine datasets used are imbalanced (Refer to Table 1). This paper accomplishes a comparative study on the performance of NL architecture: ChaosNet and CFX+ML with classical Machine Learning (ML) algorithms. The obtained results reflect an evident performance boost in terms of macro F1-score after a nonlinear chaotic transformation (ChaosFEX or CFX features). Additionally, the efficacy of CFX features can be observed in five out of nine balanced and imbalanced datasets in the high training sample regime, with a boost ranging from 2.73% (Free Spoken Digit Dataset) to 25.97% (Statlog - Heart). In the low training sample regime, the integration of CFX features has boosted the performance of classical ML algorithms in five datasets, from a total of nine datasets. A maximum boost of 144.38% on the Haberman’s Survival dataset using CFX+RF is obtained. Refer to Table 13 and 14 for the detailed performance boost using CFX features. This is the first study thoroughly evaluating the performance of NL in the imbalanced learning scenario.
NL is a unique combination of chaos and noise-enhanced classification. The enormous flexibility of NL offers endless possibilities for development of novel NL: chaos-based-hybrid ML models that suit the application at hand. As new ML algorithms get invented, they can be readily combined in the NL framework. We forsee exciting combinations of CFX with DL and other ML algorithms in the future.

7 Acknowledgments

Deeksha Sethi is thankful to Saneesh Cleatus T, Associate Professor, BMS Institute of Technology and Management for enabling her with this research opportunity. Sethi dedicates this work to her family. Harikrishnan N. B. thanks “The University of Trans-Disciplinary Health Sciences and Technology (TDU)” for permitting this research as part of the PhD programme. The authors gratefully acknowledge the financial support of Tata Trusts. The authors acknowledge the computational facility supported by NIAS Consciousness Studies Programme.

Declarations

- Funding: The authors gratefully acknowledge the financial support of Tata Trusts.
- Conflicts of Interest/Competing interests: There is no conflict of interest or competing interests.
- Ethics approval: Not Applicable.
- Consent to participate: Not Applicable.
- Consent for publication: Not Applicable.
- Availability of data and materials: Not Applicable.
- Code availability: The codes used in this research are available in the following link: https://github.com/deeksha-sethi03/nl-imbalanced-learning.
- Authors’ contributions:  
  - Conceptualization: Harikrishnan NB  
  - Methodology: Harikrishnan NB  
  - Formal analysis and investigation: Deeksha Sethi, Harikrishnan NB, Nithin Nagaraj  
  - Code implementation: Deeksha Sethi (Template codes provided by Harikrishnan NB)  
  - Writing - original draft preparation: Deeksha Sethi, Harikrishnan NB  
  - Writing - review and editing: Deeksha Sethi, Harikrishnan NB, Nithin Nagaraj  
  - Funding acquisition: Nithin Nagaraj  
  - Resources: Consciousness Studies Programme, National Institute of Advanced Studies  
  - Supervision: Harikrishnan NB, Nithin Nagaraj

References

[1] Joel Lehman, Jeff Clune, and Sebastian Risi. An anarchy of methods: Current trends in how intelligence is abstracted in ai. IEEE Intelligent Systems, 29(6):56–62, 2014.
[2] Melanie Mitchell. Artificial intelligence: A guide for thinking humans. 2019.
[3] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. Learning from class-imbalanced data: Review of methods and applications. Expert Systems with Applications, 73:220–239, 2017.
[4] Maher Maalouf and Theodore B Trafalis. Robust weighted kernel logistic regression in imbalanced and rare events data. Computational Statistics & Data Analysis, 55(1):168–183, 2011.
[5] Marco Ciotti, Massimo Ciccozzi, Alessandro Terrinoni, Wen-Can Jiang, Cheng-Bin Wang, and Sergio Bernardini. The covid-19 pandemic. Critical reviews in clinical laboratory sciences, 57(6):365–388, 2020.
[6] Antoni Trilla, Guillem Trilla, and Carolyn Daer. The 1918 “Spanish Flu” in Spain. Clinical Infectious Diseases, 47(5):668–673, 09 2008.
[7] Craig Beaman, Ashley Barkworth, Toluwalope David Akande, Saqib Hakak, and Muhammad Khurram Khan. Ransomware: Recent advances, analysis, challenges and future research directions. Computers & Security, 111:102490, 2021.
Suvasini Panigrahi, Amlan Kundu, Shamik Sural, and Arun K Majumdar. Credit card fraud detection: A fusion approach using dempster–shafer theory and bayesian learning. *Information Fusion*, 10(4):354–363, 2009.

Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Muhammad Atif Tahir, Josef Kittler, Krystian Mikolajczyk, and Fei Yan. A multiple expert approach to the class imbalance problem using inverse random under sampling. In *International workshop on multiple classifier systems*, pages 82–91. Springer, 2009.

Ludmila I Kuncheva and William J Faithfull. Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80, 2013.

Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

Ludmila I Kuncheva and William J Faithfull. Pca feature extraction for change detection in multidimensional unlabeled data. *IEEE transactions on neural networks and learning systems*, 25(1):69–80, 2013.

OO Ogundile, AM Usman, OP Babalola, and DJJ Versfeld. Dynamic mode decomposition: A feature extraction technique based hidden markov model for detection of mysticetes’ vocalisations. *Ecological Informatics*, 63:101306, 2021.

Md Afzal Hossan, Sheeraz Memon, and Mark A Gregory. A novel approach for mfcc feature extraction. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–5. IEEE, 2010.

Yu-Xiong Wang and Yu-Jin Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE Transactions on knowledge and data engineering*, 25(6):1336–1353, 2012.

Vaijayanthi Nagarajan, Elizabeth Caroline Britto, and Senthivel Murugan Veeraputhiran. Feature extraction based on empirical mode decomposition for automatic mass classification of mammogram images. *Medicine in Novel Technology and Devices*, 1:100004, 2019.

Harikrishnan Nellippalil Balakrishnan, Aditi Kathpalia, Snehashu Saha, and Nithin Nagaraj. Chaosnet: A chaos based artificial neural network architecture for classification. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 29(11):113125, 2019.

NB Harikrishnan and Nithin Nagaraj. When noise meets chaos: Stochastic resonance in neurochaos learning. *Neural Networks*, 143:425–435, 2021.

Frederico AC Azevedo, Ludmila RB Carvalho, Lea T Grinberg, José Marcelo Farfel, Renata EL Ferretti, Renata EP Leite, Wilson Jacob Filho, Roberto Lent, and Suzana Herculano-Houzel. Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. *Journal of Comparative Neurology*, 513(5):532–541, 2009.

Harikrishnan NB, Pranay SY, and Nithin Nagaraj. A neurochaos learning architecture for genome classification. *arXiv preprint arXiv:2010.10995*, 2020.

Aditi Kathpalia, Nithin Nagaraj, et al. Cause-effect preservation and classification using neurochaos learning. *arXiv preprint arXiv:2201.12181*, 2022.

NB Harikrishnan and Nithin Nagaraj. Neurochaos inspired hybrid machine learning architecture for classification. In *2020 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. A. FISHER. The use of multiple measurements in taxonomic problems. *Annals of Eugenics, 7(2):179–188, 1936.*

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

Vincent G Sigillito, Simon P Wing, Larrie V Hutton, and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

Michele Forina, Riccardo Leardi, Armanino C, and Sergio Lanteri. PARVUS: An Extendable Package of Programs for Data Exploration. 01 1998.

Eugen Gillich and Volker Lohweg. Banknote authentication. 11 2010.

Shelby J Haberman. The analysis of residuals in cross-classified tables. *Biometrics*, pages 205–220, 1973.
[31] W Nick Street, William H Wolberg, and Olvi L Mangasarian. Nuclear feature extraction for breast tumor diagnosis. In *Biomedical image processing and biomedical visualization*, volume 1905, pages 861–870. SPIE, 1993.

[32] Zohar Jackson, César Souza, Jason Flaks, Yuxin Pan, Hereman Nicolas, and Adhish Thite. Jakobovski/free-spoken-digit-dataset: v1.0.8, August 2018.

[33] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.

[34] Jonathan B Dingwell. Lyapunov exponents. *Wiley encyclopedia of biomedical engineering*, 2006.
8 Supplementary Information

This is the supplementary information pertaining to the main manuscript. It contains the following – (1) description of datasets used in our study including the coding rule for the labels of different classes, (2) hyperparameter tuning details for each dataset and for each ML algorithm (Decision Tree, Random Forest, AdaBoost, SVM, \(k\)-NN) and NL algorithm (ChaosNet) used in the study, (3) the test data macro F1-scores for each algorithm in the high training sample regime for each dataset.

8.1 Dataset Description

8.1.1 Iris

Table 16: Iris: Rule followed for renaming of the class labels.

| Class Label       | Numeric Code | Number of Total Data Instances |
|-------------------|--------------|--------------------------------|
| Iris-Setosa       | 0            | 50                             |
| Iris-Versicolour  | 1            | 50                             |
| Iris-Virginica    | 2            | 50                             |

8.1.2 Ionosphere

Table 17: Ionosphere: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| b (Bad)     | 0            | 126                            |
| g (Good)    | 1            | 225                            |

8.1.3 Wine

Table 18: Wine: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| 1           | 0            | 59                             |
| 2           | 1            | 71                             |
| 3           | 2            | 48                             |
8.1.4 Bank Note Authentication

Table 19: Bank Note Authentication: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| 0 (Genuine) | 0            | 762                            |
| 1 (Forgery) | 1            | 610                            |

8.1.5 Haberman’s Survival

Table 20: Haberman’s Survival: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| 1 (< 5yrs) | 0            | 225                            |
| 2 (≥ 5yrs) | 1            | 81                             |

8.1.6 Breast Cancer Wisconsin

Table 21: Breast Cancer Wisconsin: Rule followed for renaming of the class labels.

| Class Label  | Numeric Code | Number of Total Data Instances |
|--------------|--------------|--------------------------------|
| M (Malignant) | 0            | 212                            |
| B (Benign)   | 1            | 357                            |

8.1.7 Statlog (Heart)

Table 22: Statlog (Heart): Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| 1 (Absence) | 0            | 150                            |
| 2 (Presence)| 1            | 120                            |

8.1.8 Seeds

Table 23: Seeds: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|--------------------------------|
| 1.0 (Kama)  | 0            | 70                             |
| 2.0 (Rosa)  | 1            | 70                             |
| 3.0 (Canadian) | 2     | 70                             |

8.1.9 FSDD
Table 24: **FSDD**: Rule followed for renaming of the class labels.

| Class Label | Numeric Code | Number of Total Data Instances |
|-------------|--------------|-------------------------------|
| 0           | 0            | 50                            |
| 1           | 1            | 50                            |
| 2           | 2            | 50                            |
| 3           | 3            | 50                            |
| 4           | 4            | 46                            |
| 5           | 5            | 41                            |
| 6           | 6            | 50                            |
| 7           | 7            | 50                            |
| 8           | 8            | 43                            |
| 9           | 9            | 50                            |

8.2 Hyperparameter Tuning

The hyperparameter tuning for all algorithms for the respective datasets are provided below:

8.2.1 Decision Tree

Following are the hyperparameters tuned for Decision Tree:

1. **min_samples_leaf**: Defines the minimum number of samples required for a leaf node in the decision tree. It is tuned from 1 to 10 with a step-size of 1.
2. **max_depth**: Declares the maximum depth to which a decision tree can be grown. It is tuned from 1 to 10 with a step-size of 1.
3. **ccp_alpha**: A numpy array of alpha values obtained by devising Cost Complexity Pruning on the original decision tree. This array is obtained using the `cost_complexity_pruning_path`.

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Decision Tree are available in Table 25 and 26.
Table 25: **Decision Tree**: Tuned hyperparameters for all nine datasets (Part I). The performance metric used for the provided results is macro F1-score.

| Dataset                | Implementation            | Tuned Hyperparameters          | F1 Score |
|------------------------|---------------------------|--------------------------------|----------|
| **Iris**               | Stand-Alone Decision Tree | \( \text{min\_samples\_leaf} = 3 \) \( \text{max\_depth} = 3 \) | 0.931    |
|                        | ChaosFEX + Decision Tree  | \( \text{min\_samples\_leaf} = 3 \) \( \text{max\_depth} = 4 \) \( \text{ccp\_alpha} = 0.0 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.13 \) | 0.955    |
| **Ionosphere**         | Stand-Alone Decision Tree | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 2 \) \( \text{ccp\_alpha} = 0.0 \) | 0.882    |
|                        | ChaosFEX + Decision Tree  | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 7 \) \( \text{ccp\_alpha} = 0.0 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.22 \) | 0.921    |
| **Wine**               | Stand-Alone Decision Tree | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 3 \) \( \text{ccp\_alpha} = 0.0 \) | 0.916    |
|                        | ChaosFEX + Decision Tree  | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 6 \) \( \text{ccp\_alpha} = 0.0 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.07 \) | 0.949    |
| **Bank Note Authentication** | Stand-Alone Decision Tree | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 8 \) \( \text{ccp\_alpha} = 0.0 \) | 0.977    |
|                        | ChaosFEX + Decision Tree  | \( \text{min\_samples\_leaf} = 1 \) \( \text{max\_depth} = 6 \) \( \text{ccp\_alpha} = 0.000836 \) \( q = 0.080 \) \( b = 0.250 \) \( e = 0.233 \) | 0.961    |
| **Haberman’s Survival** | Stand-Alone Decision Tree | \( \text{min\_samples\_leaf} = 4 \) \( \text{max\_depth} = 6 \) \( \text{ccp\_alpha} = 0.005389 \) | 0.614    |
|                        | ChaosFEX + Decision Tree  | \( \text{min\_samples\_leaf} = 2 \) \( \text{max\_depth} = 8 \) \( \text{ccp\_alpha} = 0.005389 \) \( q = 0.81 \) \( b = 0.14 \) \( e = 0.003 \) | 0.648    |

### 8.2.2 Random Forest

Following are the hyperparameters tuned for Random Forest:

1. **n_estimators**: Defines the number of trees in the forest being grown. The values are tuned from the array \([1, 10, 100, 1000, 10000]\).

2. **max_depth**: Declares the maximum depth each tree the the forest is allowed to have. It is tuned from 1 to 10 with a step-size of 1.
Table 26: **Decision Tree**: Tuned hyperparameters for all nine datasets (Part II). The performance metric used for the provided results is macro F1-score.

| Dataset                          | Implementation        | Tuned Hyperparameters                                                                 | F1 Score |
|---------------------------------|-----------------------|---------------------------------------------------------------------------------------|----------|
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone Decision Tree | min_samples_leaf = 1, max_depth = 4, ccp_alpha = 0.0                                | 0.920    |
|                                 | ChaosFEX + Decision Tree | min_samples_leaf = 2, max_depth = 8, ccp_alpha = 0.005595, q = 0.930, b = 0.490, ε = 0.159 | 0.945    |
| Statlog (Heart)                 | Stand-Alone Decision Tree | min_samples_leaf = 6, max_depth = 3, ccp_alpha = 0.0006818                          | 0.779    |
|                                 | ChaosFEX + Decision Tree | min_samples_leaf = 7, max_depth = 4, ccp_alpha = 0.0, q = 0.08, b = 0.06, ε = 0.117 | 0.786    |
| Seeds                           | Stand-Alone Decision Tree | min_samples_leaf = 2, max_depth = 4, ccp_alpha = 0.005844                         | 0.918    |
|                                 | ChaosFEX + Decision Tree | min_samples_leaf = 2, max_depth = 5, ccp_alpha = 0.0, q = 0.02, b = 0.01, ε = 0.238 | 0.949    |
| FSDD                            | Stand-Alone Decision Tree | min_samples_leaf = 1, max_depth = 9, ccp_alpha = 0.0009206                           | 0.536    |
|                                 | ChaosFEX + Decision Tree | min_samples_leaf = 1, max_depth = 10, ccp_alpha = 0.0061384, q = 0.34, b = 0.499, ε = 0.178 | 0.537    |

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Random Forest are available in Table 27 and 28.
### Table 27: Random Forest

| Dataset                  | Implementation            | Tuned Hyperparameters | F1 Score |
|--------------------------|---------------------------|-----------------------|----------|
| Iris                     | Stand-Alone Random Forest | \( n_{\text{estimators}} = 100 \) \( \max_{\text{depth}} = 3 \) | 0.944    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 10 \) \( \max_{\text{depth}} = 5 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.11 \) | 0.944    |
| Ionosphere               | Stand-Alone Random Forest | \( n_{\text{estimators}} = 10000 \) \( \max_{\text{depth}} = 4 \) | 0.923    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 100 \) \( \max_{\text{depth}} = 10 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.23 \) | 0.928    |
| Wine                     | Stand-Alone Random Forest | \( n_{\text{estimators}} = 10 \) \( \max_{\text{depth}} = 4 \) | 0.980    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 10 \) \( \max_{\text{depth}} = 5 \) \( q = 0.21 \) \( b = 0.969 \) \( e = 0.05 \) | 0.987    |
| Bank Note Authentication | Stand-Alone Random Forest | \( n_{\text{estimators}} = 100 \) \( \max_{\text{depth}} = 8 \) | 0.991    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 100 \) \( \max_{\text{depth}} = 7 \) \( q = 0.080 \) \( b = 0.250 \) \( e = 0.233 \) | 0.967    |
| Haberman’s Survival      | Stand-Alone Random Forest | \( n_{\text{estimators}} = 1 \) \( \max_{\text{depth}} = 3 \) | 0.621    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 1000 \) \( \max_{\text{depth}} = 9 \) \( q = 0.810 \) \( b = 0.140 \) \( e = 0.003 \) | 0.651    |
| Breast Cancer Wisconsin  | Stand-Alone Random Forest | \( n_{\text{estimators}} = 1000 \) \( \max_{\text{depth}} = 9 \) | 0.956    |
|                          | ChaosFEX + Random Forest  | \( n_{\text{estimators}} = 100 \) \( \max_{\text{depth}} = 7 \) \( q = 0.930 \) \( b = 0.490 \) \( e = 0.159 \) | 0.955    |

#### 8.2.3 AdaBoost

Following are the hyperparameters tuned for Adaptive Boosting (AdaBoost):

1. **n_estimators**: An upper limit for the number of stumps being grown. The values are tuned from the array \([1, 10, 50, 100, 500, 1000, 5000, 10000]\).

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for AdaBoost are available in Table 29.
Table 28: **Random Forest**: Tuned hyperparameters for all nine datasets (Part II). The performance metric used for the provided results is macro F1-score.

| Dataset | Implementation | Tuned Hyperparameters | F1 Score |
|---------|----------------|-----------------------|----------|
| Statlog (Heart) | Stand-Alone Random Forest | $n_{estimators} = 1000$<br>$max\_depth = 2$ | 0.839 |
| | ChaosFEX + Random Forest | $n_{estimators} = 10000$<br>$max\_depth = 4$<br>$q = 0.08$<br>$b = 0.06$<br>$\epsilon = 0.17$ | 0.830 |
| Seeds | Stand-Alone Random Forest | $n_{estimators} = 100$<br>$max\_depth = 5$ | 0.918 |
| | ChaosFEX + Random Forest | $n_{estimators} = 1000$<br>$max\_depth = 5$<br>$q = 0.020$<br>$b = 0.070$<br>$\epsilon = 0.238$ | 0.941 |
| FSDD | Stand-Alone Random Forest | $n_{estimators} = 1000$<br>$max\_depth = 9$ | 0.9479 |
| | ChaosFEX + Random Forest | $n_{estimators} = 1000$<br>$max\_depth = 8$<br>$q = 0.340$<br>$b = 0.499$<br>$\epsilon = 0.178$ | 0.921 |

### 8.2.4 Support Vector Machine

Following are the hyperparameters tuned for Support Vector Machine (SVM):

1. **C**: Controls the bias-variance trade-off of the algorithm, known as the “Regularization Parameter”. It is tuned from 0.1 to 100.0 with a step-size of 0.1

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for Support Vector Machine are available in Table 30.
### Table 29: AdaBoost

Table hyperparameters for all nine datasets. The performance metric used for the provided results is macro F1-score.

| Dataset                      | Implementation       | Tuned Hyperparameters | F1 Score |
|------------------------------|----------------------|-----------------------|----------|
| Iris                         | Stand-Alone AdaBoost | n_estimators = 50     | 0.929    |
|                              | ChaosFEX + AdaBoost  | q = 0.21, b = 0.969, e = 0.03 | 0.915    |
| Ionosphere                   | Stand-Alone AdaBoost | n_estimators = 50     | 0.929    |
|                              | ChaosFEX + AdaBoost  | q = 0.21, b = 0.969, e = 0.02 | 0.921    |
| Wine                         | Stand-Alone AdaBoost | n_estimators = 10     | 0.896    |
|                              | ChaosFEX + AdaBoost  | q = 0.21, b = 0.969, e = 0.05 | 0.893    |
| Bank Note Authentication      | Stand-Alone AdaBoost | n_estimators = 500    | 0.999    |
|                              | ChaosFEX + AdaBoost  | q = 0.080, b = 0.250, e = 0.233 | 0.934    |
| Haberman’s Survival           | Stand-Alone AdaBoost | n_estimators = 10     | 0.620    |
|                              | ChaosFEX + AdaBoost  | q = 0.81, b = 0.14, e = 0.003 | 0.639    |
| Breast Cancer Wisconsin (Heart) | Stand-Alone AdaBoost | n_estimators = 1000   | 0.962    |
|                              | ChaosFEX + AdaBoost  | q = 0.930, b = 0.490, e = 0.159 | 0.952    |
| Statlog (Heart)              | Stand-Alone AdaBoost | n_estimators = 10     | 0.816    |
|                              | ChaosFEX + AdaBoost  | q = 0.08, b = 0.06, e = 0.17 | 0.815    |
| Seeds                        | Stand-Alone AdaBoost | n_estimators = 500    | 0.563    |
|                              | ChaosFEX + AdaBoost  | q = 0.020, b = 0.070, e = 0.238 | 0.918    |
| FSDD                         | Stand-Alone AdaBoost | n_estimators = 50     | 0.086    |
|                              | ChaosFEX + AdaBoost  | q = 0.340, b = 0.499, e = 0.178 | 0.170    |

### 8.2.5 k-Nearest Neighbors

Following are the hyperparameters tuned for k-Nearest Neighbors:

1. **k**: Defines the number of nearest training data samples to the testing data sample to be chosen. It is tuned from 1 to 6 with a step-size of 2.
**Table 30: Support Vector Machine (SVM):** Tuned hyperparameters for all nine datasets. Only FSDD uses the ‘linear’ kernel. All remaining datasets, use the ‘rbf’ kernel. The performance metric used for the provided results is macro F1-score.

| Dataset                              | Implementation                        | Tuned Hyperparameters | F1 Score |
|--------------------------------------|---------------------------------------|-----------------------|----------|
| Iris                                 | Stand-Alone SVM                        | $C = 4.4$             | 0.954    |
|                                      | ChaosFEX + SVM                         | $C = 38.7$ \(q = 0.21\) \(b = 0.969\) \(\epsilon = 0.13\) | 0.958    |
| Ionosphere                           | Stand-Alone SVM                        | $C = 2.6$             | 0.934    |
|                                      | ChaosFEX + SVM                         | $C = 3.4$ \(q = 0.21\) \(b = 0.969\) \(\epsilon = 0.37\) | 0.926    |
| Wine                                 | Stand-Alone SVM                        | $C = 0.6$             | 0.975    |
|                                      | ChaosFEX + SVM                         | $C = 16.0$ \(q = 0.21\) \(b = 0.969\) \(\epsilon = 0.01\) | 0.958    |
| Bank Note Authentication             | Stand-Alone SVM                        | $C = 1.3$             | 1.0      |
|                                      | ChaosFEX + SVM                         | $C = 16.8$ \(q = 0.080\) \(b = 0.250\) \(\epsilon = 0.233\) | 0.965    |
| Haberman’s Survival                  | Stand-Alone SVM                        | $C = 19.8$             | 0.597    |
|                                      | ChaosFEX + SVM                         | $C = 15.7$ \(q = 0.81\) \(b = 0.14\) \(\epsilon = 0.003\) | 0.609    |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone SVM                        | $C = 16.0$             | 0.981    |
|                                      | ChaosFEX + SVM                         | $C = 20.3$ \(q = 0.930\) \(b = 0.490\) \(\epsilon = 0.159\) | 0.948    |
| Statlog (Heart)                      | Stand-Alone SVM                        | $C = 0.2$             | 0.852    |
|                                      | ChaosFEX + SVM                         | $C = 2.5$ \(q = 0.08\) \(b = 0.06\) \(\epsilon = 0.17\) | 0.825    |
| Seeds                                | Stand-Alone SVM                        | $C = 0.9$             | 0.948    |
|                                      | ChaosFEX + SVM                         | $C = 2.4$ \(q = 0.020\) \(b = 0.070\) \(\epsilon = 0.238\) | 0.909    |
| FSDD                                 | Stand-Alone SVM (linear kernel)        | $C = 0.7$             | 0.952    |
|                                      | ChaosFEX + SVM (linear kernel)         | $C = 6.1$ \(q = 0.340\) \(b = 0.499\) \(\epsilon = 0.178\) | 0.978    |

All remaining hyperparameters offered by scikit-learn are retained in their default forms. The results of hyperparameter tuning for $k$-Nearest Neighbors are available in Table 31.
Table 31: **k-Nearest Neighbors (k-NN)**: Tuned hyperparameters for all nine datasets. The performance metric used for the provided results is macro F1-score.

| Dataset                        | Implementation       | Tuned Hyperparameters                  | F1 Score |
|-------------------------------|----------------------|----------------------------------------|----------|
| Iris                          | Stand-Alone k-NN     | $k = 5$                                 | 0.944    |
|                               | ChaosFEX + k-NN      | $k = 1$                                 | 0.936    |
|                               |                      | $q = 0.21$                              |          |
|                               |                      | $b = 0.969$                             |          |
|                               |                      | $c = 0.12$                              |          |
| Ionosphere                    | Stand-Alone k-NN     | $k = 1$                                 | 0.814    |
|                               | ChaosFEX + k-NN      | $k = 1$                                 | 0.886    |
|                               |                      | $q = 0.21$                              |          |
|                               |                      | $b = 0.969$                             |          |
|                               |                      | $c = 0.11$                              |          |
| Wine                          | Stand-Alone k-NN     | $k = 5$                                 | 0.957    |
|                               | ChaosFEX + k-NN      | $k = 1$                                 | 0.966    |
|                               |                      | $q = 0.21$                              |          |
|                               |                      | $b = 0.969$                             |          |
|                               |                      | $c = 0.10$                              |          |
| Bank Note Authentication      | Stand-Alone k-NN     | $k = 5$                                 | 0.998    |
|                               | ChaosFEX + k-NN      | $k = 3$                                 | 0.967    |
|                               |                      | $q = 0.080$                             |          |
|                               |                      | $b = 0.250$                             |          |
|                               |                      | $c = 0.233$                             |          |
| Haberman’s Survival           | Stand-Alone k-NN     | $k = 1$                                 | 0.562    |
|                               | ChaosFEX + k-NN      | $k = 5$                                 | 0.659    |
|                               |                      | $q = 0.81$                              |          |
|                               |                      | $b = 0.14$                              |          |
|                               |                      | $c = 0.003$                             |          |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone k-NN | $k = 5$                                 | 0.964    |
|                               |                      | $q = 0.930$                             |          |
|                               |                      | $b = 0.490$                             |          |
|                               |                      | $c = 0.159$                             |          |
| Statlog (Heart)               | Stand-Alone k-NN     | $k = 5$                                 | 0.803    |
|                               | ChaosFEX + k-NN      | $k = 5$                                 | 0.791    |
|                               |                      | $q = 0.08$                              |          |
|                               |                      | $b = 0.06$                              |          |
|                               |                      | $c = 0.17$                              |          |
| Seeds                         | Stand-Alone k-NN     | $k = 5$                                 | 0.930    |
|                               | ChaosFEX + k-NN      | $k = 1$                                 | 0.876    |
|                               |                      | $q = 0.020$                             |          |
|                               |                      | $b = 0.070$                             |          |
|                               |                      | $c = 0.238$                             |          |
| FSDD                          | Stand-Alone k-NN     | $k = 1$                                 | 0.834    |
|                               | ChaosFEX + k-NN      | $k = 1$                                 | 0.909    |
|                               |                      | $q = 0.340$                             |          |
|                               |                      | $b = 0.499$                             |          |
|                               |                      | $c = 0.178$                             |          |

**8.2.6 ChaosNet**

1. $q$: Initial Neural Activity, it is varied from 0.01 to 0.49 with a step-size of 0.01.

2. $b$: Discrimination Threshold, it is varied from 0.01 to 0.49 with a step-size of 0.01.

3. $c$: Noise Intensity, it is varied from 0.001 to 0.499 with a step-size of 0.001.
8.3 Results

8.3.1 Decision Tree

The results of all experiments using Decision Tree in the high training sample regime are shown in Table 32.

Table 32: Decision Tree: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                        | Implementation        | F1 Score (Test Data) |
|--------------------------------|-----------------------|----------------------|
| Iris                           | Stand-Alone Decision Tree | 0.967                |
|                                | ChaosFEX + Decision Tree | 1.0                  |
| Ionosphere                     | Stand-Alone Decision Tree | 0.881                |
|                                | ChaosFEX + Decision Tree | 0.891                |
| Wine                           | Stand-Alone Decision Tree | 0.904                |
|                                | ChaosFEX + Decision Tree | 0.822                |
| Bank Note Authentication       | Stand-Alone Decision Tree | 0.933                |
|                                | ChaosFEX + Decision Tree | 0.978                |
| Haberman’s Survival            | Stand-Alone Decision Tree | 0.516                |
|                                | ChaosFEX + Decision Tree | 0.482                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone Decision Tree | 0.696                |
|                                | ChaosFEX + Decision Tree | 0.882                |
| Statlog (Heart)                | Stand-Alone Decision Tree | 0.697                |
|                                | ChaosFEX + Decision Tree | 0.878                |
| Seeds                          | Stand-Alone Decision Tree | 0.880                |
|                                | ChaosFEX + Decision Tree | 0.880                |
| FSDD                           | Stand-Alone Decision Tree | 0.603                |
|                                | ChaosFEX + Decision Tree | 0.579                |
8.3.2 Random Forest

The results of all experiments using Random Forest in the high training sample regime are shown in Table 33.

Table 33: Random Forest: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                        | Implementation               | F1 Score (Test Data) |
|--------------------------------|------------------------------|----------------------|
| Iris                           | Stand-Alone Random Forest    | 1.0                  |
|                                | ChaosFEX + Random Forest     | 1.0                  |
| Ionosphere                     | Stand-Alone Random Forest    | 0.909                |
|                                | ChaosFEX + Random Forest     | 0.924                |
| Wine                           | Stand-Alone Random Forest    | 0.966                |
|                                | ChaosFEX + Random Forest     | 0.943                |
| Bank Note Authentication       | Stand-Alone Random Forest    | 0.974                |
|                                | ChaosFEX + Random Forest     | 0.978                |
| Haberman’s Survival            | Stand-Alone Random Forest    | 0.560                |
|                                | ChaosFEX + Random Forest     | 0.398                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone Random Forest | 0.919                |
|                                | ChaosFEX + Random Forest     | 0.918                |
| Statlog (Heart)                | Stand-Alone Random Forest    | 0.838                |
|                                | ChaosFEX + Random Forest     | 0.838                |
| Seeds                          | Stand-Alone Random Forest    | 0.877                |
|                                | ChaosFEX + Random Forest     | 0.828                |
| FSDD                           | Stand-Alone Random Forest    | 0.970                |
|                                | ChaosFEX + Random Forest     | 0.937                |
8.3.3 Adaptive Boosting (AdaBoost)

The results of all experiments using AdaBoost in the high training sample regime are shown in Table 34.

Table 34: *AdaBoost*: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                      | Implementation       | F1 Score (Test Data) |
|------------------------------|----------------------|----------------------|
| Iris                         | Stand-Alone AdaBoost | 0.967                |
|                              | ChaosFEX + AdaBoost  | 0.865                |
| Ionosphere                   | Stand-Alone AdaBoost | 0.926                |
|                              | ChaosFEX + AdaBoost  | 0.925                |
| Wine                         | Stand-Alone AdaBoost | 0.833                |
|                              | ChaosFEX + AdaBoost  | 0.846                |
| Bank Note Authentication     | Stand-Alone AdaBoost | 0.985                |
|                              | ChaosFEX + AdaBoost  | 0.910                |
| Haberman’s Survival          | Stand-Alone AdaBoost | 0.505                |
|                              | ChaosFEX + AdaBoost  | 0.609                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone AdaBoost | 0.858                |
|                              | ChaosFEX + AdaBoost  | 0.881                |
| Statlog (Heart)              | Stand-Alone AdaBoost | 0.777                |
|                              | ChaosFEX + AdaBoost  | 0.711                |
| Seeds                        | Stand-Alone AdaBoost | 0.746                |
|                              | ChaosFEX + AdaBoost  | 0.873                |
| FSDD                         | Stand-Alone AdaBoost | 0.080                |
|                              | ChaosFEX + AdaBoost  | 0.397                |
8.3.4 Support Vector Machine (SVM)

The results of all experiments using Support Vector Machine (SVM) in the high training sample regime are shown in Table 35.

Table 35: Support Vector Machine (SVM): Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                          | Implementation          | F1 Score (Test Data) |
|----------------------------------|-------------------------|----------------------|
| Iris                             | Stand-Alone SVM         | 0.966                |
|                                  | ChaosFEX + SVM          | 0.933                |
| Ionosphere                       | Stand-Alone SVM         | 0.924                |
|                                  | ChaosFEX + SVM          | 0.909                |
| Wine                             | Stand-Alone SVM         | 0.928                |
|                                  | ChaosFEX + SVM          | 0.896                |
| Bank Note Authentication         | Stand-Alone SVM         | 0.993                |
|                                  | ChaosFEX + SVM          | 0.978                |
| Haberman’s Survival              | Stand-Alone SVM         | 0.437                |
|                                  | ChaosFEX + SVM          | 0.447                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone SVM     | 0.824                |
|                                  | ChaosFEX + SVM          | 0.918                |
| Statlog (Heart)                  | Stand-Alone SVM         | 0.844                |
|                                  | ChaosFEX + SVM          | 0.801                |
| Seeds                            | Stand-Alone SVM         | 0.924                |
|                                  | ChaosFEX + SVM          | 0.827                |
| FSDD                             | Stand-Alone SVM         | 0.952                |
|                                  | ChaosFEX + SVM          | 0.978                |
8.3.5 k-Nearest Neighbors

The results of all experiments using $k$ - Nearest Neighbors in the high training sample regime are shown in Table 36.

Table 36: k-Nearest Neighbors ($k$-NN): Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                      | Implementation      | F1 Score (Test Data) |
|------------------------------|---------------------|----------------------|
| Iris                         | Stand-Alone $k$-NN  | 1.0                  |
|                              | ChaosFEX + $k$-NN   | 0.902                |
| Ionosphere                   | Stand-Alone $k$-NN  | 0.821                |
|                              | ChaosFEX + $k$-NN   | 0.939                |
| Wine                         | Stand-Alone $k$-NN  | 0.943                |
|                              | ChaosFEX + $k$-NN   | 0.873                |
| Bank Note Authentication      | Stand-Alone $k$-NN  | 0.993                |
|                              | ChaosFEX + $k$-NN   | 0.971                |
| Haberman’s Survival          | Stand-Alone $k$-NN  | 0.480                |
|                              | ChaosFEX + $k$-NN   | 0.455                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone $k$-NN  | 0.954                |
|                              | ChaosFEX + $k$-NN   | 0.935                |
| Statlog (Heart)              | Stand-Alone $k$-NN  | 0.847                |
|                              | ChaosFEX + $k$-NN   | 0.739                |
| Seeds                        | Stand-Alone $k$-NN  | 0.924                |
|                              | ChaosFEX + $k$-NN   | 0.854                |
| FSDD                         | Stand-Alone $k$-NN  | 0.885                |
|                              | ChaosFEX + $k$-NN   | 0.938                |
8.3.6 Gaussian Naive Bayes

The results of all experiments using Gaussian Naive Bayes (GNB) in the high training sample regime are shown in Table 37.

Table 37: **Gaussian Naive Bayes (GNB)**: Experiment results (test data macro F1-scores) for all nine datasets in the high training sample regime.

| Dataset                        | Implementation     | F1 Score (Test Data) |
|--------------------------------|--------------------|----------------------|
| Iris                           | Stand-Alone GNB    | 0.966                |
|                                | ChaosFEX + GNB     | 0.933                |
| Ionosphere                     | Stand-Alone GNB    | 0.862                |
|                                | ChaosFEX + GNB     | 0.771                |
| Wine                           | Stand-Alone GNB    | 1.0                  |
|                                | ChaosFEX + GNB     | 0.976                |
| Bank Note Authentication       | Stand-Alone GNB    | 0.785                |
|                                | ChaosFEX + GNB     | 0.781                |
| Haberman’s Survival            | Stand-Alone GNB    | 0.572                |
|                                | ChaosFEX + GNB     | 0.535                |
| Breast Cancer Wisconsin (Diagnostic) | Stand-Alone GNB | 0.858                |
|                                | ChaosFEX + GNB     | 0.812                |
| Statlog (Heart)                | Stand-Alone GNB    | 0.826                |
|                                | ChaosFEX + GNB     | 0.788                |
| Seeds                          | Stand-Alone GNB    | 0.846                |
|                                | ChaosFEX + GNB     | 0.849                |
| FSDD                           | Stand-Alone GNB    | 0.919                |
|                                | ChaosFEX + GNB     | 0.687                |