International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021

An Overview: Stochastic Gradient Descent Classifier, Linear Discriminant Analysis, Deep Learning and Naive Bayes Classifier Approaches to Network Intrusion Detection

Oyeyemi Osho
Computational Data and Enabled Science & Engineering
Jackson State University
Jackson, Mississippi, USA

Sungbum Hong (PhD)
Computational Data and Enabled Science & Engineering
Jackson State University
Jackson, Mississippi, USA.

***Abstract*—The security of Network Systems is ravaged by attacks on Systems in a bid to gain unauthorized access into the network system. The aim of Network Intrusion Detection Systems is to detect anomaly patterns either while the attack is unfolding or after evidence that an intrusion occurred.**

The demand and crave for Internet usage have surged over the years and will continue to rise, which also puts gadgets that are connected to Networks at risk of attacks by Cyber Terrorist and hackers. This problem is not limited to individuals or Corporations alone but also E-Governments and Enterprises, despite billions of dollars allocated to Cyber Security, computer systems and networks do not give a 100 percent guarantee against Cyber-attacks. It is against this backdrop that we must establish Network Intrusion Detection Systems to reveal and counter Cyber-attacks on Networks and Computer Systems.

Keywords—Component; formatting; style; styling; Cyber Security; machine learning, Netwrok Intrusion Detection.

I. INTRODUCTION

Intrusion Detection Systems are developed to counter attacks on Networks and Cyber Systems, securing the network requires a robust procedure and model which protects the network. Hackers usually look for vulnerabilities in the system by presenting several forms of attack to determine the network behavior, however, with a properly designed Intrusion Detection System, such effort will be escalated and isolated from the network to forestall an infiltration.

In time past, system administrators performed Intrusion Detection by sitting in front of a console to monitor user schemes. Subsequent Intrusion Detection systems was in the late ‘70s and early ‘80s in which network administrators printed audit logs on fan-folded paper, stacks could be as tall as four to five feet high. Administrators used the audit logs as forensic tool for the purpose of detecting an attack in advance which makes it an uphill task to detect any form of attack due to the tedious task [107].

Technological advancement led audit logs to be moved online and many other programs used to analyze data, auditing data logs usually consumed the bandwidth of the network which led the administrators to audit the data logs at night when systems user load was low. This approach also meant that attacks would have occurred well in advance before it was detected. Real time Intrusion Detection Systems was developed in the early 90’s enabling attacks to be detected as well as attack preemption.

II. DETECTION TECHNIQUES

There are two categories of Detection System, namely Anomaly and Misuse Detection [106].

A.

Anomaly Detection: This is a kind of detection in which models are used to determine normal behavior by learning the behavioral pattern from network logs, any such activity different from this established pattern is taken as an anomaly or attack.

This approach models the normal network and system behavior and determines any abnormality. It can detect zero-day attacks it also customizes normal activity for each system thereby limiting attack on the system due to unpredictability. There are two categories of anomaly detection namely: static and dynamic.

Static Anomaly Identifier: The static portion of a system comprises of the system code and the portion of the system that remains constant. The static part of the system is the portion that contains the system files and represented as a bit of strings. If the attacker gains access to this part of the system and makes an alteration to it, such attack is said to be static anomaly attack, the static anomaly identifier checks for file integrity.

Dynamic Anomaly Identifier: This refers to the constantly changing portion of the system, the system may rely on parameters which are set during initialization to depict the conduct of the system. The system conduct is sequence of distinct events defined at the initialization of the system. The system conduct is dynamic and assumed to be normal, if the uncertain conduct is not taken as anomalous, intrusion detection into the system may not be possible. If on the other hand the uncertain conduct is considered anomalous, then intrusion detection could be possible.

The principal edge that anomaly detection systems have is that they can detect old unknown attacks by determining a normal intrusion pattern and recognizing any breach whether

it is part of the menace model or not, this system leads to a high false-positive rate [106].
B.
Misuse Detection: This approach is used to detect known attacks; the abnormal system behavior is first defined, and all other behavior is defined as normal. This approach requires updating the database frequently with rules.

The principal edge of misuse detection systems is that they channel the analysis on the audit data from the router logs and network logs looking for signature attacks which results in low false positives, however, the disadvantage of this system is that it can only detect known attacks which then requires that developers constantly update the model to include signature pattern of newly discovered attack types.

III. SGD (STOCHASTIC GRADIENT DESCENT) CLASSIFIER

Stochastic gradient descent is also known as incremental gradient descent, it is defined as an iterative method for optimizing a differentiable objective function, a stochastic approximation of gradient descent optimization.

L´eon [36] defines SGD as a drastic simplification of a gradient $E_N(f_w)$ with each iteration estimates the gradient based on a single randomly picked example z_t:

$$w_{t+1} = w_t - \gamma_t \nabla_w Q(z_t, w_t). \qquad (1)$$

The stochastic process $\{w_t, t = 1, \ldots\}$ depends on the examples randomly picked at each iteration. It is hoped that (1) behaves like its expectation despite the noise introduced by this simplified procedure.

The convergence speed of stochastic gradient descent is limited by the noisy approximation of the true gradient, when the gains decrease too slowly, the variance of the parameter estimate w_t decreases equally slowly. When the gains decrease too quickly, the expectation of the parameter estimate w_t takes a very long time to approach the optimum.

The second order stochastic gradient descent (2SGD) multiplies the gradients by a positive definite matrix Γ_t approaching the inverse of the Hessian:

$$w_{t+1} = w_t - \gamma_t \Gamma_t \nabla_w Q(z_t, w_t). \qquad (2)$$

This second order modification does not eradicate the stochastic noise and does not significantly improve the variance of w_t.
The SGD classifier implements a first-order SGD learning, and the algorithm iterates over the training samples and updates each sample according to the update rule:

$$w \leftarrow w - \eta\left(\alpha \frac{\partial R(w)}{\partial w} + \frac{\partial L(w^T x_i + b, y_i)}{\partial w}\right) \quad (3)$$

Where $\boldsymbol{\eta}$ is the learning rate which controls the step-size in the parameter space. The learning rate can either be constant or gradually decaying and (learning rate='optimal') is given by:

$$\eta^{(t)} = \frac{1}{\alpha(t_0 + t)} \qquad (4)$$

Learning SGD with CICIDS2017 training set
The dataset is split up into 70% training set and 30% testing set, after pruning the dataset the multi-dimensional data is given as:

```
Out[21]: (2830743, 55)
```

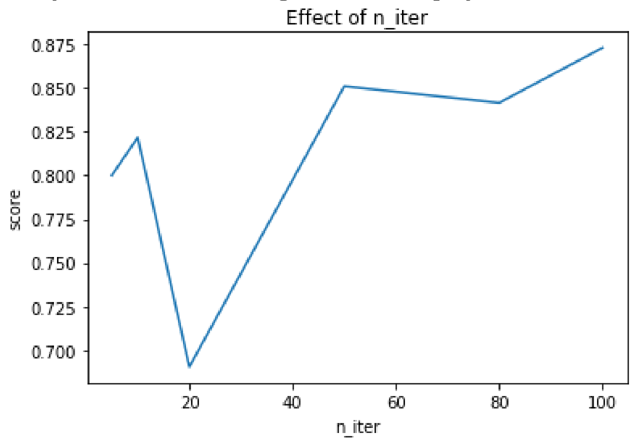
Fitting SGD into two arrays, X which includes the sample size and the features of the dataset and Y which holds the size of the samples and target values for the training set.

The SGD classifier is evaluated based on its accuracy to classify the training and testing test which yield:

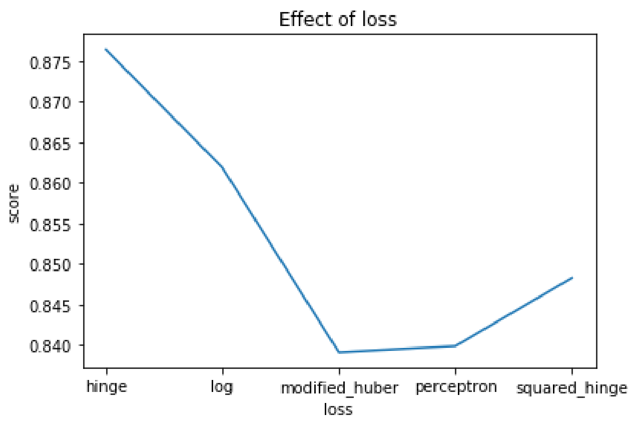
```
Accuracy of SGD classifier on training set: 0.87


Accuracy of SGD classifier on test set: 0.77
```

The effect of the number of iterations and the respective accuracy score on the training dataset is displayed:



Real world data has noise in it. This noise is expressed by random changes in values provided to the classifier. Because of this, the classifiers do not learn on a model, but rather on a combination of model with a background. Noise changes the output of the model, but to a much lesser extent than model parameters do. As displayed in the graph, the accuracy score is lowest at 20 iterations and increases as the iteration increases. As you increase the number of iterations, the precision with which stochastic gradient descent tries to fit the data grows, the algorithm modifies model parameters to account for noise induced fluctuations and it iterates over the training samples and updates each sample according to the update rule.



From the graph depicted hinge loss function has the highest predicted probability score, as the predicted probability approaches 1, the loss function decreases, the modified huber loss function has the lowest predicted probability score.

The gradient of the hinge loss is given as:

Hinge loss function $l_{hinge} = \max\left(0{,}1 - yx \cdot w\right)$ (5)

Rewrite hinge loss in terms of w as $f\left(g(w)\right)$ where $f(z) = \max(0{,}1 - y\, z)\ and\ g(w) = x \cdot w$ (6)

Using chain rule, we get

$$\frac{\partial}{\partial w_i} f\left(g(w)\right) = \frac{\partial f}{\partial z}\frac{\partial g}{\partial w_i} \quad (7)$$

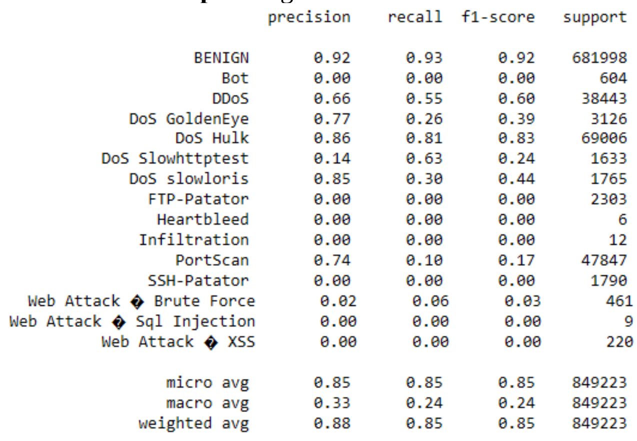
First derivative term is evaluated at $g(w) = x \cdot w$ becoming $-y\ when\ X \cdot w < 1.$

Second derivative term becomes x_i. In the end the result is:

$$\frac{\partial f(g(w))}{\partial w_i} = \begin{cases} -yx_i\ if\ yX \cdot W < 1 \\ 0\ if\ yX \cdot W > 1 \end{cases} \quad (8)$$

This is because i ranges over the components of x, this expression (8) can be viewed as a vector quantity, where $\left(\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}{}' \ldots\right)$.

The classification report is given below:


```
                          precision    recall  f1-score   support

                  BENIGN       0.92      0.93      0.92    681998
                     Bot       0.00      0.00      0.00       604
                    DDoS       0.66      0.55      0.60     38443
           DoS GoldenEye       0.77      0.26      0.39      3126
                DoS Hulk       0.86      0.81      0.83     69006
        DoS Slowhttptest       0.14      0.63      0.24      1633
           DoS slowloris       0.85      0.30      0.44      1765
             FTP-Patator       0.00      0.00      0.00      2303
              Heartbleed       0.00      0.00      0.00         6
            Infiltration       0.00      0.00      0.00        12
                PortScan       0.74      0.10      0.17     47847
             SSH-Patator       0.00      0.00      0.00      1790
  Web Attack � Brute Force       0.02      0.06      0.03       461
Web Attack � Sql Injection       0.00      0.00      0.00         9
          Web Attack � XSS       0.00      0.00      0.00       220

               micro avg       0.85      0.85      0.85    849223
               macro avg       0.33      0.24      0.24    849223
            weighted avg       0.88      0.85      0.85    849223
```

From the result above, the SDG classifier predictability on the BENIGN attack is the highest while the lowest predicted probability score is Heartbleed, Infiltration, Brute force, SQL injection and web attack. This attack types also have the lowest samples supported, the f score is computed with the harmonic mean of precision and recall, the classifier will get a high f score if both recall, and precision are high. The result shows that the weighted average for the SDG is 88%. The accuracy of the classifier is computed by comparing the actual test set values and predicted values, this is given as:

```
Accuracy: 0.8482824888162473
```

It can also be inferred from the result that this model predicts 88% of the time an attack is detected. The recall depicts that it predicts an attack occurrence of 85% of the time of an actual attack.

Precision Recall Curve

For the SDG to make a proper classification, it computes a probability based on a function and it classifies an input on the network based on the set of rules updated for each set of input sample data. It classifies each sample based on the learned dataset to differentiate a normal data from anomaly, if an anomaly intrusion is detected, it classifies the form of attack detected from the network based on the set threshold by the precision and recall curve, if the score is bigger than the threshold the input is regarded as anomaly but if less than the threshold, it is regarded as normal behavior by the input.

Precision (P) is defined as the number of true positives (T_p) over the number of true positives plus the number of false positives (F_p).

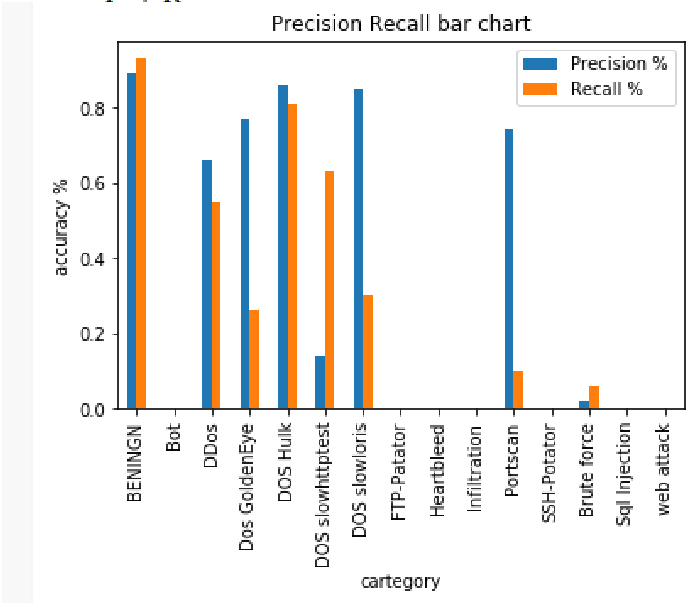
$$P = \frac{T_p}{T_p + F_p}$$

Recall (R) is defined as the number of true positives (T_p) over the number of true positives plus the number of false negatives (F_n).

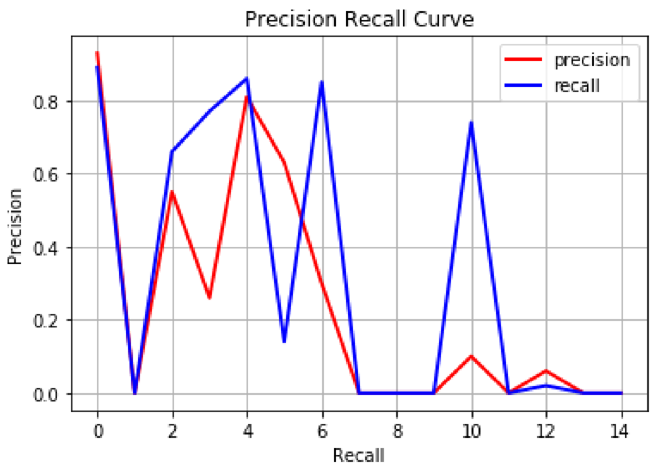
$$R = \frac{T_p}{T_p + F_n}$$

The F1 score is computed as:

$$F1 = 2\frac{P \times R}{P + R}$$



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021



This model prediction accuracy is given as:

```
Accuracy: 0.8224482850794197
```

IV. LINEAR DISCRIMINANT ANALYSIS

Linear Discriminant Analysis is used to cast a dataset matrix unto a lower dimensional space [93] such that the projected feature vectors of a class on the lower dimensional feature space is well separated from the feature vectors of other classes [94]. There are three steps needed to achieve a Linear Discriminant Analysis of a dataset matrix, these includes calculating the between-class variance, calculate the distance between the mean and the samples of each class and construct the lower dimensional space.

a. Calculating the between-class variance (S_B)

This is the distance between the mean of the ith class (μ_i) and the total mean (μ), it examines the lower-dimensional space to optimize the separation distance between classes. Given a dataset with matrix $X = \{x_1, x_2, \dots, x_N\}$ where x_1 is the ith observation and N is the number of samples collected. The samples are represented as a point in M-dimensional space $(x_i \in R^M)$, assume the dataset matrix is divided into 3 classes such that $X = [\omega_1, \omega_2, \omega_3]$, assume each class has five samples $(n_1 = n_2 = n_3 = 5)$ where n_i is the number of samples of the ith class [94]. The total number of samples is calculated as $N = \sum_{i=1}^{3} n_i$.

The between-class variance S_B is calculated by determining the separating distance between the different classes $(m_i - m)$, this is calculated as follows:

$$(m_i - m)^2 = (W^T\mu_i - W^T\mu)^2$$
$$= W^T(\mu_i - \mu)(\mu_i - \mu)^T \qquad (1)$$

Where m_i is the projection of the mean of the ith class and $m_i = W^T\mu_i$, m is the projection of the total mean of all classes and is given as, $m = W^T\mu$.

The mean of each class is $\mu_j = \frac{1}{n_j}\sum_{x_i \in \omega_j} x_i \qquad (2)$

The total mean is given as:

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i = \sum_{i=1}^{c} \frac{n_i}{N}\mu_i \qquad (3)$$

The term $(\mu_i - \mu)(\mu_i - \mu)^T$ is the between class variance of the ith class (SB_i), hence replacing it in equation (1) will result into:

$$(m_i - m)^2 = W^T S_{Bi} W$$

With this, the between-class matrix S_{Bi} is derived, subsequently, the total between-class matrix is derived as:

$$S_B = \sum_{i=1}^{c} n_i S_{Bi}$$

b. Calculating the within-class variance (S_W)

This is the difference between the mean and the dataset samples of that class, it uses the lower dimensional space to reduce the contrast between the projected data sample of the class $(W^T{}_{x_i})$ and the projected mean (m_i). The withing-class variance (S_{w_i}) is calculated as:

$$\sum_{x_i \in \omega_j, j=1,\dots,c} (W^T x_i - m_j)^2$$
$$= \sum_{x_i \in \omega_j, j=1,\dots,c} (W^T x_{ij} - W^T\mu_j)^2$$
$$= \sum_{x_i \in \omega_j, j=1,\dots,c} W^T(x_{ij} - \mu_j)^2 W$$
$$= \sum_{x_i \in \omega_j, j=1,\dots,c} W^T(x_{ij} - \mu_j)(x_{ij} - \mu_j)^T W$$

Equation (5)

$$= \sum_{x_i \in \omega_j, j=1,\dots,c} W^T SW_j W$$

Hence, within-class variance can be deduced as

$$SW_j = d_j^T *$$
$$d_j = \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T$$

Where d_j is the center data of the jth class and x_{ij} is the ith sample of the jth class; The total within-class variance is calculated as follows:

$$S_W = \sum_{i=1}^{3} SW_i$$

Equation (6)

$$= \sum_{x_j \in \omega_1} (x_i - \mu_1)(x_i - \mu_1)^T + \sum_{x_i \in \omega_2} ((x_i - \mu_2)(x_i - \mu_2)^T + \sum_{x_i \in \omega_3} (x_i - \mu_3)(x_i - \mu_3)^T$$

c. Constructing the lower dimensional space.

Constructing the lower dimensional space helps to optimize the between class variance and reduces the within-class variance. Applying the Fisher's criterion, Linear Discriminant Analysis can be calculated as [94]:

Equation (7)

$$arg_W^{max} \frac{W^T S_B W}{W^T S_W W}$$

Equation (8)

$S_W W = \lambda S_B W$

Where λ represents the eigen values of the transformation matrix (W), (V) represents the eigen vectors and the transformation matrix can be calculated as:

$W = S_W^{-1} S_B$

The dimension of the original dataset matrix $(X \in \mathbb{R}^{N\times M})$ is lowered by projecting it unto the lower dimensional space of LDA $(V_k \in \mathbb{R}^{M\times k})$, the dimension lowering into a lower space via the LDA is made possible by $M - k$ features which may either be left unused or removed for each sample. The resulting matrix will be the lower dimension space (V_k) which is $y_i = x_i V_k$.

LDA (Linear Discriminant analysis) determines group means and computes, for everyone, the probability of belonging to the different groups. The individual is then assigned to the group with the highest probability score. It is a dimensional reduction technique used for pattern classification in machine learning applications.

Learning LDA Models

LDA makes some simplifying assumptions about your data:

1. That your data is Gaussian, that each variable is shaped like a bell curve when plotted.
2. That each attribute has the same variance, that values of each variable vary around the mean by the same amount on average.

With these assumptions, the LDA model estimates the mean and variance from your data for each class. It is easy to think about this in the univariate (single input variable) case with two classes.

Advantages

It is a probabilistic model with interpretable topics.

It is appropriate for High dimensional datasets.

It helps to examine difference that exist among different groups.

Objective of LDA

It develops different discriminant functions which are linear combinations of the independent variables that can be used to completely discriminate between these categories of dependent variables in the best way.

Different approaches to LDA Class-dependent transformation: It involves maximizing the ratio of between class variance to within class variance, the essence of this approach is to maximize this ratio to obtain class separability.

Class-independent transformation: This approach involves maximizing the ratio of overall variance to within class variance. In this method, each class is taken as a separate class against all other classes.

Notation Table [37]

Notation	Description
$\boldsymbol{X}$	Data matrix
$\boldsymbol{N}$	Total number of samples in $\boldsymbol{X}$
$\boldsymbol{W}$	Transformation matrix
$\boldsymbol{n_i}$	Number of samples in $\boldsymbol{\omega_i}$
μ_i	The mean of the $\boldsymbol{ith}$ class
μ	Total or global mean of all samples
$\boldsymbol{S_{Wi}}$	Within-class variance or scatter matrix of the $\boldsymbol{ith}$ class $(\boldsymbol{\omega_i})$
$\boldsymbol{S_{Bi}}$	Between-class variance of the $\boldsymbol{ith}$ class $(\boldsymbol{\omega_i})$
$\boldsymbol{V}$	Eigenvectors of $\boldsymbol{W}$
$\boldsymbol{V_i}$	$\boldsymbol{ith}$ eigenvector
$\boldsymbol{x_{ij}}$	The $\boldsymbol{ith}$ sample in the j$\boldsymbol{th}$ class
$\boldsymbol{k}$	The dimension of the lower dimensional space $(\boldsymbol{V_k})$
$\boldsymbol{x_i}$	$\boldsymbol{ith}$ sample
$\boldsymbol{M}$	Dimension of $\boldsymbol{X}$ or the number of features of $\boldsymbol{X}$
$\boldsymbol{V_k}$	The lower dimensional space
$\boldsymbol{c}$	Total number of classes
$\boldsymbol{m_i}$	The mean of the $\boldsymbol{ith}$ class after projection
$\boldsymbol{m}$	The total mean of all classes after projection
$\boldsymbol{S_W}$	Within-class variance
$\boldsymbol{S_B}$	Between-class variance
Λ	Eigenvalue matrix
$\boldsymbol{\lambda_i}$	$\boldsymbol{ith}$ eigenvalue
$\boldsymbol{\gamma}$	Projection of the original data
$\boldsymbol{\omega_i}$	$\boldsymbol{ith}$ class

Table 3.6 Notation Table

Algorithm Multiple Linear Discriminant Analysis (LDA) with projection plane for multi dimension data

1. Given a set of N samples $[x_i]_{i=1}^{N}$, each of which is represented as a row of length M as (step (A)), and $X(N \times M)$ is given by:

$$X = \begin{bmatrix} x_{(1,1)} & x_{(1,2)} & \cdots & x_{(1,M)} \\ x_{(2,1)} & x_{(2,2)} & \cdots & x_{(2,M)} \\ \vdots & \vdots & \vdots & \vdots \\ x_{(N,1)} & x_{(N,2)} & \cdots & x_{(N,M)} \end{bmatrix}$$

2. Compute the within class scatter matrix (S_w):

   $S_w = S_1 + S_2$

   S_1 is the covariance for the class matrix C_1

   S_2 is the covariance for the class matrix C_2

$$S_1 = \sum_{x\in C_1} (x_1 - \mu_1)(x_1 - \mu_1)^T$$

$$S_2 = \sum_{x\in C_1} (x_2 - \mu_2)(x_2 - \mu_2)^T$$

3. Compute the best LDA projection vector. The vector will carry all the necessary features required for the target data.

4. Find the eigen vectors having the largest eigen values, eigen vector carry good balance between features.
   $S_w^{-1} S_B V = \lambda V$ Projection vector (scalar)
   $|S_w^{-1} S_B V - \lambda V| = 0$
5. Compute the dimension reduction.
   $y = W^T X$
6. Compute the Eigen vectors $AX = \lambda X$
7. Compute the projection space which maintains separability between the classes and tries to form compact structures.
   $S_W = Shrink$
   $S_B = Expand$
8. Compute the projection direction and plot the plane.

For classification, get a set of scalar values for each cluster then identify a point on the line which will separate the two or more classes. Find the mean and variance of each cluster points and identify the point on the line which separate the clusters.

Numerical examples

Given two different classes, $C_1(5 \times 2)$ and $C_2(5 \times 2)$ have sample data with each have two features as follows:

$$C_1 = \begin{bmatrix} 1 & 3 \\ 3 & 6 \\ 4 & 2 \\ 5 & 1 \\ 6 & 1 \end{bmatrix} \quad C_2 = \begin{bmatrix} 9 & 10 \\ 7 & 4 \\ 2 & 4 \\ 8 & 7 \\ 10 & 8 \end{bmatrix}$$

$$C_2 = \begin{bmatrix} 9 & 10 \\ 7 & 4 \\ 2 & 4 \\ 8 & 7 \\ 10 & 8 \end{bmatrix}$$

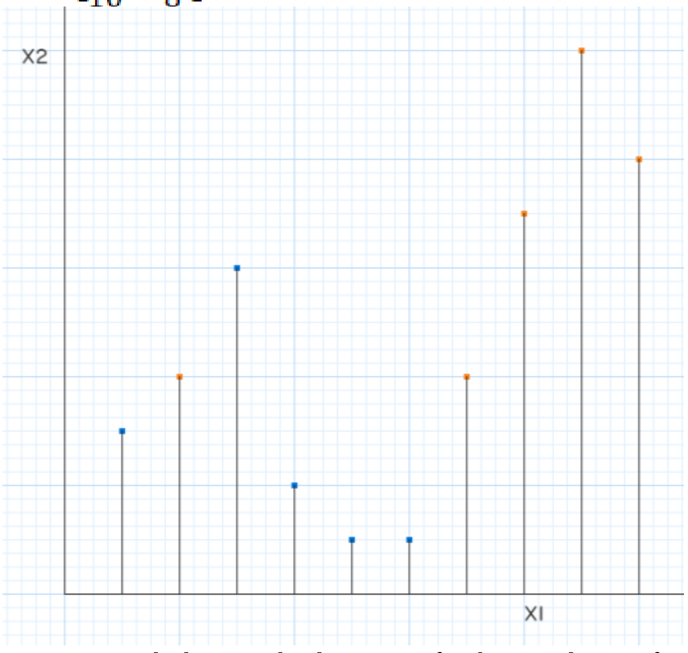


Fig 3.6 Graph showing the data points for the two classes of data in LDA.

Graph showing the data points for the two classes of data.

To compute the within class scatter matrix (S_w)

$$S_w = S_1 + S_2$$

$$S_1 = \sum_{x \in C_1} (x_1 - \mu_1)(x_1 - \mu_1)^T$$

$$S_2 = \sum_{x \in C_1} (x_2 - \mu_2)(x_2 - \mu_2)^T$$

μ_1	C_1					C_2					μ_2
3.8	1	3	4	5	6	9	7	2	8	10	7.2
2.6	3	6	2	1	1	10	4	4	7	8	6.6
$X_1 - \mu_1$	-2.8	-0.8	0.2	1.2	2.2	1.8	-0.2	-5.2	0.8	2.8	$X_1 - \mu_1$
$X_2 - \mu_2$	0.4	3.4	-0.6	-1.6	-1.6	3.4	-2.6	-2.6	0.4	1.4	$X_2 - \mu_2$

Table 3.6.1 Values Table

Compute $(X_1 - \mu_1)(X_1 - \mu_1)^T$ and $(X_2 - \mu_2)(X_2 - \mu_2)^T$

The following set of matrices was obtained after the computation:

$$\begin{pmatrix} -2.8 \\ 0.4 \end{pmatrix} [-2.8 \quad 0.4] = \begin{bmatrix} 7.84 & -1.12 \\ -1.12 & 0.16 \end{bmatrix} \Rightarrow M_1$$

$$\begin{pmatrix} -0.8 \\ 3.4 \end{pmatrix} [-0.8 \quad 3.4] = \begin{bmatrix} 0.64 & -2.72 \\ -2.72 & 0.16 \end{bmatrix} \Rightarrow M_2$$

$$\begin{pmatrix} 0.2 \\ -0.6 \end{pmatrix} [0.2 \quad -0.6] = \begin{bmatrix} 0.04 & -0.12 \\ -0.12 & 0.36 \end{bmatrix} \Rightarrow M_3$$

$$\begin{pmatrix} 1.2 \\ -1.6 \end{pmatrix} [1.2 \quad -1.6] = \begin{bmatrix} 1.44 & -1.92 \\ -1.92 & 2.56 \end{bmatrix} \Rightarrow M_4$$

$$\begin{pmatrix} 2.2 \\ -1.6 \end{pmatrix} [2.2 \quad -1.6] = \begin{bmatrix} 4.84 & -3.52 \\ -3.52 & 2.56 \end{bmatrix} \Rightarrow M_5$$

$$\begin{pmatrix} 1.8 \\ 3.4 \end{pmatrix} [1.8 \quad 3.4] = \begin{bmatrix} 3.24 & 6.12 \\ 6.12 & 11.56 \end{bmatrix} \Rightarrow M_6$$

$$\begin{pmatrix} -0.2 \\ -2.6 \end{pmatrix} [-0.2 \quad -2.6] = \begin{bmatrix} 0.04 & 0.52 \\ 0.52 & 6.76 \end{bmatrix} \Rightarrow M_7$$

$$\begin{pmatrix} -5.2 \\ -2.6 \end{pmatrix} [-5.2 \quad -2.6] = \begin{bmatrix} 27.04 & 13.52 \\ 13.52 & 6.76 \end{bmatrix} \Rightarrow M_8$$

$$\begin{pmatrix} 0.8 \\ 0.4 \end{pmatrix} [0.8 \quad 0.4] = \begin{bmatrix} 0.64 & 0.32 \\ 0.32 & 0.16 \end{bmatrix} \Rightarrow M_9$$

$$\begin{pmatrix} 2.8 \\ 1.4 \end{pmatrix} [2.8 \quad 1.4] = \begin{bmatrix} 7.84 & 3.92 \\ 3.92 & 1.96 \end{bmatrix} \Rightarrow M_{10}$$

$$S_1 = \begin{bmatrix} 2.96 & -1.88 \\ -1.88 & 3.44 \end{bmatrix} \quad S_2 = \begin{bmatrix} 7.76 & 4.88 \\ 4.88 & 5.44 \end{bmatrix}$$

$$S_w = S_1 + S_2$$

$$S_w = \begin{bmatrix} 10.72 & 3 \\ 3 & 8.88 \end{bmatrix}$$

Compute the projection vectors.

$$\vec{e} = S_w^{-1}(\mu_1 - \mu_2)$$

Compute S_w^{-1} as:

$$S_w^{-1} = \begin{bmatrix} 0.103 & -0.035 \\ -0.035 & 0.124 \end{bmatrix}$$

$$\mu_1 = \begin{pmatrix} 3.8 \\ 2.6 \end{pmatrix} \quad \mu_2 = \begin{pmatrix} 7.2 \\ 6.6 \end{pmatrix}$$

$$\mu_1 - \mu_2 = \begin{pmatrix} -3.4 \\ -4 \end{pmatrix}$$

$$\vec{e} = S_w^{-1}(\mu_1 - \mu_2)$$

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0.103 & -0.035 \\ -0.035 & 0.124 \end{bmatrix} \begin{bmatrix} -3.4 \\ -4 \end{bmatrix}$$

Projector vector $\vec{e} = \begin{pmatrix} -0.2102 \\ -0.377 \end{pmatrix}$

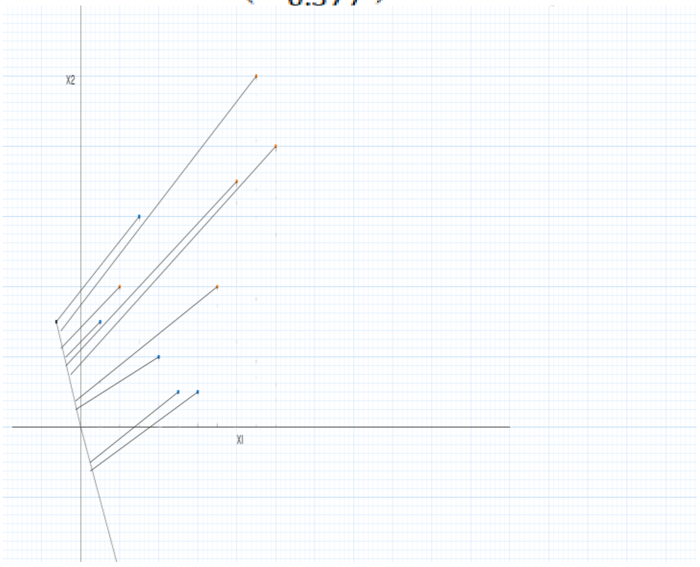


Fig 3.6.1 The projection plane of the LDA

The projection plane clearly separates the classes into clusters. Compute Dimension reduction and classification.

$$y = \vec{e}^T X$$

$$y_1 = \begin{bmatrix} 9 & 10 \\ 7 & 4 \\ 2 & 4 \\ 8 & 7 \\ 10 & 8 \end{bmatrix} \begin{bmatrix} -0.2102 \\ -0.377 \end{bmatrix}$$

$$y_1 = \begin{bmatrix} -1.3412 \\ -2.8926 \\ -1.5948 \\ -1.428 \\ -1.6382 \end{bmatrix}$$

$$y_2 = \begin{bmatrix} 9 & 10 \\ 7 & 4 \\ 2 & 4 \\ 8 & 7 \\ 10 & 8 \end{bmatrix} \begin{bmatrix} -0.2102 \\ -0.377 \end{bmatrix}$$

$$y_2 = \begin{bmatrix} -5.6618 \\ -2.9794 \\ -1.9284 \\ -4.3206 \\ -5.118 \end{bmatrix}$$

From the above solution the classes of data are well discriminated, for classification get a set of scalar values for each cluster then identify a point on the line which will separate the two classes. Find the mean and variance of each cluster points and identify the point on the line which separate the clusters.

if $\hat{\delta}_1(x) > \hat{\delta}_2(x) \Longrightarrow classify\ X\ as\ C_1$

if $\hat{\delta}_1(x) < \hat{\delta}_2(x) \Longrightarrow classify\ X\ as\ C_2$

V. DEEP LEARNING APPROACH

Deep learning is a machine learning technique used for feature extraction, data analysis and machine learning. It uses layers which are linked in its operations, each layer receives the output of the previous layer as input. The word "deep" in "deep learning" refers to the number of layers through which the data is transformed.

V.I GENERATIVE DEEP LEARNING METHOD

They model independent or dependent distributions in data and high order relation by identifying patterns. Some implementations of generative deep learning model are:

V.I.I Deep Restricted Boltzmann machine methods.

This method detects the performance evaluation different extraction and dimensional reduction techniques such as autoencoder, principal component analysis and statistical features. Restrictive Boltzmann machine are a variant of Boltzmann machines and have a pair of nodes from each of the groups of units where they may be symmetric connection between the groups of units, however there are no connections between nodes within a group.

The algorithm for Restrictive Boltzmann machine method is below [74]:

1. Take a training sample v, compute the probabilities of the hidden units, and sample a hidden activation vector h from this distribution.

2. Compute the outer product of v and h and call this the positive gradient.

3. From h, sample a reconstruction v' of the visible units, then resample the hidden activations h' from this.

4. Compute the outer product of v' and h' and call this the negative gradient.

5. Let the update to the weight matrix W be the positive gradient minus the negative gradient, times some learning rate: $\Delta W = \epsilon(v - v'), \Delta b = \epsilon(h - h')$.

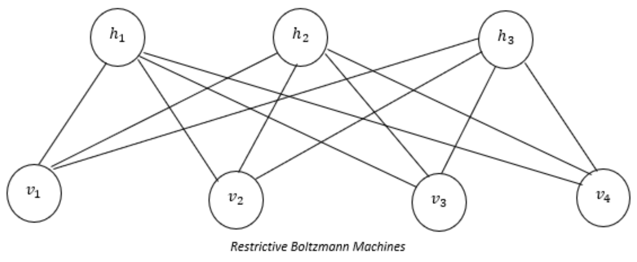


Fig 3.7 Restrictive Boltzmann Machines

Each neuron stores weight calculations that take place in each layer and each node can transmit the input weight in a random process through a method known as randomly generated stochastic coefficient.

For each visible unit $v = (v_1, v_2, \dots, v_n)$ and for each hidden unit $h = (h_1, h_2, \dots h_m)$ for a given Restricted Boltzmann machine with energy function:

$$Energy(v,h) = -b^T h - c^T v - hWv^T$$

Where b, c are offsets and biases while W is the weight matrix connecting the neurons.

The joint probability of (v, h) is as follows:

$$P(v,h) = \frac{1}{z} e^{-Energy(v,h)}$$

Where Z is the normalization used to restrict the result to lie between 0 and 1. The conditional probability of each visible unit is as follows:

$$P(v_i = 1|h) = \frac{P(v_i = 1, h)}{P(v_i = 0, h) + P(v_i = 1, h)}$$

$$= \frac{e^{c_i + W_i v}}{1 + e^{c_i + W_i v}} = sigm(c_i + W_i v)$$

Conditional probabilities for hidden units is obtained in same manner:

$$P(h_j = 1|v) = sigm(b_j + W_j^{-1} h)$$

$$P(v) \approx P_{train}(v)$$

Gibb's sampling is used to reduce the computational complexity.

V.I.II DEEP AUTOENCODER METHOD

Multimodal Deep autoencoder has a three-stage architecture, the first and third stages uses two autoencoders for learning the inner representations of 2D and 3D poses, the second stage uses a two-layer neural network to transform the 2D representation [75]. Deep Auto-Encoders are created by daisy chaining auto-encoders together [76], the output of each auto-encoder in the current layer is used as the input of the auto-encoder in the next layer. An auto-encoder is used to learn data coding in an unsupervised method, the autoencoder learns the dataset to achieve reduction in the dataset dimensionality by training the data to ignore noise. A pattern is created based on the learnt dataset which in turn used as a predictive model for other inputs. The simplest form of autoencoder is the feedforward neural network which has an input layer and one or more hidden layers connecting them where the output layer has the number of neurons as the input layer, this allows the system to reconstruct inputs instead of predicting the target value Y given a set of inputs X, hence autoencoders fall into the category of unsupervised models.

Autoencoder consist of the encoder and decoder with transitions such as:

$$\emptyset : X \rightarrow F$$

$$\varphi : F \rightarrow X$$

$$\emptyset, \varphi = \arg\min \; ||X - (\varphi^\circ \emptyset) X||^2$$

The encoder stage of an autoencoder takes the input $x \in R^d = X$ and maps it to $h \in R^p = F$

$$h = \sigma(Wx + b)$$

Where h is hidden variables, σ is the activation function, W is the weight matrix and b is the bias factor. The Daisy chaining autoencoder [76] based of Deep Auto encoder is as follows:

```
1. Input: Dataset $X = \{x_1, x_2, \dots, x_m\}$ with $m$ samples, number of hidden layers $L$
2. For $l \in [1, L]$ $do$
3.     Initialize $W_l = 0$, $\dot{W}_l = 0$, $b_l = 0$, $\dot{b}_l = 0$
4.     Define the $l - th$ hidden layer representation vector $h_l = s(W_l h_{l-1} + b_l)$
5.     Define the $l - th$ hidden layer output $y_l = s(\dot{W}_l h_l + \dot{b}_l)$
6.     While not stopping criterion do
7.         Compute $h_l$ for $h_{l-1}$
8.         Compute $y_1$
9.         Compute the loss function.
10.        Update layer parameters $\theta_l = (W_l, b_l)$ and $\dot{\theta}_l = \{\dot{W}_l, \dot{b}_l\}$.
11.  End while
12. End for
13. Initialize $(W_{l+1}, b_{l+1})$ at the supervised layer.
14. Calculate the labels for each sample $x_i$ of the training dataset $X$
15. Performs BP in a supervised way to tune parameter of layers.
```

V.I.III LONG SHORT-TERM MEMORY (LSTM)

LSTM is an implementation of Recurrent Neural Network [77], a common LSTM is composed of a cell, an input gate, an output gate and a forget gate. In comparison to a traditional Recurrent Neural Network which suffers from gradient vanishing problem the LSTM's cell vector can encapsulate the notion of forgetting part of its previously stored memory and also add new data to the memory unit.

The computations of the gates are described in the equations below [67]:

$$f_t = \sigma(W_f x_t + w_f h_{t-1} + b_f)$$

$$i_t = \sigma(W_i x_t + w_i h_{t-1} + b_i)$$

$$o_t = \sigma(W_o x_t + w_o h_{t-1} + b_o)$$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \sigma_c(W_c x_t + w_c h_{t-1} + b_c)$$

$$h_t = o_t \otimes \sigma_h(c_t)$$

Where f is forget, i is input, o is output gate vectors, respectively. W is the weights of input, w is weight of recurrent output, b is bias, σ is activation function, c is the cell state and $\otimes$ is the element-wise multiplication, LSTM recurrent neural networks can keep track of long-term dependencies,

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

The memory block is outlined in a dished box [77] and consists of an input gate, output gate and a forget gate, where the outputs of three gates are represented as i_t, o_t, f_t. The

activation vector is denoted as c_t and the memory block is denoted as m_t.
Training LSTM Neural Network is based on truncated Back Propagation through Time using the gradient descent optimization method, the essence of this function is to minimize the sum of square errors. This function eradicates errors as they arrive at a memory cell output where the memory cell's linear CEC is entered, [78] and errors can flow back outside the cell and then decay exponentially.

V.I.IV CONVOLUTION NEURAL NETWORK

CNN is an extension of [79] to traditional feed forward networks (FFN), they are regularized versions of multilayer perceptron, which allows a fully connected network which makes it prone to overfitting data. Normalization of this factor is adding some magnitude measurement of weights and using the loss function. The difference between a CNN and the normal neural networks is that CNN has [80] a feature extractor that consists of a convolution layer and a subsampling layer. The convolution layer of the CNN has several feature planes, with each plane having a number of neuron arranged in matrix, There's also the convolution kernels which are neurons of the feature plane and shared weights, the convolution kernels reduces the connections between layers of the network to reduce the risk of overfitting of data.
The CNN structure has two layers, the first layer is the feature extraction layer [80] which connects the entry of each neuron to the receptive domain of the previous layer and extracts the topical feature, this then determines the relationship with location of other feature. The second layer is the feature mapping layer, each layer comprises of multiple feature maps, the weight of all neurons on the same plane are equal and structure of feature mapping uses an activation function to fix the feature map.
The training process of CNN are as follows [80]:

a) Convolution: Each convolution filter represents a feature; the convolution process helps to determine whether a feature exists or not.

b) Subsampling: This process of fine-tuning signals from the convolution layer, this process eliminates sensitivity associated with filtered signals from the convolution layer.

c) Activation: This layer is responsible for signal flow control from one layer to another, the output signal associated with the reference will activate more neurons, this also makes the signal to propagate effectively and identifiable.

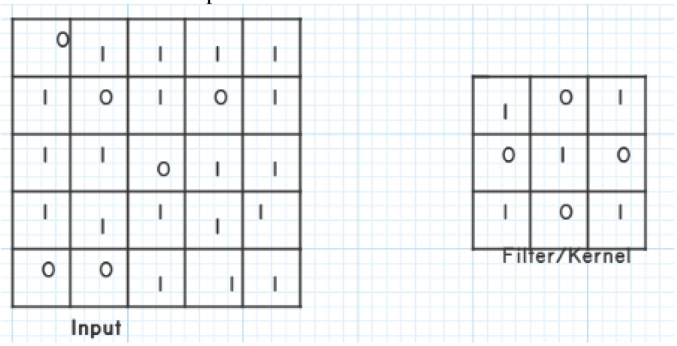
d) Full Connection: This layer is a fully connected layer, and it simulates all possible paths from input to output.

e) Loss: This layer provides feedback to the neural network, this system helps to detect if the input signal matches with the signal at the output, if incorrect, it will calculate the difference of measuring. This process enables the neural network to correct input identification in the training time.

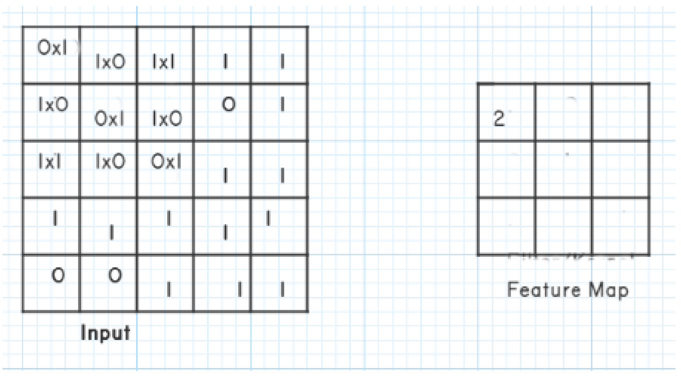
CNN Back propagation Algorithm [67]:

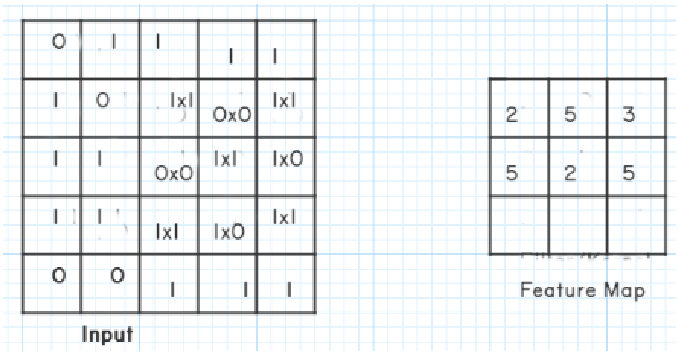
1. Initialization weights to randomly generated value.
2. Set learning rate to a small value (positive).
3. Iteration $n = 1; Begin$
4. For n<max iteration OR Cost function criteria met, do
5. For image x_1 *to* x_i, do
6. A. Forward propagate through convolution, pooling, and then fully connected layers.
7. B. Derive Cost Function value for the image.
8. C. Calculate error term $\delta^{(l)}$ with respect to weighs for each type of layers.
9. Note that the error gets propagated from layer to layer in the following sequence.
10. i. fully connected layer
11. ii. Pooling layer
12. iii. Convolution layer
13. d. Calculate gradient $\nabla_{w_k}(l)$ *and* $\nabla_{b_k}(l)$ for weights $\nabla_{w_k}(l)$ and bias respectively for each layer.
14. Gradient calculated in the following sequence.
15. i. convolution layer
16. ii. Pooling layer
17. iii. Fully connected layer
18. e. Update weights
19. $w_{ji}^{(l)} \leftarrow w_{ji}^{(l)} + \Delta w_{ji}^{(l)}$
20. F. Update bias
21. $b_j^{(l)} \leftarrow b_j^{(l)} + \Delta b_j^{(l)}$

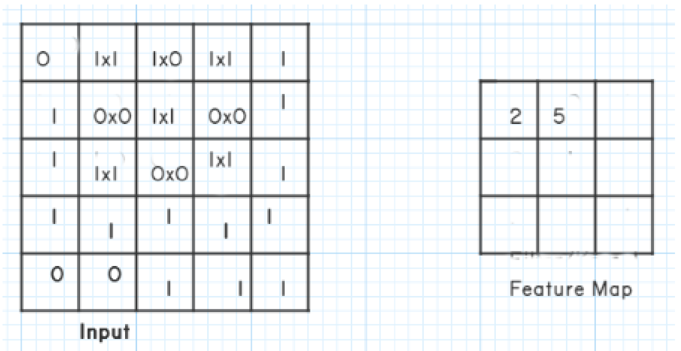
Forward propagation could be executed by flicking the kernel by 180° and stream the kernel across the input feature map repeatedly, the process is executed using distinct kernels to create as many feature maps craved.
The functionality of CNN is based of the main hidden layers listed:
a. Convolution layers: This is the main building block of the CNN; Convolution is a mathematical action used to combine two sets of data. A simple analysis of convolution operation is the application of a convolution filter on an input data to give rise to a feature map.

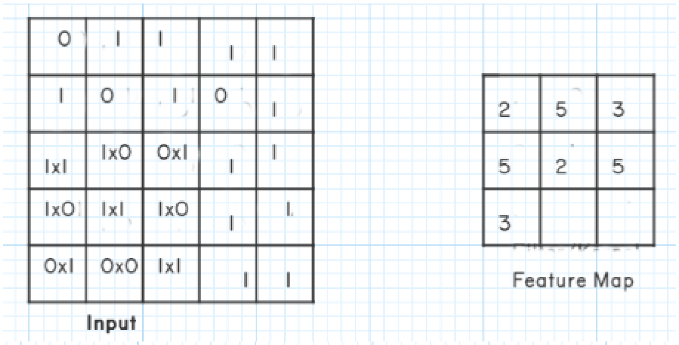


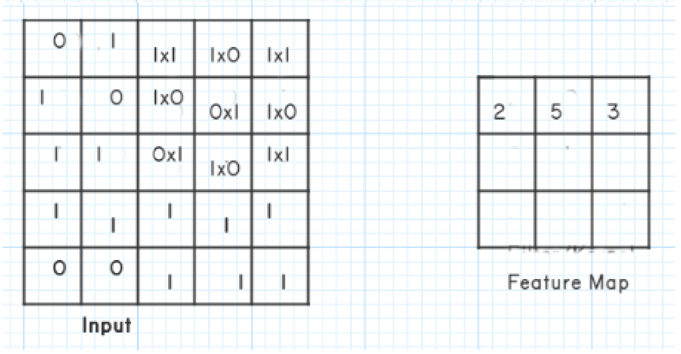
The input is the convolution layer while the convolution filter or Kernel is a 3×3 convolution filter. The convolution operation is performed below:

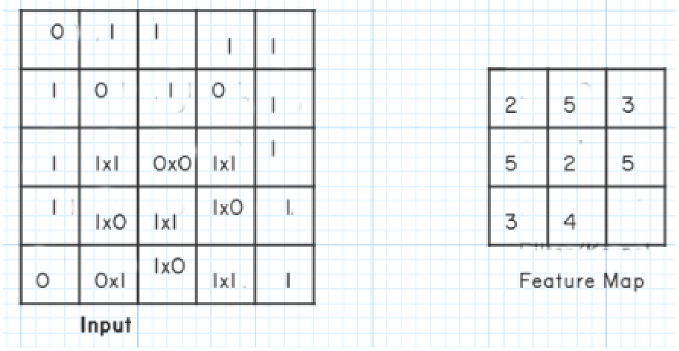


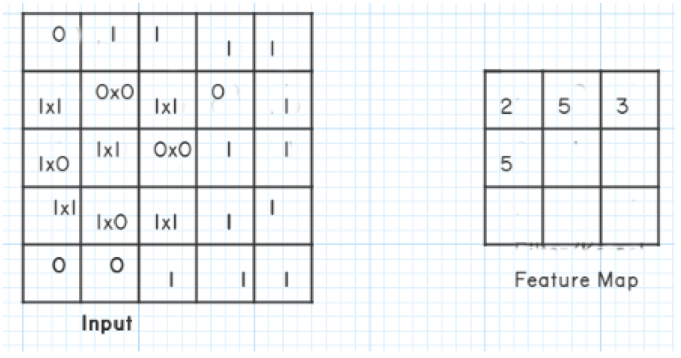


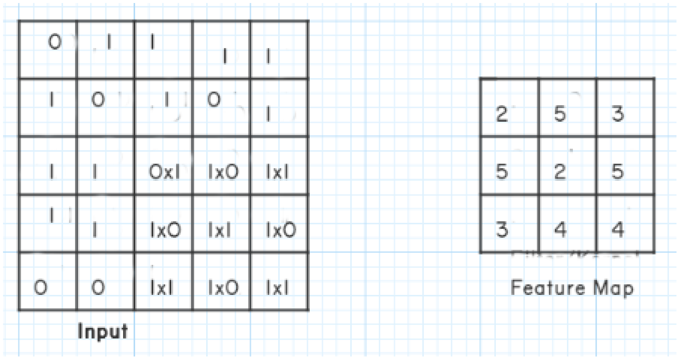


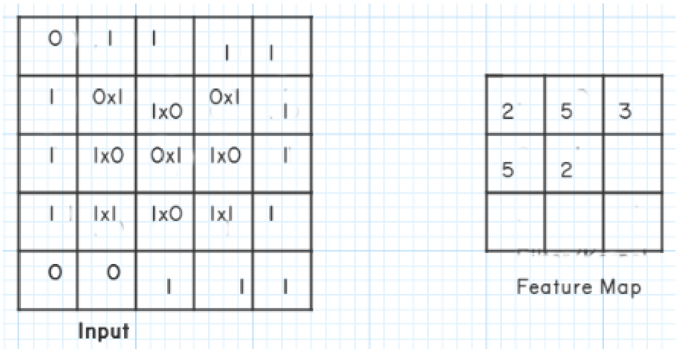




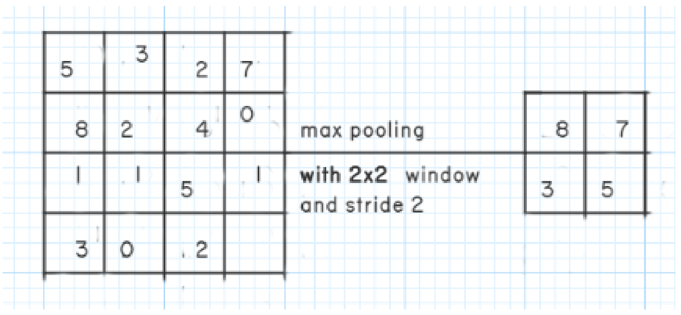








b. Pooling layer: This is the process of dimensionality reduction; it reduces the special size of input filter and data to have a minimized aggregate of parameters and processing time on the network to control overfitting.
Pooling units are derived by applications of functions such as max-pooling, average pooling, pooling has no parameters. Max pooling takes the maximum value in a pooling window, when the window is slide over the input, the output taken is the maximum value in the window. Max pooling is demonstrated below using a 2×2 window and a stride 2:



In CNN architectures, pooling is done using 2×2 windows, stride 2 with no padding, while convolution is done with 3×3 windows stride 1 and with padding depending on the required output dimension.

c. Fully Connected layer: This layer allows neuron in the layers to be interconnected to the neurons from the previous and next layer which enables the input filter to pass the matrix input from the previous layers flatten to the output layer. The output of the convolution layer and the pooling layers are 3D while the fully connected layer expects a 1D vector of numbers, the process of arranging the 3D volume of numbers into a 1D vector required by the fully connected layer is flattening.

CNN engages a weight splitting plan which yields a reduction in dimensionality of the input data to be learned. The derivations of forward and backward propagations will vary based on the layer the propagation is taking place.

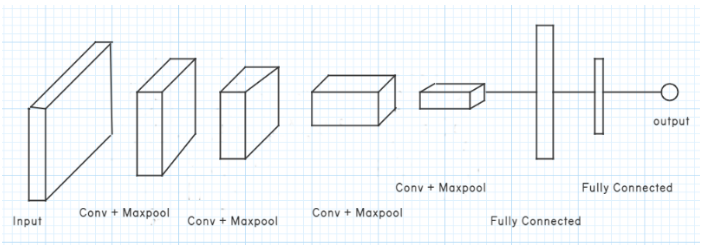


Image of 4 convolution plus pooling layers with 2 fully connected layers.

VI. NAÏVE BAYES APPROACH

The Bayesian approach is based on acyclic graph in which the nodes constitute the attributes and arcs constitute dependencies, the Bayesian network is based on the conditional probabilities of each node [105]. Each node is based on its parent's attribute and measures the attribute dependencies, a feature consists of attributes $\{A_1, A_2, \dots, A_i, \dots, A_n\}$ and $(a_1, a_2, \dots, a_i, \dots, a_n)$ are the attribute values for an instance E. The Bayesian network classifier can be defined as:

$$c(E) = \arg^{maxP}_{c\epsilon C}(c)P(a_1, a_2, \dots, a_n|c) \qquad (1)$$

Where C is the class variable constituted as the top node in the Bayesian network [105], c is the value that C takes for the instance E.

Naïve Bayes Classifier: Is a classifier in which all then attributes are naively assumed to be independent and given as:

$$P(E|c) = P(a_1, a_2, \dots, a_n|c) = \prod_{i=1}^{n} P(a_i|c) \qquad (2)$$

$$c(E) = \arg{}^{max}_{c\epsilon C} P(c) \prod_{i=1}^{n} P(a_i|C) \qquad (3)$$

Naïve Bayes classification model has computational clarity and efficiency because of its conditional probability. Levent et al. [105] introduced two approach to Naïve bayes Classifier as Intrusion Detection model:

Structure Extension: The extension of the Naïve Bayes structure is done with directed arcs to clearly represent attribute dependencies as a Bayesian Naïve network model [105]. The Tree-augmented Naïve bayes (TAN) is an extension of Naïve Bayes such that an attribute node will have at most one additional node than other class mode. The conditional mutual model for TAN learning algorithm is as follows:

$$I_p(X; Y|Z) = \sum_{x,y,z} P(X, Y, Z) \log \frac{P(x,y|z)}{P(x|z)P(y|z)} \qquad (4)$$

Where x, y, z $\in$ X, Y, Z respectively, $I_p(A_i;\ A_j|C)$ is calculated for each attribute pair and it depicts the weight of the arc's connection to attribute nodes A_i and A_j present on the Naïve Bayes network.

Feature Selection: This perspective separate irrelevant features that reduces the classification accuracy from the intrusion dataset. This method involves three approaches which are: Embedded, wrapper, and filter approaches. The Embedded approach is implanted in specific mining methods, the wrapper approach uses the response obtained from specific classifier to assess the value of the feature subset during their look through the entire data points. The filter approach depends solely on the widespread attribute of the training intrusion dataset.

Other approaches: L Kol et al. [105] introduces another approach to Naïve Bayes Intrusion Detection method where each attributes weight is seton according to the classification. In this method weighted Naïve Bayes model implements this approach:

$$c(E) = arg{}^{max}_{c\epsilon C} P(c) \prod_{i=1}^{n} P(a_i|C)^{w_i} \qquad (5)$$

Where w_i is the weight of attribute A_i, the approach is referred to as local approach which is developed on a subset of the training dataset, the method is founded on the assumption that a subset of a dataset has a higher classification accuracy than the entire dataset where the negative impact of the conditional independence is higher. Hence, the local learning allows more new models that are inserted in each other be formed. The data expansion approach in the local learning addresses the high variance problem in learning due to little available intrusion dataset by creating more instances with the similar pattern of the ultimate intrusion dataset distribution.

Naïve Bayes is a machine learning classifier assigns class labels to problem instances represented as vector values, the class values are finite set. All naïve Bayes classifiers assume that the value of a particular feature is independent of the other value of any feature, given the class variable.

The Gaussian distribution is common and important [81] in probability calculations, the theory describes errors and is given as:

$$P(x) = \frac{1}{\delta\sqrt{2\pi}} e^{\frac{(x-\mu)^2}{2\delta^2}}$$

Where μ is the average and δ is the standard deviation, to calculate μ and δ values using the formula given as:

$$\mu = \frac{\sum_{i-1}^{n} xi}{n}$$

$$\delta^2 = \frac{\sum_{i=1}^{n} (xi - \mu)^2}{n-1}$$

The Bayesian theory considers [81] how to choose the best class labels based on probabilities and misclassified losses. Bayesian theory is given as follows:

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

$P(x|c)$ is the conditional probability of a record x relative to the class label c, and $P(x)$ is the evidence factor used for normalization; The evidence X can be dispatched [83] into pieces of evidence, say $x_1, x_2, \ldots, x_n$ relative to the attributes $X_1, X_2, . . . , X_n$, respectively. Naïve Bayesian theory works under the assumption that these attributes are independent, their combined probability is given as follows:

$$P(c_i|x) = \frac{P(a_1|c_i) * P(a_2|c_i).,,,.P(a_n|c_i) * P(c_i)}{P(x)}$$

There is no need to explicitly compute the denominator $P(x)$ because it is determined by the normalization condition. Therefore, it is sufficient to compute for each class c_i its likelihood.

$P(a_1|c_i) * P(a_2|c_i).,,,.P(a_n|c_i) * P(c_i)$

To classify any new object characterized by its attribute's values $x_1, x_2, \ldots, x_n$.

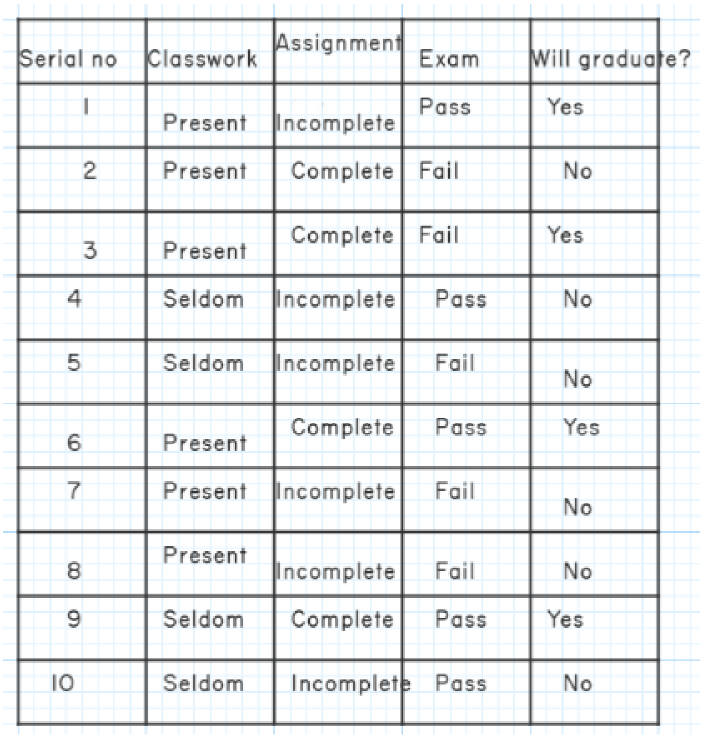
The Gaussian Bayes Classifier is applied to the dimensionality reduced test dataset to classify the category of each record. The conditional probability [84] $P(x_i|c)$ of each attribute is calculated according to

$$P(x_i|c) = \frac{1}{\sqrt{2\pi\sigma_{c,i}}} e^{\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)}$$

The prior class probability $P(c)$ of records belonging to normal and anomaly are calculated separately. The prior probability of recording with normal and anomaly is computed and the category of the record with the large prior probability is selected as the detection result of the record.

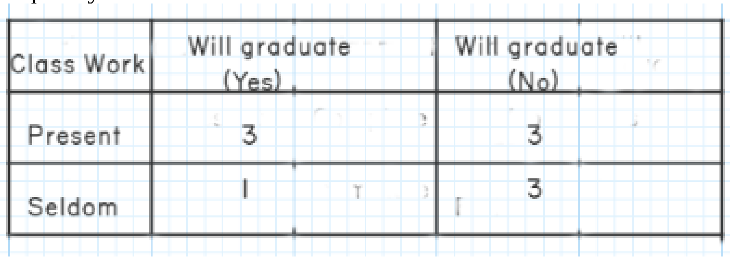
The Gaussian Naïve Bayes is a likelihood classifier [82] that estimates a scaled probability density function of each class with a Gaussian distribution centered on the mean of that class. The variance and covariance between training class attributes are used to classify the data, the model also assumes the output responses of a given class fit a normal distribution.

Suppose we want to find if students passed and have the following dataset.



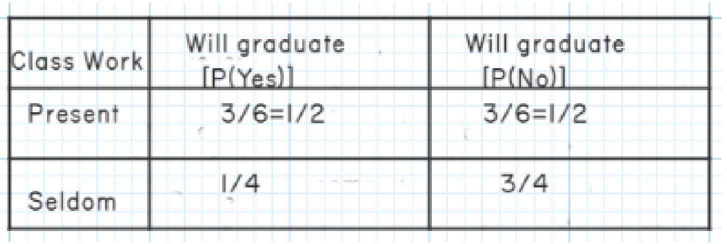
Serial no	Classwork	Assignment	Exam	Will graduate?
1	Present	Incomplete	Pass	Yes
2	Present	Complete	Fail	No
3	Present	Complete	Fail	Yes
4	Seldom	Incomplete	Pass	No
5	Seldom	Incomplete	Fail	No
6	Present	Complete	Pass	Yes
7	Present	Incomplete	Fail	No
8	Present	Incomplete	Fail	No
9	Seldom	Complete	Pass	Yes
10	Seldom	Incomplete	Pass	No

Frequency table of classwork



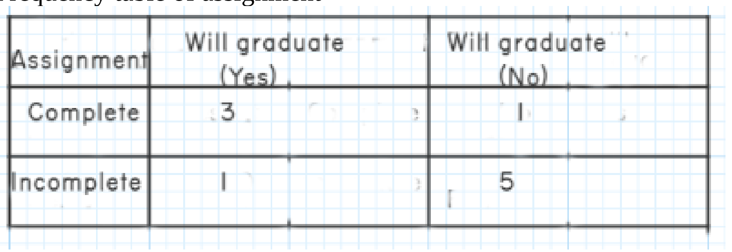
Class Work	Will graduate (Yes)	Will graduate (No)
Present	3	3
Seldom	1	3

Likelihood table of classwork



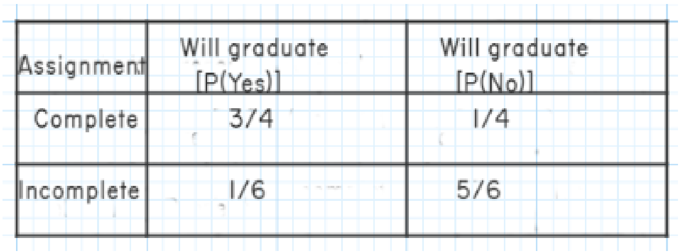
Class Work	Will graduate [P(Yes)]	Will graduate [P(No)]
Present	3/6=1/2	3/6=1/2
Seldom	1/4	3/4

Frequency table of assignment



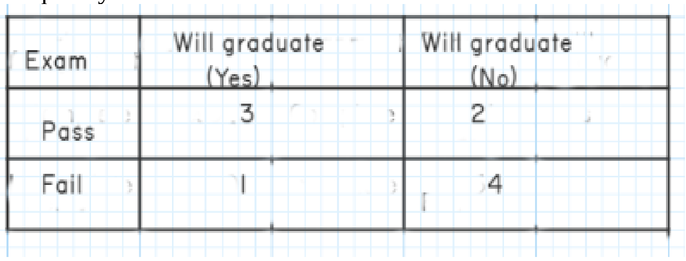
Assignment	Will graduate (Yes)	Will graduate (No)
Complete	3	1
Incomplete	1	5

Likelihood table of assignment



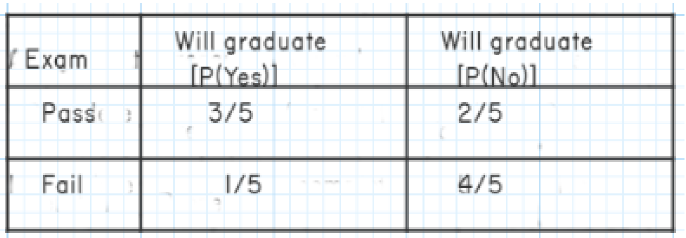
Assignment	Will graduate [P(Yes)]	Will graduate [P(No)]
Complete	3/4	1/4
Incomplete	1/6	5/6

Frequency table of Exam



Exam	Will graduate (Yes)	Will graduate (No)
Pass	3	2
Fail	1	4

Likelihood table of Exam



Exam	Will graduate [P(Yes)]	Will graduate [P(No)]
Pass	3/5	2/5
Fail	1/5	4/5

Out problem has three predictors for X, the posterior probability

$$P(Yes|X) = P(Seldom|Yes) * (Incomplete|Yes) * (Pass|Yes) * P(Yes)$$

$$= {}^{1}/_{4} \times {}^{1}/_{6} \times {}^{3}/_{5} \times {}^{4}/_{10}$$
$$= 0.01$$

$P(No|X)$ would be:

$$P(No|X) = P(Seldom|No) * P(Incomplete|No) * P(Pass|No) * P(No)$$

$$= {}^{3}/_{4} \times {}^{5}/_{6} \times {}^{2}/_{5} \times {}^{6}/_{10}$$
$$= 0.15$$

Hence, as the posterior probability $P(No|X)$ is higher than the posterior probability $P(Yes|X)$, our Seldom, Incomplete, Pass will have a 'No' in the 'Will graduate?' section.

3.8.1 Advantages of Gaussian Naïve Bayes Classifier:

1. When the assumption of independent predictors is true, the classifier performs better compared to other models.
2. Gaussian naïve Bayes does not require much training time to estimate the test data.
3. It is easy to implement.

VII. OPEN CHALLENGES

A major challenge in Intrusion Detection Model is the intrinsic randomness in results of the machine learning training models. The essence of learning rate schedule in machine learning models is to achieve convergence at each training period, however, results from training the same dataset differ at each period even when the dataset is trained with the same model configuration and hardware. Machine Learning models usually regenerates its estimated path by putting into cognizance the entire dataset perspective, this leads to a reproduced training period functioning marginally from the initial assumptions even with same set of data from antecedent instance. This can yield a slight change in prediction and evaluation, update in libraries of machine learning models and the Integrated Development Environment are some factors that can lead to variations in results of Machine Leaning Models as Intrusion Detection Systems.

VIII. CONCLUSION

This article is an overview of Stochastic Gradient Descent Classifier, Linear Discriminant Analysis, Deep learning, and Naïve Bayes which are machine learning techniques and approaches to Network Intrusion Detection. It also discusses their properties, characteristics, and mode of operation. Open challenges that will lead to future work is also highlighted.

REFERENCES

[1] C.L Phillip Chen, Chun-Yang Zhang *Data-intensive applications; challenges, techniques, and technologies: A survey on Big Data, 2014.*
[2] K. Krishnan *Data Warehousing in the age of Big Data, 2013*.
[3] Suresh Lakavath, Ramlal Naik.L. *A Big Data Hadoop Architecture for Online Analysis, 2015*.
[4] E. Goldin, D.Feldman, G.Georgoulas, M.Castana and G.Nikakopoulas *Cloud Computing for Big Data Analytics in the process Control Industry, 2017.*
[5] L. Breiman *"Random forests" Mach. Learn., vol. 45, no. 1, pp, 5-32, 2001.*
[6] Srinivas Mukkamala, Andrew Sung and Ajith Abraham *Cyber Security Challenges: Designing Efficient Intrusion Detection Systems and Antivirus Tools, 2005.*
[7] Shaik Akbar, K. Nageswara Rao, J.A. Chandulal *Intrusion Detection System Methodologies Based on Data Analysis, 2010*
[8] Ajith Abraham, Crina Grosan. Yuechui Chen *Cyber Security and the Evolution of Intrusion Detection Systems, 2005*
[9] Julien Corsini *Analysis and Evaluation of Network Intrusion Detection Methods to Uncover Data Theft, 2009*
[10] G.Nikhitta Reddy, G.J.Ugander Reddy *A study of Cyber Security Challenges and its Emerging Trends on Latest Technology, 2014*
[11] G. McGraw and G. Morrisett *Attacking malicious code: A report to the infosec research council. IEEE Software, 17(5):33-44, 2000.*
[12] Kutub Thakur, Meikang Qui, Keke Gai, Md Liakat Ali *An Investigation on Cyber Security Threats and Security Models, 2015*
[13] https://www.globalsign.com/en/blog/cybersecurity-trends-and-challenges-2018/
[14] Sophoslabs 2018 Malware Forecast *https://www.sophos.com/en-us/en-us/medialibrary/PDFs/technical-papers/malware-forecast-2018.pdf?la=en*
[15] A third of Americans live in a household with three or more smartphones, 2017. *http://www.pewresearch.org/*
[16] https://www.statista.com
[17] Mohammed J. Aljebreen *Towards Intelligent Intrusion Detection Systems for Cloud Computing, 2018*
[18] Zulaiha Ali Othman, Lew Mei Theng, Suhaila Zainudin, Hafiz Mohd Sarim *Great Deluge Algorithm Feature Selection for Network Intrusion Detection. 2013*
[19] Mafarja, M. and S. Abdullah, 2011. *Modified great deluge for attribute reduction in rough set theory. Fuzzy Systems and Knowledge Discovery, pp: 1464-1469. 2011*
[20] Chebrolu, S., A. Abraham, J.P. Thomas, 2005. *Feature Deduction and Ensemble Design of Intrusion Detection Systems. Journal of Computers and Security, 24(4): 295-307.*
[21] Zainal, A., M. Maarof, S. Shamsuddin, 2007. *Feature* selection *using rough-DPSO in anomaly intrusion detection. Computational Science and Its Applications–ICCSA, pp: 512-524.*
[22] Pratik N., Neelakantan, C. Nagesh M. Tech, 2011. *"Role of Feature Selection in Intrusion Detection Systems for 802.00 Networks"*

International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN) 1(1).

[23] Amr S Abed, T Charles Clancy, and David S Levy. *Applying bag of system calls for anomalous behavior detection of applications in linux containers. In Globecom Workshops (GC Wkshps), 2015 IEEE, pages 1–5. IEEE, 2015.*

[24] S Barlev, Z Basil, S Kohanim, R Peleg, S Regev, and Alexandra Shulman-Peleg. *Secure yet usable: Protecting servers and linux containers. IBM Journal of Research and Development, 60(4):12–1, 2016.*

[25] Qiang Wang Vasileios Megalooikonomou *A Clustering Algorithm for Intrusion Detection, 2005.*

[26] Martin Roesch *Snort — Lightweight Intrusion Detection for Networks, Proceedings of LISA '99: 13th Systems Administration Conference Seattle, Washington, USA, November 7–12, 1999*

[27] Nahla Ben Amor, Salem Benferhat, Zied Elouedi *Naive Bayes vs Decision Trees in Intrusion Detection Systems, ACM Symposium on Applied Computing, 2004.*

[28] Quinlan, J. R. *C4.5, Programs for machine learning, Morgan Kaufmann San Mateo Ca, 1993.*

[29] http://www.netresec.com/?page=PcapFiles

[30] http://www.cybersecurity.unsw.adfa.edu.au/ADFA%20IDS%20Datasets/

[31] http://bit.ly/csic-2010-http-dataset_csv

[32] Haydar Teymourlouei, Lethia Jackson, 2017 How big data can improve cyber security, Proceedings of the 2017 International Conference on Advances in Big Data Analytics, pp: 9-13.

[33] Miltiadis Allamanis, Earl T. Barr, Premkumar Devanbu, Charles Sutton *A survey of Machine Learning for Big Code and Naturalness, 2018.*

[34] Robert Mitchell and Ing-Ray Chen *A Survey of Intrusion Detection Techniques for Cyber-Physical Systems, 2014.*

[35] Robert A. Bridges, Tarrah R. Glass-Vanderlan, Michael D. Jannacone and Maria S. Vincent *A survey of Intrusion Detection Systems Leveraging Host Data, 2019*

[36] Abiodun Ayodeji, Tong-Kuo Liu, Nan Chao, Li-qun Yang *A new perspective towards the development of robust data-driven intrusion detection for industrial control sytems, 2020.*

[37] Hadeel Alazzam, Ahmad Sharieh, Khair Eddin Sabri *A Feature Selection Algorithm for Intrusion Detection System Based on Pigeon Inspired Optimizer, 2020.*

[38] Chaouki Khammassi, Saoussen Krichen *A NSGA2-LR wrapper approach for feature selection in network intrusion detection, 2020.*

[39] Faezah Hamad Almasoudy, Wathiq Laftah Al-Yaseen, Ali Kadhum Idrees *Differential Evolution Wrapper Feature Selection for Intrusion Detection System, 2019.*

[40] Amol Borkar, Akshay Donode, Anjali Kumari *A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and Protection System (IIDPS), 2017.*

[41] Said Ouiazzane, Malika Addou, Fatimazahra *A Multi-Agent Model for Network Intrusion Detection, 2019.*

[42] Manoj s. koli, Manik K. Chavan *An Advanced method for detection of botnet traffic using Internal Intrusion Detection, 2017.*

[43] Azzedine Boukerche, Lining Zheng, Omar Alfandi *Outlier Detection: Methods, Models and Classification, 2020.*

[44] Srinivas Mukkamala, Guadalupe Janoski, Andrew Sung *Intrusion Detection Using Neural Networks and Support Vector Machines, 2002.*

[45] Heba F. Eid, Ashraf Darwish, Aboul Ella Hassanien, and Ajith Abraham *Principle Components Analysis and Support Vector Machine-based Intrusion Detection System, 2010.*

[46] Fidalcastro. A, Baburaj. E. *Sequential Pattern Mining for Intrusion Detection System with Feature Selection on Big Data, 2017.*

[47] Shi-Jie Song, Zunguo Huang, Hua-Ping Hu and Shi-Yao Jing. *A Sequential Pattern Mining Algorithm for Misuse Intrusion Detection, 2004.*

[48] Pakinam Elamein Abd Elaziz, Mohamed Sobh, Hoda K. Mohamed *Database Intrusion Detection Using Sequential Data Mining Approaches, 2014.*

[49] Wenke Lee and Salvatore J. Stolfo Data Mining Approaches for Intrusion Detection, 1998.

[50] Zhendong Wu, Jingjing Wang, Liqing Hu, Zhang Zhang, Han Wu *A network intrusion detection method based on semantic Re-encoding and deep learning, 2020.*

[51] Arwa Aldweesh, Abdelouahid Derhab, Ahmed Z. Emam *Deep Learning Approaches for Anomaly-Based Intrusion Detection Systems: A survey, Taxonomy and Open Issues, 2020.*

[52] Omar Y. Al-Jarrah, Yousof Al-Hammdi, Paul D. Yoo, Sami Muhaidat, Mahmoud Al-Qutayri *Semi-supervised multi-layered clustering model for intrusion detection, 2017.*

[53] Naila Belhadj Aissa, Mohamed Guerroumi *A Genetic Clustering Technique for Anomaly-Based Intrusion Detection Systems, 2015.*

[54] Mohammad Khubeb Siddiqui and Shams Naahid *Analysis of KDD CUP 99 Dataset using Clustering based Data Mining, 2013.*

[55] Joshua Oldmeadow, Siddarth Ravinutaka and Christopher Lechie *Adaptive Clustering for Network Intrusion Detection, 2004.*

[56] S.Sathya Bama, M.S.Irfan Ahmed, A.Saravanan *Network Intrusion Detection using Clustering: A Data Mining Approach, 2011.*

[57] M. Mazhar Rathore, Anand Paul, Awais Ahmad, Seungmin Rho, Muhammad Imran, Mohsen Guizani *Hadoop Based Real-Time Intrusion Detection for High-speed Networks, 2016.*

[58] Sanraj Rajendra Bandre, Jyoti N. Nandimath *Design Consideration of Network Intrusion Detection System using Hadoop and GPDPU, 2015.*

[59] Sanjai Veeti and Qigang Gao *Real-time Netwok Intrusion Detection Using Hadoop-Based Bayesian Classifier, 2014.*

[60] Sandhya Peddabachigari, Ajith Abraham, Johnson Thomas *Intrusion Detection Systems Using Decision Trees and Support Vector Machines, 2007.*

[61] Shilpashree. S, S. C. Lingareddy, Nayana G Bhat, Sunil Kumar G *Decision Tree: A machine Learning for Intrusion Detection, 2019.*

[62] Manish Kumar, Dr. M. Hanumanthappa, Dr. T. V. Suresh Kumar *Intrusion Detection System Using Decision Tree Algorithm, 2012.*

[63] Christopher Kruegel and Thomas Toth *Using Decision Trees to Improve Signature-Based Intrusion Detection, 2003.*

[64] Kajal Rai, M. Syamala Devi, Ajay Guleria *Decision Tree Based Algorithm for Intrusion Detection, 2015.*

[65] Jianwu Zhang, Yu Ling, Xingbing Fu, Xiongkun Yang, Gang Xiong, Rui Zhang *Model of Intrusion Detection System Based on the Integration of Spatial-Temporal Features, 2019.*

[66] Iman Sharafaldin, Arash Habibi Laskkari and Ali A. Ghorbani *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization, 2018.*

[67] Ajay Shrestha and Ausif Mahmood *Review of Deep Learning Algorithms and Architectures, 2019.*

[68] Henry Friday Nweke, Ying Wah The, Mohammed Alli Al-garadi, Uzoma Rita Alo *Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges, 2018.*

[69] Tuan A Tang, Lotfi Mhandi, Des McLernon, Syed Ali Raza Zaidi and Mounir Ghogho *Deep Learning Approach for Network Intrusion Detection in Software Defined Networking, 2016.*

[70] Quamar Niyaz, Weiqing Sun, Ahmad Y. Javaid, and Mansorrr Alam *A Deep Learning Approach for Network Intrusion Detection System, 2016.*

[71] R. Vinayakumar, Mamoun Alazab, K.P. Soman, Prabaharan Poornachandran, Ameer Al-Nemrat and Sitalakshmi Venkattraman *Deep Learning Approach for Intelligent Intrusion Detection System, 2019.*

[72] Godze Karatas, Onder Demir, Ozgur Koray sahingoz *Deep Learning in Intrusion Detection Systems, 2018.*

[73] Zheng Wang *Deep Learning-Based Intrusion Detection with Adversaries, 2018.*

[74] Khaled Alrawashdeh, Carla Purdy *Toward an online Anomaly Intrusion Detection System Based on Deep Learning, 2016.*

[75] Chaoqun Hong, Jun Yu, Jian wan, Dacheng Tao and Meng Wang *Multimodal Deep Autoencoder for Human Pose Recovery, 2015.*

[76] Fahimeh Farahnakian, Jukka Heikkonen *A Deep Auto-Encoder based Approach for Intrusion Detection System, 2018.*

[77] Sydney Mambwe Kasongo, Yanxia Sun *A Deep Long Short-Term Memory based classifier for Wireless Intrusion Detection System, 2020.*

[78] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, Yunpeng Wang *Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, 2015.*

[79] Vinayakumar R, Soman KP and Prabaharan Poornachandran *Applying Convolutional Neural Network for Network Intrusion Detection, 2017.*

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 04, April-2021

[80] Hui Wang, Zijian Cao and Bo Hong *A network intrusion detection system based on convolutional neural network, 2020.*

[81] Abdul Fadil, Imam Riadi and Sukma Aji *A Novel CCoS Attack Detection Based on Gaussian Naïve Bayes, 2017.*

[82] Farmaz Gharibian and Ali A. Ghorbani *Comparative Study of Supervised Machine Learning Techniques for Intrusion Detection.*

[83] Nahla Ben Amor, Salem Benferhat and Zied Elouedi *Naïve Bayesian Networks in Intrusion Detection Systems.*

[84] Bing Zhang, Zhiyang Liu, Yanguo Jia, Jiadong Ren and Xiaolin Zhao *Network Intrusion Detection Method Based on PCA and Bayes Algorithm, 2018.*

[85] Efficient L_1 Regularized Logistic Regression *Su-In Lee, Honglak Lee, Pieter Abbeel and Andrew Y. Ng, 2006.*

[86] Active Learning for Logistic Regression: An Evaluation *Andrew I. Schein and Lyle H. Ungar, 2007.*

[87] Towards Intelligent Intrusion Detection Systems for Cloud Computing *Mohammed J Aljebreen, William Allen, PhD., 2018.*

[88] Andreas Fuchsberger *Intrusion Detection Systems and Intrusion Detection Systems, Information Security Technical Report 135-139 (10), 2005.*

[89] Nikhil Tripathi, Mayank Swarnkar and Neminath Hubballi *DNS Spoofing in Local Networks in Local Networks Made Easy, 2017.*

[90] Alan Dahgwo Yein, Cheng-Yeh Chen, Te-Cheng Hsu, Wen-Shyong Hsieh, and Jiun-An Lin *Attack Wireless Sensor Network using Compromised Key Redistribution, 2013.*

[91] Yuang Zhang, Chunxiang Xu, Nan Cheng, and Xuemin (Sherman) Shen *Secure Encrypted Data Deduplication for Cloud Storage against Compromised Key Servers, 2019.*

[92] Xanthopoulos, Petros, Pardalos, Panos M, Traflis, Theodre B. *Robust Data Mining || Linear Discriminant Analysis, 10.1007/978-1-4419-9878-1(Chapter 4), 27-33, 2013.*

[93] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Abdoul Ella Hassanien *Linear Discriminant Analysis: A Detailed Tutorial, Al Communications, 30(2), 169-190, 2017.*

[94] Alok Sharma, Kuldip K. Paliwal *Linear Discriminant Analysis for the small sample size problem: an overview, 2014.*

[95] Riaz Ullah khan, Xiaosong Zhang, Mamoun Alazab, and Rajesh Kumar *An Improved Convolutional Neural Network Model for Intrusion Detection in Networks, 2019.*

[96] Rajesh Kumar, Zhang Xiaosong, Riaz Ullah Khan, Ijaz Ahad, and Jay Kumar *Malicious Code Detection based on Image Processing Using Deep Learning, 2018.*

[97] Hyson-Joong Yoo *Deep Convolution Neural Networks in Computer Vision: a Review, 2015.*

[98] Prudhvi Thirumalaraju, Manoj Kumar Kanakasabapathy, Charles L. Bormann, Raghav Gupta, Rohan Pooniwala, Hemant Kandula, Irene Souter, Irene Dimitriadis, and Hadi Shafiee *Evaluation of Deep Convolutional Neural Networks in Classifying Human Embryo Images based on their morphological Quality, 2021.*

[99] Mohammad Amang Syarifudin, Dian Candra Rini novitasari, Faridawaty Marpaung, Noor Wahyudi, Dian Puspita Hapsari, Endang Supriyati, Yuniar Farida, Faris Muslihul Amin, RR. Diah Nugraheni, Ilham, Rinda Nariswari, and Fajar Setiawan *Hotspot Prediction Using ID Convolutional Neural Network, 2021.*

[100] Alam Ahmad Hidayat, Kartika Purwandari, Tjeng Wawan Cenggoro, and Bens Pardamean *A Convolutional Neural Network-based Ancient Sudanese Character Classifier with Data Augumentation, 2021.*

[101] Keunwoo Choi, Gyorgy Fazekas, and Mark Sandler *Explaining Deep Convolutional Neural Networks on Music Classificatio, 2016.*

[102] Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg *Understanding Convolutional Neural Networks for Text Classification, 2020.*

[103] Vinayakumar R, Soman KP and Prahaharan Poormachandran *Appling Convolutional Neural Network for Network Intrusion Detection, 2017.*

[104] Mrutyunjaya Panda and Manas Ranjan Patra *Network Intrusion Detection Using Naïve Bayes, 2007*

[105] Levent koc, Thomas A. Mazzuchi, and Shahram Sarkani *A network intrusion detection system based on a Hidden Naïve Bayes Multiclass Classifier, 2012.*

[106] Richard A. Kemmerer and Giovanni Vigna *Intrusion Detection: A Brief History and Overview, 2002.*

[107] Herve Debar. *An Introduction to Intrusion Detection Systems, IBM Research, Zurich Research Laboratory, Saumerstrasse 4, CH-8803 Russchlikon, Switzerland, 2000.*

[108] Jeffrey R. Yost *The March of IDES: The Advent and Early History of Intrusion Detection Expert Systems, IEEE Annals of the History of Computing, (), 1-1, 2015.*