ML Model Comparison for Network Security

Vaibhavi Mutya[Ω]
Bourns College of Engineering
University of California,
Riverside Riverside, CA, USA
vmuty002@ucr.edu

Aira Bhaima Shrestha[ɣ]
Bourns College of Engineering
University of California,
Riverside Riverside, CA, USA
ashre009@ucr.edu

Akhilesh Reddy Gali[β]
Bourns College of Engineering
University of California,
Riverside Riverside, CA, USA
agali030@ucr.edu

ABSTRACT[Ω]

Network security is a growing area with rapidly evolving spam attack patterns and threats. In order to be able to detect rapidly evolving threats one needs robust generalizable solutions that can learn complex data distributions and use them to make further spam/ham inferences. Therefore, Machine Learning can play a key role in learning anomalous data distributions in data and use them to make threat inferences into the future. In this paper, we implement PCA along with 3 classification algorithms on a network security dataset and compare model performance. We achieve a maximum weighted F1 score of 96.19% with a RF classifier.

KEYWORDS[Ω]

Principal Component Analysis, Logistic Regression, Naive Bayes, Random Forest, F1 Score

1. Introduction[Ω]

The well known DoS and DDoS attacks is to stop the computer or the network from being accessible to its users either by consuming the bandwidth of the network or the resources of the host. DDoS attacks happen on Internet Control Message Protocol version 6 (ICMPv6).

There are different kinds of attacks on IPv6 networks. Among them the most common attacks are Denial of service (DoS) and distributed denial of service (DDoS). These attacks are thorny and a grave problem of today's Internet, resulting in economic damages for organizations and individuals.To avoid these attacks in IPv4 network the network administrators blocked ICMPv4 by dropping all its messages. However, in an IPv6 network this method can not be used due to the ICMPv6 highly relevant for the correct functioning of IPv6 nodes and networks. Therefore, the only way to avoid ICMPv6 vulnerabilities is to deploy a detection (and possibly prevention) system within the network to monitor and detect the abnormal behaviors of ICMPv6 that lead to detect the attacks against it. [1]

2. Related Work[Ω]

2.1 Types of Dataset

Datasets for such applications are generated in 2 ways. One of which is an Artificially generated dataset. These are datasets that are crafted manually by network security experts using automated scripts mimicking all possible attacks. Due to this, they can include all types of attacks. However, since they are synthetically created they do not mimic the real world exactly. The other dataset are the ones collected from different organizations. These are closer to the real world but may not include all variants of the attacks. [1]

2.2 Dataset Representation

Another important aspect of datasets is how they are represented. Datasets are represented in 2 ways: Flow based and packet based. Flow based datasets represent the packets that have common characteristics in one record. They create small sized datasets which enable lightweight detection systems to be built with presumably less space and time complexity. They also pay attention to privacy details by removing network details. However, due to their small size they possess fewer details, more processing time is needed to construct flows and they are not suitable for detecting attacks which depend on the packet's payload.

Packet based dataset represents the traffic by monitoring the network and capturing the full packets including the payload and header. These packets have several details available and no processing time is needed for flow construction etc. However, they are large in size making algorithmic design challenging, expose sensitive data and signatures matching are impossible in the cases of encrypted payload. [1]

2.3 Dataset Overview

In this work, we use the Kaggle Intrusion Detection dataset. This dataset has a total of 25192 rows & 42 columns in the dataset. Of the 42 columns 38 numerical, 3 are categorical features and 2 label classes: 11743 anomaly and 13449 normal.

3. Proposed Methods

On the Kaggle Intrusion Detection dataset we preprocess data and implement 4 classification algorithms to solve the problem. These are: Logistic Regression, Random Forest and Naive Bayes.

3.1 PCA and Logistic Regression$^{\Omega}$

3.1.1 Overview of Principal Component Analysis

Principal component analysis is a method to find an orthogonal basis for our data such that variance of our data projected onto this subspace governed by the orthogonal basis is maximized. We can frame it as a Variance Optimization Problem [3]:

$$\max_{\mathbf{v}} J(\mathbf{v}) = \mathbf{v}^T \boldsymbol{\Sigma} \mathbf{v} - \alpha(\mathbf{v}^T\mathbf{v} - 1) - \sum_{i=1}^{j-1} \beta_i (\mathbf{u}_i^T \mathbf{v} - 0)$$

Solving for which we obtain:

$$\boldsymbol{\Sigma} \mathbf{v} = \alpha \mathbf{v}$$

which is the same as the eigendecomposition of the centered covariance matrix for our initial data. Now, choosing the dimensions with maximum variance is equal to selecting eigenvectors corresponding to the largest eigenvalues.

3.1.2 Implementing Principal Component Analysis for Dimensionality Reduction

Since there are 8 categorical features in our dataset we use one-hot encoding to convert them into binary features, this results in a total of 118 features. Thereafter, we perform PCA on this 25192 x 118 dimensional dataset.

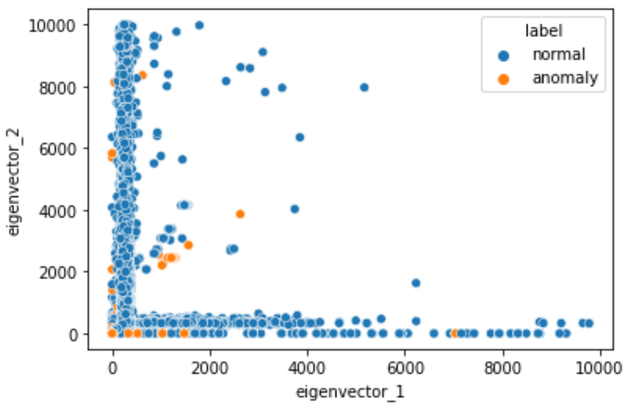
Zaki et. al propose a linear algebra based algorithm for implementing PCA [3]. It involves centering our input data, computing it's covariance matrix and then performing an eigendecomposition.

```
**PCA** $(\mathbf{D}, \alpha)$:
1 $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i$ // compute mean
2 $\overline{\mathbf{D}} = \mathbf{D} - \mathbf{1} \cdot \boldsymbol{\mu}^T$ // center the data
3 $\boldsymbol{\Sigma} = \frac{1}{n}\left(\overline{\mathbf{D}}^T \, \overline{\mathbf{D}}\right)$ // compute covariance matrix
4 $(\lambda_1, \lambda_2, \ldots, \lambda_d) = \text{eigenvalues}(\boldsymbol{\Sigma})$ // compute eigenvalues
5 $\mathbf{U} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_d \end{pmatrix} = \text{eigenvectors}(\boldsymbol{\Sigma})$ // compute eigenvectors
6 $f(r) = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$, for all $r = 1, 2, \ldots, d$ // fraction of total variance
7 Choose smallest $r$ so that $f(r) \geq \alpha$ // choose dimensionality
8 $\mathbf{U}_r = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_r \end{pmatrix}$ // reduced basis
9 $\mathbf{A} = \{\mathbf{a}_i \mid \mathbf{a}_i = \mathbf{U}_r^T \bar{\mathbf{x}}_i, \text{for } i = 1, \ldots, n\}$ // reduced dimensionality data
```

Thereafter, we can use a scree plot to select the no. of principal components required using:

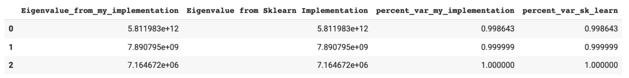
$$f(r) = \frac{\sum_{i=1}^{r} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$$

where f(r), fraction of variance required was set to 95% in our case. However, we obtained 99.9% with just the first 2 principal components. Here is a 2-D plot of our original data represented by the first two eigenvectors.

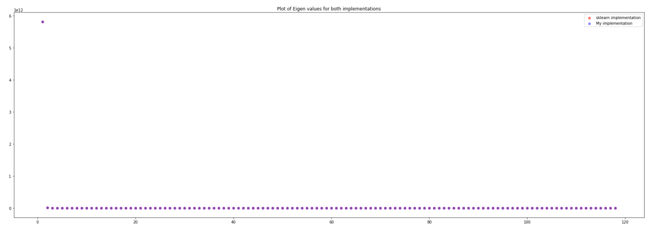


3.1.3 Comparison with sklearn PCA

Implementation of PCA using Zaki et. al algorithm [3] is almost identical to sklearn PCA implementation. Here is a dataframe with eigenvalues and percentage of total variance explained for the top 3 eigenvectors, we can observe that the eigenvalues are identical. Moreover, in the plot below the eigenvalue raw values also seem to identically overlap.



	Eigenvalue_from_my_implementation	Eigenvalue from Sklearn Implementation	percent_var_my_implementation	percent_var_sk_learn
0	5.811983e+12	5.811983e+12	0.998643	0.998643
1	7.890795e+09	7.890795e+09	0.999999	0.999999
2	7.164672e+06	7.164672e+06	1.000000	1.000000



3.1.4 Overview of Logistic Regression

The idea with Logistic Regression is to Identify P(Y|X). We do this using a linear model and then squishing it to 0-1 using a sigmoid/logistic function. This gives us a resulting model:

$$p(Y=1|X) = \frac{e^{\beta^T X}}{1+e^{\beta^T X}}$$
$$p(Y=0|X) = 1-\sigma(\beta^T X)$$

From here, we can compute the Log Likelihood of data and do MLE. This is equivalent to minimizing the cross-entropy error. [4]

$$ln(L(\beta)) = \sum_i y_i \, log(\sigma(\beta^T x_i)) + (1-y_i) \, log(\sigma(-\beta^T x_i))$$

We can also perform regularization using the logistic model by imposing norm constraints on the weights of the linear model. However, given we use PCA, we will not be going the regularization route as we will implement logistic regression on the first two principal components of data earlier identified.

3.1.5 Implementation of Logistic Regression

Since logistic regression doesn't have a closed form solution we can solve it via iterative optimization. Here, we use stochastic gradient descent to implement it. SGD uses line search with gradient of the weights at each step to move in the direction of the global optimum.

$$w_{k+1} = w_k + \alpha \nabla \, ln(L(w))$$

The algorithm we use here is the one proposed in Zaki et al [3]:

```
LOGISTICREGRESSION-SGA ($\mathbf{D}, \eta, \epsilon$):
foreach $\mathbf{x}_i \in \mathbf{D}$ do $\tilde{\mathbf{x}}_i^T \leftarrow \begin{pmatrix} 1 & \mathbf{x}_i^T \end{pmatrix}$ // map to $\mathbb{R}^{d+1}$
$t \leftarrow 0$ // step/iteration counter
$\tilde{\mathbf{w}}^0 \leftarrow (0, \ldots, 0)^T \in \mathbb{R}^{d+1}$ // initial weight vector
repeat
    $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}}^t$ // make a copy of $\tilde{\mathbf{w}}^t$
    foreach $\tilde{\mathbf{x}}_i \in \tilde{\mathbf{D}}$ in random order do
        $\nabla(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i) \leftarrow \left(y_i - \theta(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i)\right) \cdot \tilde{\mathbf{x}}_i$ // compute gradient at $\tilde{\mathbf{x}}_i$
        $\tilde{\mathbf{w}} \leftarrow \tilde{\mathbf{w}} + \eta \cdot \nabla(\tilde{\mathbf{w}}, \tilde{\mathbf{x}}_i)$ // update estimate for $\tilde{\mathbf{w}}$
    $\tilde{\mathbf{w}}^{t+1} \leftarrow \tilde{\mathbf{w}}$ // update $\tilde{\mathbf{w}}^{t+1}$
    $t \leftarrow t+1$
until $\|\tilde{\mathbf{w}}^t - \tilde{\mathbf{w}}^{t-1}\| \le \epsilon$
```

3.1.6 Comparison with sklearn

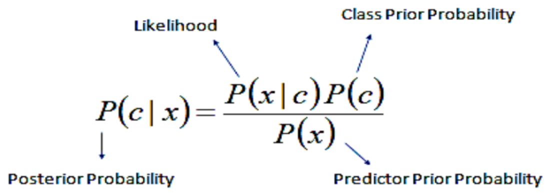
Tuning the optimization algorithm with different values of step size we obtain almost similar parameters as sklearn with a step size of 0.1.

```
sklearn Coefficients [[ 6.46313285e-09 -5.84257244e-06]]
skelarn Intercept [-1.79925285e-09]
my implemntation Coefficients [[-7.280123875e-06]]
my implementation Intercept -2.142344e-09
```

3.2 Naive Bayes$^\beta$

3.2.1 Overview of Naive Bayes

Naive Bayes classifier is a type of "probabilistic classifier" which is based on Bayes' theorem. This supervised machine learning algorithm assumes strong independence between features hence the name naive. The number of parameters required by Nave Bayes classifiers is linear in the number of variables (features/predictors) in a learning problem. In contrast to many other forms of classifiers, maximum-likelihood training can be done simply evaluating a closed-form expression, which requires linear time, rather than by expensive iterative approximation. Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that may be relevant to the event.


Likelihood
Class Prior Probability
$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$
Posterior Probability
Predictor Prior Probability

$$P(c \mid \mathbf{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

This is the equation of Bayes theorem. Let's breakdown this equation,

- P(c|X) is the posterior Probability of a class(target) given predictor(attribute).

- P(c) is the prior probability of class.
- P(X|c) is the likelihood which is the probability of predictor given class.
- P(X) is the prior probability of predictor.

To calculate posterior probability, first we calculate frequency for each attribute against the target. Then, transform the frequency to likelihood values and finally use the Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

A Gaussian distribution is used to estimate the parameters for the likelihood probabilities in Naive Bayes classifiers. The distribution is characterized by two parameters, its mean and variance.

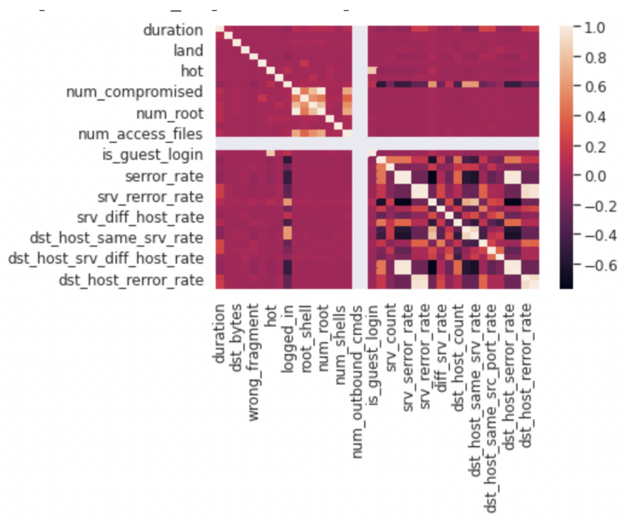
The Gaussian Naïve Bayes formula is given by

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

σ *is the standard deviation and* μ *is the mean*

3.2.2 Data Preprocessing for Naïve Bayes Classifier

Statistical based feature selection is done on the numeric data to find the features which are constant this in turn results in improving the overall efficiency and performance of the algorithm.



From the heatmap, we can notice that correlation for num_outbounds_cmds with other features is constant so we can drop it. Next, we find the features that are highly correlated and drop one feature from the highly correlated feature.

We perform Label Encoding to convert the label data into numeric data. Chi square test is performed on the encoded data to check whether categorical features depend on the output value or not. The significant value is assumed to be 0.05 and the test is performed on all the categorical features and the result was found that all the categorical features were not independent of the output value.

3.2 Decision Trees$^{\gamma}$

Random forest classifiers were developed by LEO Breiman and Adele Cutler. They combine tree classifiers to predict new unlabeled data, the predictor depends on the number of as that are represented by the number of trees in the forest, the attributes are selected randomly, each number of trees represents a single forest and each forest represents a predation class for new unlabeled data [5].

In this algorithm, I have put our training data into the decision tree algorithm to create a decision tree. And that tree we then use to classify a new, unknown example from the test data set. And in this case, the tree predicts the label of that example to be a 1. Then as the definition Random Forest I used different decision trees to classify a new unique set of examples. Then, for the final prediction of the random forest I'll be using the majority vote of all the trees. This way we introduced randomness via the examples (i.e. rows) in the data set. This approach is called bootstrapping. Moreover, to further randomize the training data via the features (i.e. columns) in the data set where only a random subset of features will be used to build the tree. This approach is called the random subspace method. So, we use this method on top of a bootstrapped data set to obtain different-looking trees. However, to increase the efficiency of the whole algorithm I have applied a random subspace method to the decision tree algorithm itself. Finally, while determining all the potential splits I have used a random subset of the features in each iteration (i.e. recursive call) of the decision tree algorithm. Consequently, in this way we will obtain unique trees with a high degree of randomness in the algorithm [6].

4. EVALUATION$^{\Omega}$

The metrics we use for evaluation are:

- Accuracy: It is the ratio of correctly predicted observations to the total observations. Accuracy = (TP + TN)/(TP + TN + FP + FN)

- F1 Score: It is the harmonic mean between precision and recall. F1 = 2 * (Recall * Precision)/(Recall + Precision)

Classifier	Accuracy	F1 score
Naïve Bayes	88.92%	89.33%
Random Forest	95.89%	96.19%

Logistic Regression	89.7%	89.7%

5. CONCLUSION$^{\Omega}$

We observe that Random Forest has the best performance among the 3 classifiers used. This is primarily due to non-linear nature of data as observed in the 2D PCA plot. Since logistic regression and naive bayes have a linear decision boundary their performance takes a hit compared to random forest which has a non-linear decision boundary. From this analysis, it shows that dimensionality reduction followed by non-linear classification can yield generalizable results for network security problems.

6. AUTHOR CONTRIBUTION

The symbol listed here indicates the author's contribution to this paper.

Ω - Vaibhavi Mutya
β - Akhilesh
ɣ - Aira

7. REFERENCES$^{\Omega}$

[1] Elejla, Omar E.; Anbar, Mohammed; Belaton, Bahari; Hamouda, Shady (2018). _Labeled flow-based dataset of ICMPv6-based DDoS attacks. Neural Computing and Applications, (), –._ doi:10.1007/s00521-017-3319-7

[2] M. Tayyab, B. Belaton and M. Anbar, "ICMPv6-Based DoS and DDoS Attacks Detection Using Machine Learning Techniques, Open Challenges, and Blockchain Applicability: A Review," in _IEEE Access_, vol. 8, pp. 170529-170547, 2020, doi: 10.1109/ACCESS.2020.3022963.

[3] Zaki, Mohammed J., Wagner Meira Jr, and Wagner Meira. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[4] Murphy, Kevin P. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[5] S. Apale, R. Kamble, M. Ghodekar, H. Nemade, and R. Waghmode, “Defense mechanism for ddos attack through machine learning,”

[6] Marina Skurichina and Robert P. W. Duin “An Ensemble Random Forest Algorithm for Insurance Big Data Analysis”