

Attack on Lattice Shortest Vector Problem using K-Nearest Neighbour

Shaurya Pratap Singh¹, Brijesh Kumar Chaurasia², Siddharth Gupta³,
Tanmay Tripathi⁴, Ayush Pal⁵

¹⁻⁵ Department of Computer Science and Engineering,
¹⁻⁵ Pranveer Singh Institute of Technology, Kanpur, INDIA
brijeshchaurasia@ieee.org

Abstract. Lattice-based cryptography is now the most effective and adaptable branch of post-quantum cryptography. The prime number factoring assumption or the presumption that the discrete logarithm problem is intractable are the two assumptions that underlie nearly all cryptographic security systems. Lattice-based cryptography has recently gained popularity as a means of improving security as worldwide prepares for the onset of quantum computing. Lattices are used to secure the systems; however, one of the problems is the Shortest Vector Problem. We addressed in this work the attack on lattice problems, especially two-dimensional, four-dimensional, six-dimensional, and ten-dimensional with the help of the machine learning algorithm K-Nearest Neighbour (KNN). Results and analysis findings demonstrate that the suggested approach can achieve accuracy of up to 65% on self-prepared datasets.

Keywords: Shortest Vector Problem (SVP), Lenstra, Lenstra, and Lovasz (LLL), K-Nearest Neighbour Algorithm (KNN), Lattice-based Cryptography (LBC), Euclidean Distance, Post-Quantum Cryptography (PQC).

1. Introduction

The shortest vector problem (*SVP*) and the closest vector problem (*CVP*) are two main problems of lattice [1]. The knapsack problem was simplified to the *SVP* in the 1980s, and it was solved using the LLL method [2], [3]. Lattice-based Cryptography (LBC) is one of the main branches of post-quantum cryptography; however, a lattice is a partially ordered set [4]. The goal is to locate the lattice points that are closest to a particular target. *CVP* is the homogeneous counterpart of *SVP* and aims to locate the shortest non-zero vectors in a lattice. Find a lattice vector for a given lattice L and a vector $v \in \mathbb{R}^m$ that minimizes the distance to v . Currently, cryptographic systems based on lattice theory have been shown to be resistant to quantum computers, as there has yet to be an efficient quantum algorithm for solving hard problems in lattices [5]. LBC system secure from attack because there no efficient algorithm for solving problem of lattice till date in existing literature, however, in era of quantum cryptography may be possibility to attack [6]. In this work, possibility to attack on SVP through ML is studied.

Structure of paper

The remainder of this paper is organized as follows. In Section 2, we introduce preliminaries on lattice, SVP and hardness of SVP, KNN algorithm with it's working procedure prerequisite for the proposed scheme. Section 3 provides a proposed mechanism attack on lattices using the KNN mechanism. The efficacy of the proposed mechanism based on results and simulation is presented in Section 4. A brief discussion of potential future research directions concludes Section 5.

2. Preliminary

In this section we have introduced lattice, SVP, and hardness of SVP.

2.1 Lattice

Let \mathbb{R}^m be the m -dimensional Euclidean space. A lattice \mathcal{L} in \mathbb{R}^m is the set

$$\mathcal{L}(\mathcal{B}) = \left\{ \sum_{i=1}^n a_i b_i : a_i \in \mathbb{Z} \right\} \quad (1)$$

$\mathcal{B} = \{b_1, b_2, \dots, b_n\}$ be a set of linearly independent vectors (also known as basis vectors) of \mathbb{R}^m , n is represent rank and m represent dimension. If $m = n$ then it becomes a full rank lattice [5]. An abstract structure known as a lattice is investigated in the mathematical fields of order theory and abstract algebra [6]. It consists of a partially ordered set with unique supremums (also known as least upper bounds or joins) and unique infimums (also known as greatest lower bounds or meet) for each pair of elements. Fig. 1 depicts a lattice in two dimensions.

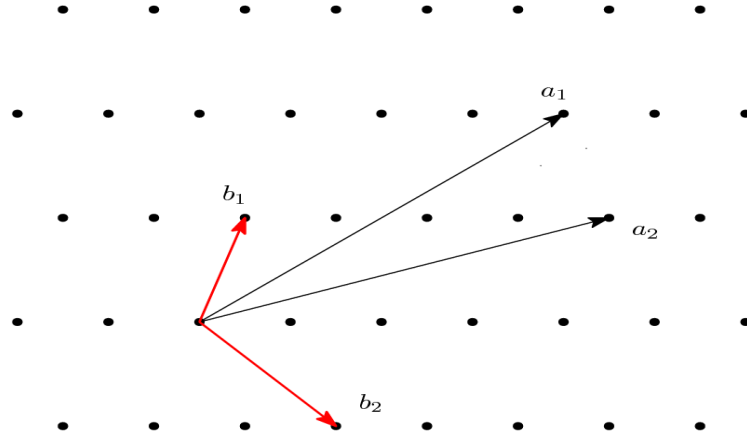


Fig. 1. Two Dimensional Lattices

2.2 Shortest Vector Problem (SVP)

The *SVP* is one of the most well-known issues in lattice. The goal of this task is to determine the shortest vector from an n -dimensional lattice (Fig. 2), and *SVP* is thought to be less challenging than *CVP*. Although it is simple to determine that [7] is NP-hard, it was unclear for a long time, however, the possibility is proposed in [8] for randomized reductions. However, the technique is not particularly effective, and because the time complexity is so high for big lattices, it is exceedingly challenging to tackle these issues with a conventional computer. There is no appropriate algorithm since the *SVP* cannot be solved in polynomial time. The most convincing NP-hardness result for *SVP* was in [9], which demonstrated that it is difficult to estimate *SVP* even within a factor of $\sqrt{2}$.

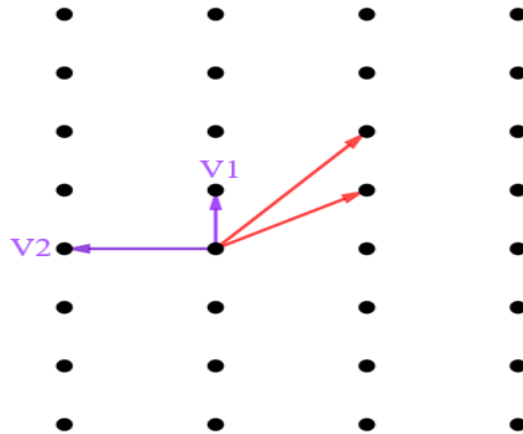


Fig. 2. Shortest Vector v_1 in \mathcal{L}

2.3 Definitions and Related Problems

Let (b_1, \dots, b_n) be a basis in R^n and N a norm on R^n .

Let \mathcal{L} denote $\bigoplus_{i=1}^n \mathbb{Z} b_i$, \mathcal{L} is called a lattice. Set

$$\lambda(\mathcal{L}) = \min_{v \in \mathcal{L} \setminus \{0\}} N(v) \quad (2)$$

We are now able to define *SVP*. For \mathcal{L} a lattice, the exact form *SVP*, denoted by $SVP(\mathcal{L})$ is as follows.

$$SVP(\mathcal{L}) : \text{Find } v \in \mathcal{L} \text{ such that } N(v) = \lambda(\mathcal{L})$$

As it is often the case, we will actually be interested in algorithms solving an approximation of *SVP*.

For $\gamma \geq 1$ the approximated form of $SVP(\mathcal{L})$, denoted by $SVP_\gamma(\mathcal{L})$ is

$$SVP_\gamma(\mathcal{L}) : \text{Find } V \in \mathcal{L} \setminus \{0\} \text{ such that } N(V) \leq \gamma \lambda(\mathcal{L}) \quad (3)$$

2.4 Hardness of *SVP*

The hardness of *SVP* depends on several factors, including the lattice dimension (n), the basis vectors, and the norm being minimized. In general, as the dimension n increases, *SVP* becomes computationally harder [10]. This is because the search space of possible lattice points grows exponentially with n . The hardness also depends on the choice of basis vectors and the structure of the lattice. For certain well-structured lattices, finding the shortest vector can be easier. The Euclidean norm is commonly used, but different norms can be considered depending on the problem, and the choice of norm affects the difficulty of *SVP*.

The class of problems known as *NP* includes those that are challenging to solve but whose solutions are straightforward to verify. These problems are especially helpful for cryptography. The *SVP* is known to be *NP-hard*. This means that it is at least as hard as the hardest problems in non-deterministic polynomial time (*NP*). In simple terms, if we can solve *SVP* efficiently, we can solve certain hard problems efficiently as well. This *NP-hardness* result implies that, unless $P = NP$ (which is an open problem), there is no known polynomial-time algorithm to solve *SVP* for all instances.

There are several known algorithms for approximating the *SVP* to within a certain factor, but these algorithms are all exponential in the dimension of the lattice. The best known approximation algorithm for the *SVP* is the Blockwise Korkine-Zolotarev (BKZ) algorithm [14], which can approximate the *SVP* to within a factor of $2^{\text{poly}(\log(n))}$ in polynomial time, where n is the dimension of the lattice as shown in Fig. 3.

According to Fig. 3, the complexity of the algorithm is labelled as $2^{n \log \log n / \log n}$ in the green box. It may be solved in polynomial time, though. Numerous cryptographic primitives, including lattice-based public-key encryption and digital signature systems, have been built using the *SVP's* hardness. These cryptographic fundamentals are a promising topic for post-quantum cryptography study since they can withstand assault from quantum computers.

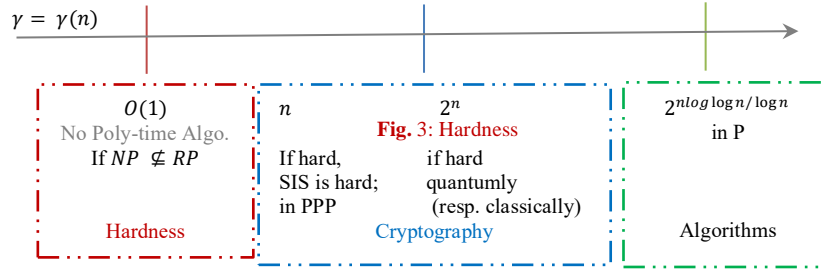


Fig. 3. Hardness of SVP on \mathcal{L} dimension n

2.5 KNN Algorithm

When there is a relationship between the input and output variables, ML uses regression techniques. It is used to predict continuous variables like the weather, property prices, and other things. The KNN [11] is one of the popular regression methods that is included in the general category of supervised learning.

Given that lattice vectors can be represented as coordinates and that KNN's Euclidean formula can be used to calculate distance, KNN is one of the best classifier algorithms. As shown in Fig. 4, we can also build two classification groups. The KNN four algorithm is used to calculate the classification distance: the Manhattan distance, the Minkowski distance [12], the Euclidean distance, and the Hamming distance. In our proposed study, Euclidean distance has been utilized.

If Euclidean distance is ($p = 2$), then computed distance is

$$d(x, y) = \sqrt{\sum_{i=1}^n (y_i - x_i)^2} \quad (4)$$

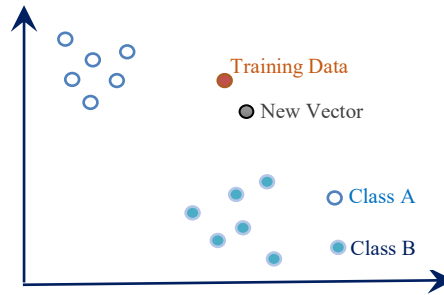


Fig. 4. Classification of New Data

2.5.1 Algorithm of Working Process of Proposed Method

In this sub-section working process of proposed method of attacks on SVM using ML and KNN is presented. Two algorithms are presented to compute the distance. The algorithm(s) are as follows:

Algorithm1: Euclidean Distance

Input: Two coordinates $pt_1 [x_1, y_1]$, $pt_2 [x_2, y_2]$

Output: Distance of vector(s)

1. Calculate Euclidean Distance(pt_1, pt_2)
2. Distance \leftarrow 0.0
3. *for* $i \leftarrow 0$ to $length(pt_1) - 1$ *do*
4. squared_diff $\leftarrow (pt_1[i], pt_2[i])^2$
5. distance \leftarrow distance + squared_diff
6. *end for*
7. Euclidean_distance \leftarrow square_root(distance)
8. *return* Euclidean_distance

Algorithm 2: K-Nearest Neighbour Algorithm


Input: Vector data (uses euclidean distance function for distance)

Output: Classes of vector(s)

1. Nearest_Neighbour ($train, test_obs, n$)
2. Neighbour_distance= []
3. *for* $i \leftarrow 0$ to $length(train)$ *do*
4. $l_1 = list(train.i\ loc\ Def[i]) + [Euclidean_dist(train.iloc[i, -1], test_obs)]$
5. Neighbour_distance=Neighbour_distance + [l_1]
6. *end for*
7. Neighbour_distance.sort($key=\lambda_x: x[-1]$)
8. Nearest_Neighbour = [Neighbour_distance[i] for i in range(0, n)]
9. $y_pred = [i[-2]$ for i in Nearest_Neighbour]
10. *return*($int(max(y_pred, key=y_pred.count))$)

3. Proposed Methodology

This section presents an SVM attack utilising ML and KNN. The SVM is described as a large-size, multi-dimensional lattice with the ability to extend in n dimensions. In a massive lattice, it is crucial to identify the shortest vector. The SVP problem still cannot be solved in polynomial time using any known algorithm. There are very few attempts in the literature currently in published. To the best of my knowledge, this is the first time a KNN attack has been made against the Lattice SVP. Lattice attributes were used to pinpoint the assault on a self-generated dataset up to 10 \mathcal{D} .

 **Lattice Vector Property:** The lattice problem is based on the lattice vector, and lattice vectors can be depicted as coordinates, as in Fig. 5.

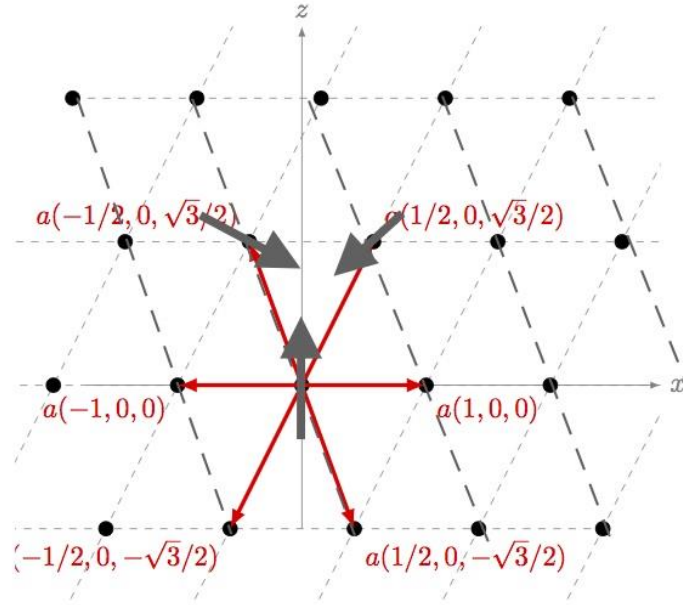


Fig. 5. Vector in Lattice

- **Size of Lattice used in Cryptography:** The size of the lattice is dependent on the system capacity because it is stated explicitly in the section under “Basis” that computers have finite amounts of memory and so cannot employ an arbitrarily large lattice.
- **KNN Algorithm Ideology for SVP:** For an n -dimensional lattice, it is very simple to calculate the distance of vectors and categories them since KNN can be extended to an n -dimensional lattice using Euclidean distance.

Fig. 5 depicts the suggested methodology’s operational flow. To begin with, we have produced a self-generated lattice dataset for our ML with KNN assault against SVP. Second, the resulting dataset is processed using the KNN method. Following that, generation is carried out as depicted in Fig. 5 based on vector size classes. The procedure is then repeated until the shortest vector is identified.

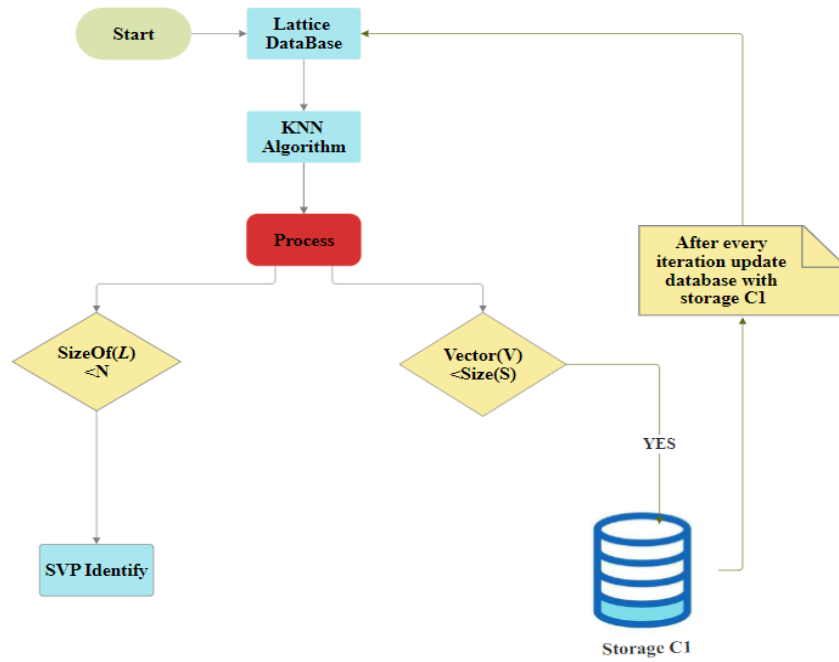


Fig. 6. Working Process of Proposed Methodology

When a new vector arrives, it moves to its nearest neighbor class based on prior information (Fig. 6). Since the shortest non-zero vector is always in the group with a small vector, as shown in Fig. 7, we eliminate the group of big vectors in the first iteration. We lower the lattice size by half ($10/2$) each time we repeat this operation n times.

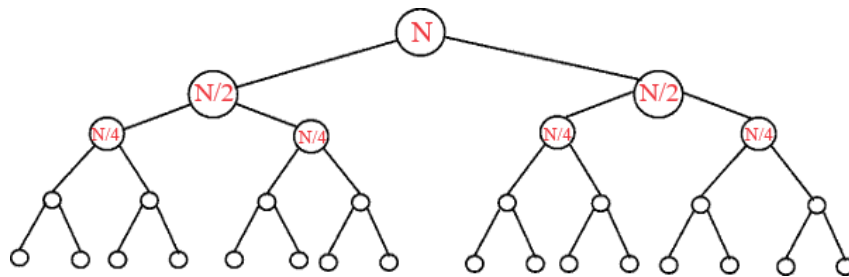


Fig. 7. Reduction of size of lattice

4. Results and Analysis

An attack on lattice SVP using KNN is presented in this section. We have used KNN to classify the shortest non-zero vectors in \mathcal{L} (or basis). To evaluate the efficacy of the proposed scheme, an Intel Core *i5* processor at 2.70 GHz and 8 GB of installed RAM with a Windows 11 OS laptop is used. The Python language [13] and Jupiter notebook is also used to classify the empirical results in terms of SVP non-zero vectors classification. First, we have created datasets of $2\mathcal{D}$, $4\mathcal{D}$, and $10\mathcal{D}$. Here, we have considered that the dimensions of the basis are represented by n . After that, we have applied the proposed mechanism. The experimental evaluation of attacks and comparison among $2\mathcal{D}$, $4\mathcal{D}$, and $10\mathcal{D}$, respectively, is as follows:

4.1 Attacks on $2\mathcal{D}$ Lattice using KNN

The following is an outline of the analysis of the suggested approach across a $2\mathcal{D}$ lattice:

4.1.1 Dataset of 2 dimensional Lattice

We have generated more than 1000 pairs of coordinates to evaluate the efficacy of our proposed scheme. Table 1 of this dataset demonstrates that the range of the mean value in this dataset is from 7.95 to 10.30, the range of the standard deviation is from 4.32 to 5.83, the range of the minimum value is 1.0, and the range of the maximum value is 20.0. After that, we processed the data in terms of classification into 0 and 1 randomly. In addition, consider the lattice dataset for $2\mathcal{D}$ lattice. After preprocessing the dataset, we have applied the proposed methodology as per Fig. 6.

Table1: Parameters for Dataset of $2\mathcal{D}$ Lattice

	x1	x2	y1	y2	outcome
count	999.000000	999.000000	999.000000	999.000000	999.000000
mean	7.951952	8.034034	10.166166	10.30030	0.513514
std	4.419588	4.325999	5.834939	5.69181	0.500068
min	1.000000	1.000000	1.000000	1.00000	0.000000
25%	4.000000	4.000000	5.000000	5.00000	0.000000
50%	8.000000	8.000000	10.000000	10.00000	1.000000
75%	12.000000	12.000000	15.000000	15.00000	1.000000
max	15.000000	15.000000	20.000000	20.00000	1.000000

The processed data are shown in Fig. 8 as two classes, 0 and 1. We have taken into account the fact that this categorization is based on a dataset.

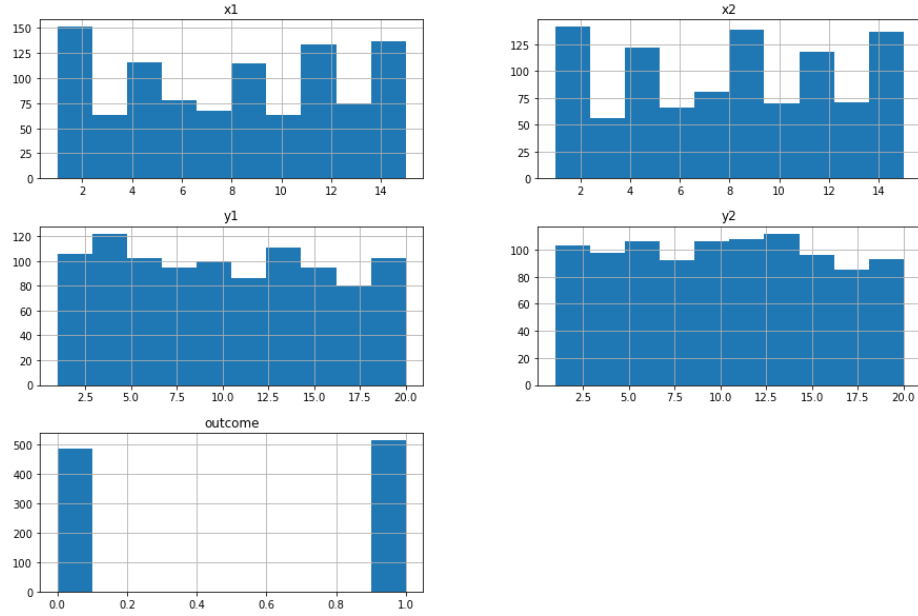


Fig. 8. Classification of 2 \mathcal{D} Lattice Dataset

In Fig. 8, there are five results showing that the variation of values in a 2 \mathcal{D} lattice dataset. The first is that the highest value is 150 and the lowest value varies between 50 and 75. The second shows that the highest value is more than 125 and the lowest value varies between 50 and 75, the highest value is more than 120 and the lowest value varies between 80 and 100. The highest value is more than 100, and the lowest value varies between 80 and 100. The graph is for outcome; it shows the value distribution after the execution of the proposed methodology.

4.1.2 Results of 2 \mathcal{D} Lattice

Results are shown in this subsection in terms of data classification, a heat map, and performance using ML parameters. The inefficient correlations or interactions between variables in a dataset are depicted in Fig. 9. In this illustration, the colour classes for x_1 , x_2 , y_1 , and y_2 0 and 1 are orange and blue, respectively. The KNN algorithm, which determines the distance metric among the pertinent characteristics, is used to distribute the distinctive feature vector in various classes in the figure. Using KNN and the best possible separation bounds, correlated feature points are shown as different orange and blue dots.

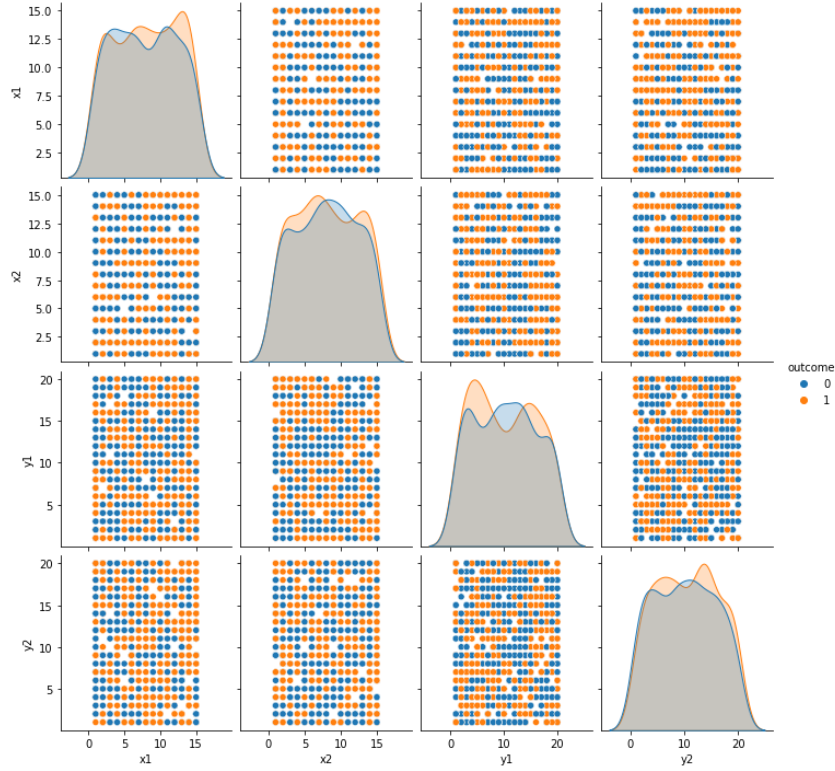


Fig. 9. Classification of 2 D Lattice Dataset

When a value of 1 is displayed for total correlation, as expected, Fig. 10 shows that a feature is completely connected with itself. Furthermore, the qualities are less related to the lower -0.004 value. However, 0.23 and -0.004 have a larger correlation for x_1 . The fact that 0.032 has a stronger correlation than -0.015 suggests that the two results cannot be compared.

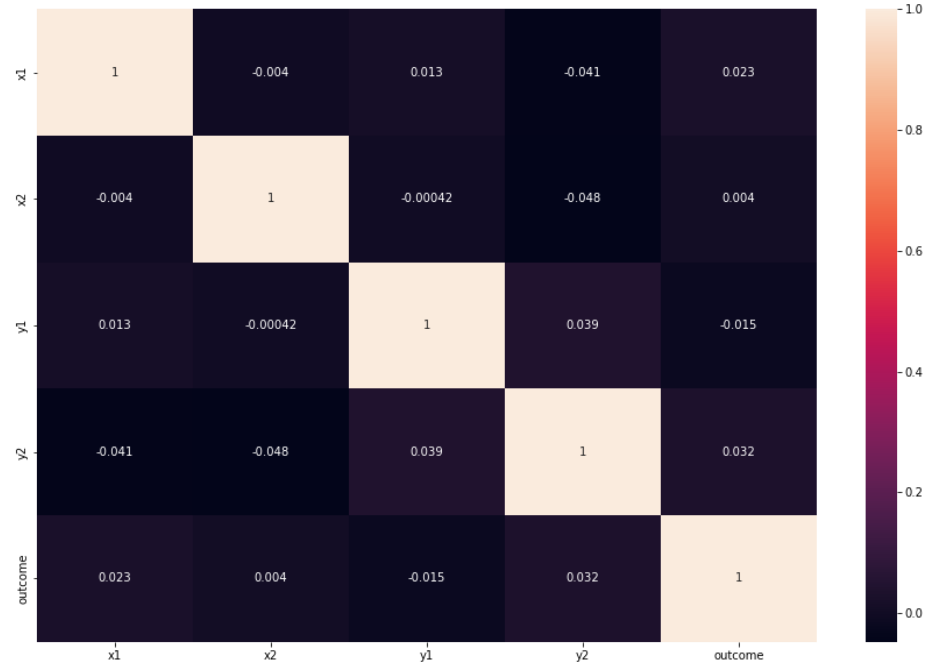


Fig. 10. Heat Map of 2 D Lattice Dataset

The results for class 0 are better than those for class 1, according to Fig. 11. For class 0, precision, recall, and F1-Score are 53, 61, and 57, respectively. It is also observed that for class 1, precision and F1-Score are increasing; however, recall is significantly less than in class 1 as per comparison to class 0.

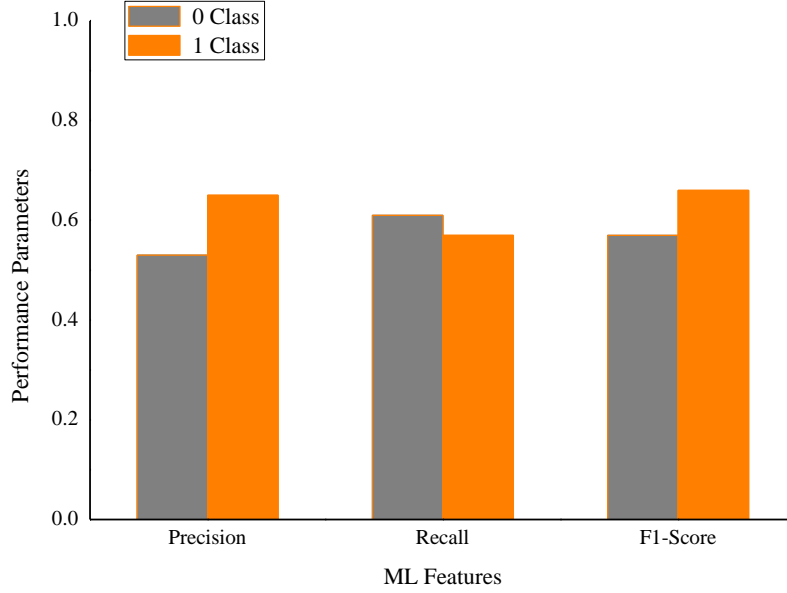


Fig. 11. Performance of proposed KNN scheme over 2 \mathcal{D} Lattice Dataset

4.2 Attacks on 4 \mathcal{D} Lattice using KNN

The following is an outline of the analysis of the suggested approach across a 4 \mathcal{D} lattice:

4.2.1 Dataset of 4 dimensional Lattice

We have considered more than 1000 pairs of coordinates to evaluate the efficacy of our proposed scheme. In this dataset, Table 2 shows that the mean value is 7.9 to 10.54, the standard deviation is considered 4.23 to 5.8, the minimum value is initialized at 1.0, and the maximum value is considered 20.0. After that, we processed the data in terms of classification into 0 and 1 randomly. After preprocessing the dataset, we have applied the proposed methodology as per Fig. 6.

Table 2: Parameters for Dataset of 4 \mathcal{D} Lattice

	x1	x2	x3	x4	y1	y2	y3	y4	outcome
count	999.000000	999.000000	999.000000	999.000000	999.000000	999.000000	999.000000	999.000000	999.000000
mean	8.030030	10.541542	7.951952	10.30030	8.034034	10.374374	7.880881	10.166166	0.513514
std	4.232366	5.707116	4.419588	5.69181	4.325999	5.883980	4.292981	5.834939	0.500068
min	1.000000	1.000000	1.000000	1.00000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	5.000000	6.000000	4.000000	5.00000	4.000000	5.000000	4.000000	5.000000	0.000000
50%	8.000000	10.000000	8.000000	10.00000	8.000000	10.000000	8.000000	10.000000	1.000000
75%	12.000000	15.000000	12.000000	15.00000	12.000000	16.000000	12.000000	15.000000	1.000000
max	15.000000	20.000000	15.000000	20.00000	15.000000	20.000000	15.000000	20.000000	1.000000

Fig. 12 is the representation of processed data into two classes 0 and 1. Here, we have considered that classification is based on a dataset.

In 4- D lattice there are four dimensions $(x_1, x_2, x_3, x_4), (y_1, y_2, y_3, \text{ and } y_4)$ using Euclidean formula algorithm divide the data in to two groups.

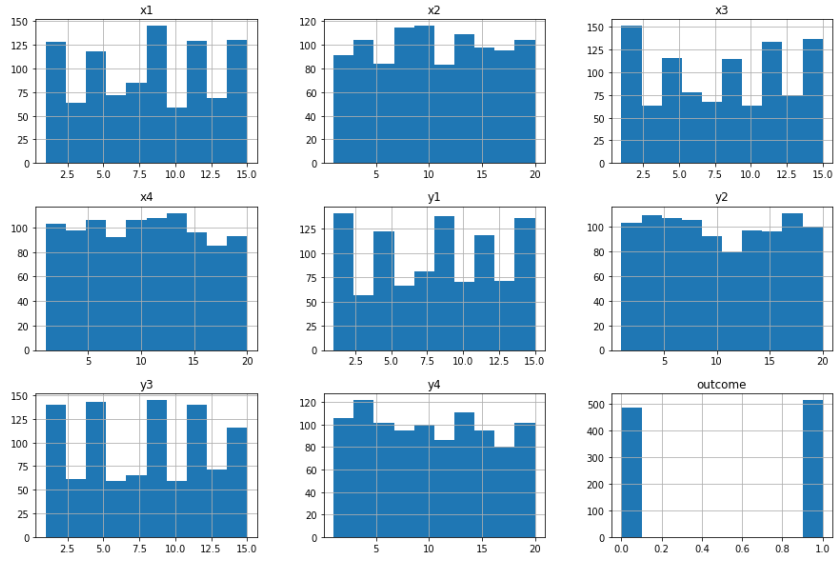


Fig. 12. Classification of 4 D Lattice Dataset

Fig. 12 shows nine results that show the variation of values in the 4 D lattice dataset. The first is that the highest values vary between 125 and 150, and the lowest values vary between 50 and 75. The same shows that the highest values vary between 100 and 120, and the lowest values vary between 80 and 100. The highest value is more than 150, and the lowest value is between 50 and 75. In the graph heights, the value is more than 100, and the lowest value is between 80 and 100. The highest value is more than 125, and the lowest value varies between 50 and 75. The highest value is more than 100, and the lowest value varies between 80 and 100; the highest value varies between 125 and 150; and the lowest value varies between 50 and 75. The highest value in the graph is greater than 120, and the lowest value ranges between 80 and 100. The last graph is for outcome; it shows the value distribution after the execution of the proposed methodology.

4.2.2 Results of 4 D Lattice

In this section, results are represented in terms of classification of data, heat map, and performance based on ML parameters. Fig. 13 shows the ineffective relationships or correlations between variables in a dataset. In this figure, orange and blue color

classes are considered for $x_1, x_2, x_3, x_4, y_1, y_2, y_3$, and y_4 0 and 1, respectively. The result shows the distribution of the distinctive feature vector in distinctive classes using the KNN algorithm that finds the distance metric among the relevant features. Correlated feature points are plotted in various orange and blue dots using KNN with optimal separation boundaries.



Fig. 13. Classification of 4 \mathcal{D} Lattice Dataset

Fig. 14 demonstrates that a feature is entirely correlated with itself when a value of 1 is displayed for total correlation, as expected. Additionally, the attributes are less associated with the lower the value of -0.0092. For x_1 , however, 0.0092 has a stronger correlation than -0.0092. The fact that 0.081 is more correlated than -0.00028 indicates that the two outcomes are not comparable.

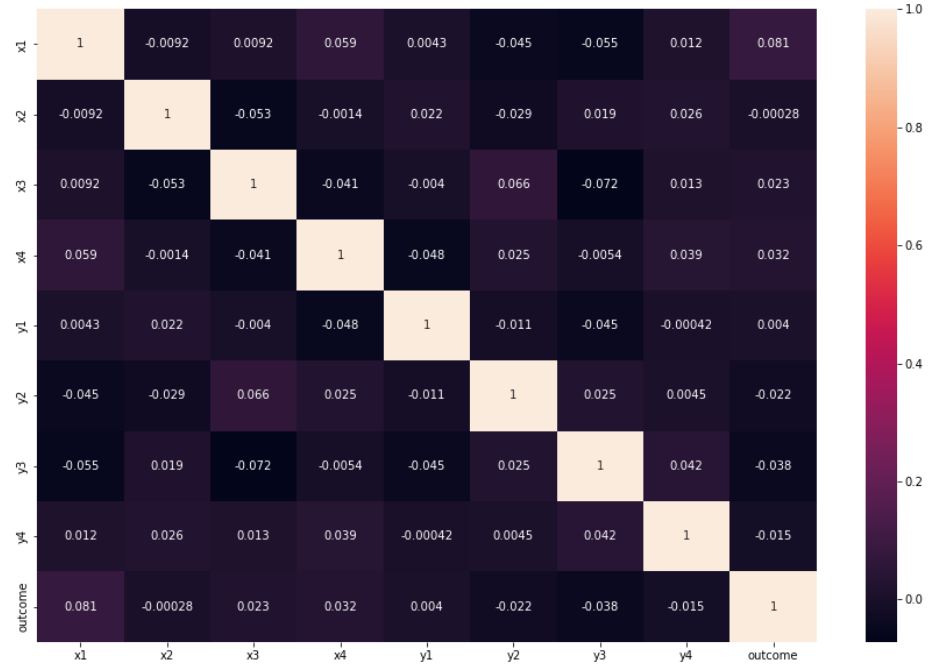


Fig. 14. Heat Map of 4 D Lattice Dataset

Fig. 15 shows that the results are better for class 1 compared to class 0. For class 0, precision, recall, and F1-Score are 45, 47, and 46, respectively.

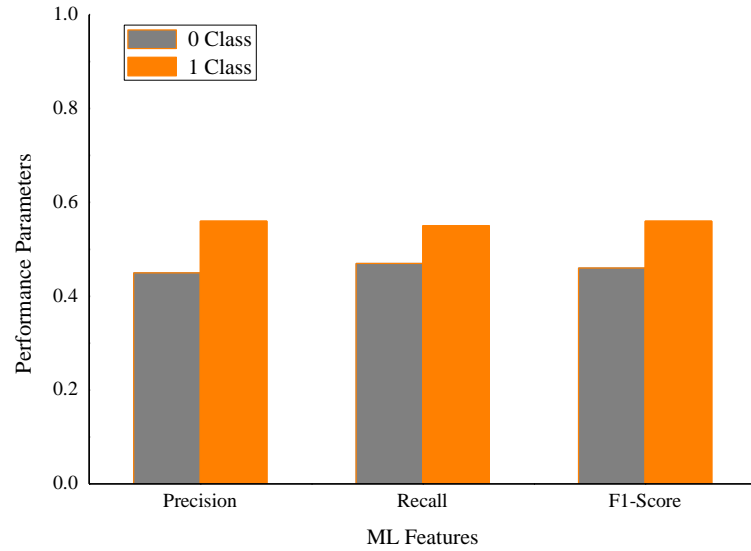


Fig. 15. Performance of proposed KNN scheme over 4 D Lattice Dataset

It is also observed that for class 1, all parameters are significantly higher than in class 0. The result of 4 \mathcal{D} is also proved that the attack on 4 \mathcal{D} is harder than 2 \mathcal{D} dataset.

4.3 Attacks on 10- \mathcal{D} Lattice using KNN

4.3.1 Dataset of 10 dimensional Lattice

We have considered more than 300 pairs of coordinates to evaluate the efficacy of our proposed scheme. In this dataset, Table 3 shows that the mean value is 7.30 to 10.79, the standard deviation is considered 4.25 to 5.66, the minimum value is initialized at 1.0, and the maximum value is considered 20.0. After that, we processed the data in terms of classification into 0 and 1 randomly. In addition, consider the lattice dataset for 10 \mathcal{D} lattice. After preprocessing the dataset, we have applied the proposed methodology as per Fig. 5.

Table 3: Parameters for Dataset of 10 \mathcal{D} Lattice

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	...	y2	y3	y4	y5	y6	y7	y8	y9	y10	outcome
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	...	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
mean	8.237458	10.503361	8.096980	10.023411	7.803685	8.344462	10.150902	10.120401	8.294314	9.411371	...	10.220738	8.100334	10.410380	8.200869	10.795987	7.307982	10.444818	7.812708	10.825418	9.501872
std	4.256274	5.791987	4.380421	5.551116	4.328721	4.338424	5.718821	5.178878	4.813285	5.479198	...	5.683788	4.284332	5.786281	4.530271	5.771833	4.438285	5.782588	4.363328	5.734837	5.008335
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
25%	5.000000	8.000000	4.000000	8.000000	4.000000	8.000000	5.000000	8.000000	4.000000	8.000000	...	5.000000	8.000000	5.000000	4.000000	5.000000	3.000000	5.000000	4.000000	8.000000	0.000000
50%	8.000000	10.000000	8.000000	10.000000	8.000000	8.000000	10.000000	10.000000	8.000000	9.000000	...	9.000000	8.000000	10.000000	8.000000	10.000000	7.000000	11.000000	8.000000	10.000000	1.000000
75%	12.000000	15.000000	12.000000	14.000000	11.000000	12.000000	15.000000	14.000000	12.000000	13.000000	...	15.000000	12.000000	15.000000	12.000000	16.000000	11.000000	15.000000	11.000000	16.000000	1.000000
max	15.000000	20.000000	15.000000	20.000000	15.000000	15.000000	20.000000	20.000000	20.000000	20.000000	...	20.000000	15.000000	20.000000	15.000000	20.000000	15.000000	20.000000	15.000000	20.000000	1.000000

Fig. 16 is the representation of processed data into two classes 0 and 1 for 10 \mathcal{D} basis. Here, we have considered that classification is based on a self-generated dataset. In 10 \mathcal{D} lattice there are **ten** dimension ($x_1, x_2, x_3, \dots, x_{10}$) and ($y_1, y_2, y_3, \dots, y_{10}$) are present and Euclidean formula for 10 \mathcal{D} is used make two classes.

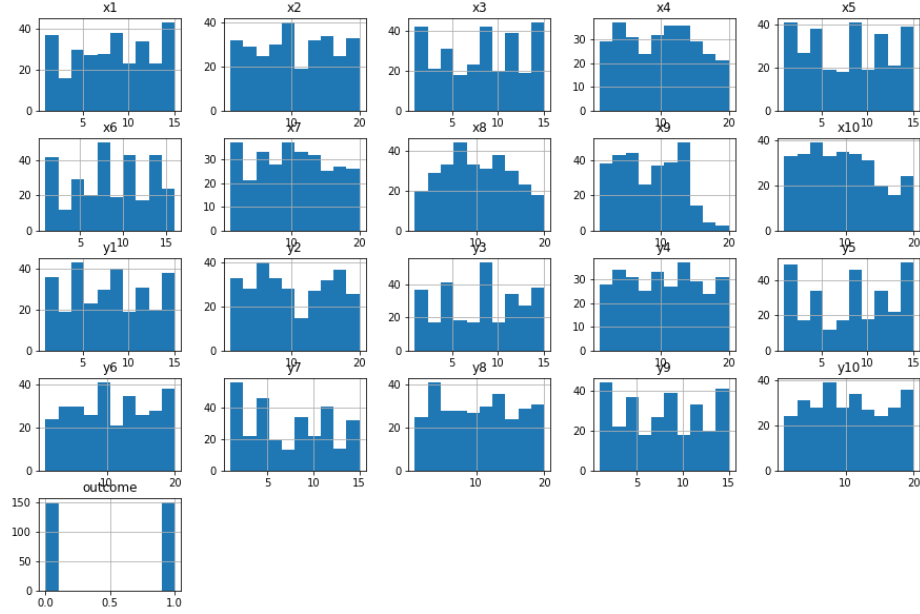


Fig.16. Classification of 10 D Lattice Dataset (Diagram will be changed)

There are sixteen results that demonstrate the variation of values in the 10d lattice dataset in Fig. 16. In the first, the highest value is greater than 40 and the lowest value is between 0 and 20. In the second, the highest value is between 20 and 40 and the lowest value is between 0 and 20. In the third, the highest value is greater than 30 and the lowest value is between 20 and 30, the highest value is greater than 40 and the lowest value is between 0 and 20, and the highest value is greater than 40 in the highest value. In graphs where the highest value is more than 30 and the lowest value is between 20 and 30, highest value is more than 40 and lowest value is between 0 and 20, highest value is more than 20 and lowest value is between 0 and 20, highest value is more than 40 and lowest value is between 0 and 20, highest value is more than 40 and lowest value is between 20 and 40, highest value is more than 40 and lowest value is between 0 and 20, etc. The final graph represents the results and displays the value distribution following the use of the suggested methods.

4.3.2 Results of 10 D Lattice

In this section, results are represented in terms of classification of data, heat map, and performance based on ML parameters. Fig. 17 shows the ineffective relationships or correlations between variables in a dataset. In this figure, orange and blue color classes are considered for $(x_1, x_2, x_3, \dots, x_{10})$ and $(y_1, y_2, y_3, \dots, y_{10})$ 0 and 1, respectively. The result shows the distribution of the distinctive feature vector in distinctive classes using the KNN algorithm that finds the distance metric among the relevant features. Correlated feature points are plotted in various orange and blue dots using KNN with optimal separation boundaries.

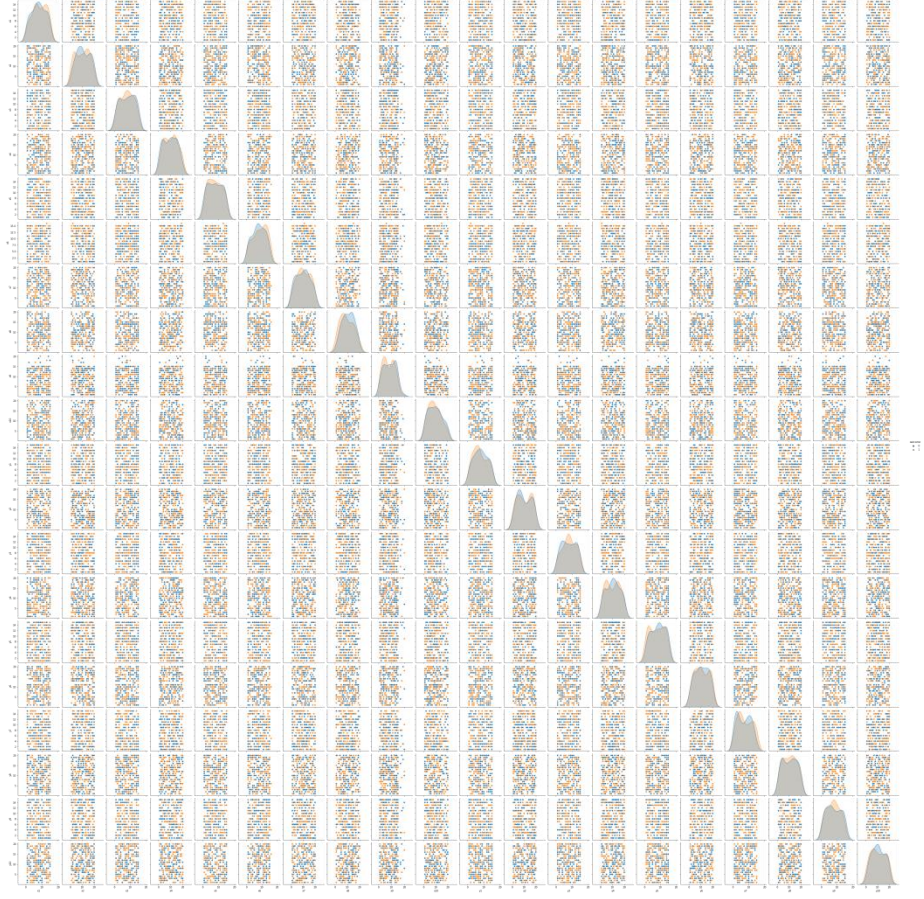


Fig. 17: Classification of 4 \mathcal{D} Lattice Dataset

Fig. 18 demonstrates that a feature is entirely correlated with itself when a value of 1 is displayed for total correlation, as expected. Additionally, the attributes are less associated with the lower the value of -0.0027 . For x_1 , however, 0.11 has a stronger correlation than -0.0027 . The fact that 0.085 is more correlated than -0.086 indicates that the two outcomes are not comparable.



Fig. 18. Heat Map of 4 D Lattice Dataset

Fig. 19 shows that the results are better for class 0 compared to class 1. All parameters are achieved up to 65%; however, for one class, it is up to 58%.

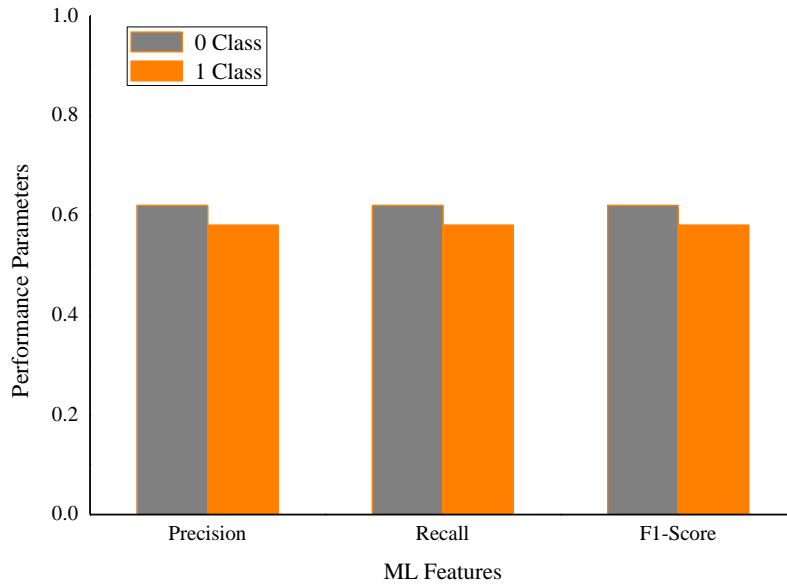


Fig. 19. Performance of proposed KNN scheme over 10 D Lattice Dataset

The analysis shows that for 4 \mathcal{D} datasets, the attack is a little bit harder compared to 2 \mathcal{D} and 10 \mathcal{D} . However, only at 10 \mathcal{D} database class 1 is it hard to attack using the proposed mechanism with a consequence of class 0. Moreover, in the case of the other two classes, the results are inverse.

Accuracy at all dimensions.

Comparison with K-Means, one paragraph.

5. Conclusion and Future Work

The field of post-quantum cryptography is currently one of the most active fields of research, and companies like IBM, Microsoft, and Google work on this field to secure information because it is very fast in comparison to classical computers and can break current cryptography algorithms like RSA, ECC, etc. In this work, an attack on the SVM of the lattice using ML with KNN is studied. The proposed model is able to achieve significant accuracy of up to 79% over a 2D lattice. In the future, we will propose hybrid models and other approaches other than KNN, such as 2D lattice.

References

1. Sun, Z., Gu, C., Zheng, Y.: A Review of Sieve Algorithms in Solving the Shortest Lattice Vector Problem. In IEEE Access. 8, 190475- 190486 (2020)
2. Lenstra, A. K., Lenstra, H. W., Lovász, L.: Factoring polynomials with rational coefficients. In Mathematische Annalen. 261(4), 515–534 (1982)
3. Lagarias, J.C.: Knapsack public key cryptosystems and diophantine approximation. In Advances in Cryptology, 3–23 (1983)
4. Micciancio, D., Regev, O.: Lattice-based Cryptography, 1-33, <https://cims.nyu.edu/~regev/papers/pqc.pdf> (2008). Accessed 30 September 2023
5. Zhang, J., Zhang, Z.: Lattice-Based Cryptosystems-A Design Perspective. Springer Singapore. XIII, 174 (2020)
6. Bandara, H., Herath, Y., Weerasundara, T., Alawatugoda, J.: On Advances of Lattice-Based Cryptographic Schemes and Their Implementations. In Cryptography, MDPI. 6(56), 1-22 (2022)
7. Guo, G., Wang, H., Bell, D., Bi, Y., Greer, K.: KNN Model-Based Approach in Classification. Lecture Notes in Computer Science, 986–996 (2003)
8. Cunningham, P., Delany, S. J.: k-Nearest Neighbour Classifiers: 2nd Edition (with Python examples), Online available at: <https://arxiv.org/pdf/2004.04523.pdf> (2020). Accessed 30 September 2023
9. Damien, S., Ron, S.: Making NTRU as secure as worst-case problems over ideal lattices. In EUROCRYPT: Advances in Cryptology – EUROCRYPT 2011, 27–47(2011)
10. Seber, George, A.F., Alan, J. L. Linear regression analysis. John Wiley & Sons, 329 (2012).

11. Parthasarathy, G., Chatterji, B. N.: A class of new KNN methods for low sample problems. In IEEE Transactions on Systems, Man, and Cybernetics. 20 (3), 715–718 (1990)
12. Helfrich, B. Algorithms to construct Minkowski reduced and hermite reduced lattice bases. In Theor. Comput. Sci. 41, 125–139 (1985)
13. Dataset, Online available at: https://techpubs.jurassic.nl/manuals/0530/developer/Expl_MWG/sgi_html/ch03.html, Last accessed on 29 March 2023
14. Li, J., Nguyen, Phong Q.: A Complete Analysis of the BKZ Lattice Reduction Algorithm, 1-45, Online available at: <https://eprint.iacr.org/2020/1237.pdf>, Last accessed on 29 March 2023