A comparative study using supervised learning for anomaly detection in network traffic

R Garg and S Mukherjee
Department of Electronics and Communications, Manipal Institute of Technology, Manipal, Karnataka, India

Email: riddhigarg26@outlook.com

Abstract. A user connects to hundreds of remote networks daily, some of which can be corrupted by malicious sources. To overcome this problem, a variety of Network Intrusion Detection systems are built, which aim to detect harmful networks before they establish a connection with the user’s local system. This paper focuses on proposing a model for Anomaly based Network Intrusion Detection systems (NIDS), by performing comparisons of various Supervised Learning Algorithms on metric of their accuracy. Two datasets were used and analysed, each having different properties in terms of the volume of data they contain and their use cases. Feature engineering was done to retrieve the most optimum features of both the datasets and only the top 25% best features were used to build the models – a smaller subset of features not only aids in decreasing the capital required to collect the data but also gets rid of redundant and noisy information. Two different splicing methods were used to train the data and each method showed different trends on the ML models.

1. Introduction
Network traffic, also referred to as Data Traffic is the amount of data moving across a computer network at a specified time. Monitoring of traffic and further analysis is extremely important in many applications, majority of them security related. With the evolution in technology and our life becoming increasingly online, the internet has become a major part of our routine. From e-commerce websites to information systems, hundreds of remote connections are made daily. With the increase in number of connections being made, comes the risk of spam and viruses attacking our systems. Historically, to overcome the problem of virus injection in their systems, users have made use of antivirus softwares and firewalls. However, these softwares follow the mechanism of ‘cure’ rather than ‘prevention.’ Intrusion Detection Systems on the other hand, examine the traffic and detect intruders before they get a chance to make any connection with a user’s system. Problems in NIDs can arise if the system is too lax or too severe. If it is too lax, there can be security breaches and if it is too severe, there is a risk of false alarms. Hence the system must be built carefully, after proper examination and analysis of the data and ML models.

NIDs (Network Intrusion Detection Systems) can be Signature based or Anomaly based. Signature based methods examine various patterns to detect the malicious connections, for
instance, sequence of bytes (0’s and 1’s) in the network traffic. Whereas anomaly-based systems make use of models built on ML Algorithms. This paper focuses on Anomaly based approach to produce an accurate model for Intrusive network predictions. The paper conducts the study over two different network traffic datasets which differ in volume of data, precisely with respect to number of features and amount of network traffic recorded. To investigate different routes of efficient detection, for each of the datasets the predicting models have been trained in two diverse ways. The results of the four separate set of values are then compared.

1.1 Datasets
The two datasets used in this study have been obtained from two different sources. They are quite distinguishable in terms of their size, shape, and features.

1.1.1. NSL-KDD. It is an improved version of KDD’99, the revisions done at University of New Brunswick. While it still suffers from a few problems that are present in its predecessor, it does not have irrelevant and redundant records which is a significant drawback of KDD’99 [7]. However, seeing the lack of public datasets available for network traffic and intrusion detection, we believe NSL-KDD dataset has a significantly large amount of clean data, which is helpful in solving the given problem. The dataset contains 42 features and roughly 47000 observations.

1.1.2 CSE-CIC-IDS2018. This dataset owes its name to Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) since it is a result of collaborative effort between the two organizations. A notion of profiles is used to generate multiple datasets for multiple days. Each dataset contains intrusion details and abstract distribution models. Profiles are useful to create various events on the simulated network. Since the profiles generated are abstract in nature, they can be applied to a diverse range of computer networking protocols having different topologies. The dataset contains 80 features and roughly 650000 observations.

To gain clarity on the proportion of benign to anomalous observations, some visualisation was conducted on the datasets. The results have been shown in the figure 1.

![Figure 1. Histograms showing the ratio of normal vs anomalous labels for CIC-IDS and NSL-KDD datasets respectively.](image)

1.2 Tools and Software Used
There are many languages and tools that can be used for Machine Learning and Deep learning analysis, some of them being Python, R, Java, Lisp, SWI Prolog, C++. However, for this research, Python was used
for a variety of reasons. Major ones being the diversity this language has in machine learning libraries it contains and its simple to edit syntax. Models require a lot of edits in their parameters for the result to be an optimised one. Python’s ability to ease rapid testing of complex and time taking algorithms makes it the best in the field.

2. Classification problem
Our end goal is to get a model that accurately predicts normal (0) and anomalous (1) connections. Since the class label to be predicted is categorical, this type of predictive modelling is referred to as Classification. To be precise, the task involves binary classification since there are only two class labels - 0 and 1. We deal with a variety of features which are then analysed and further used in algorithms to predict the correct result. Both the datasets have used a qualitative feature each, to classify each observation to their nature. The two datasets use different terminologies. In the NSL-KDD dataset a connection is labelled either as ‘normal’ or ‘anomaly,’ whereas in the CIC-IDS2018 dataset a normal connection is termed ‘Benign’ while the anomalous connections have specific attack labels to signify their nature. A categorical feature is created based on this information to make it feasible for a model to predict 0 or 1 accordingly.

3. Feature Engineering
In this process, the features of each dataset are analysed, and a smaller number of features are selected. This is done to get rid of redundant features, save cost for future network traffic collection and prevent the model from becoming extremely complex. This is achieved by dropping the missing data that is present in the form of NaN (Not A Number) and discarding qualitative features which do not add value to the predictive model. Based on Mutual Information score and Importance factor (for Random Forest), the top 25% features are used which contribute the most for each dataset (10 features for NSL-KDD dataset and 20 features for the CIC-IDS2018 dataset). This qualification is chosen by estimation. For example, the original dataset of NSL-KDD contains four categorical features, six binary features, 23 discrete features and 10 continuous features. Number 10 itself was chosen by hit and trial method. On average, all the ML models gave better accuracy on using 10 features when compared to models having 12, 13, 9, 8, 5 and 20 features.

Feature engineering methods used:
3.1 Mutual Information:
Simply put, mutual information between any two random variables is the measure of mutual dependence between them. It tells about the amount of information that can be gained for any one variable by observing some other random variable. It can be formulated as:

\[ I(X; Y) = H(X) - H(X|Y) \]

Here, LHS is the measure of mutual information obtained on examining X and Y; H(X) is X’s entropy and H(X|Y) is the conditional entropy of X given Y.

If the calculated Mutual Information between a random chosen variable and class label is high, it would imply that the class label mimics the given variable. For e.g., if a person X always takes the train to reach their destination, then the Mutual Information between the train and class is high. This technique prevents us from using redundant features in our analysis. For instance, if a person X always takes the train to reach their destination and eats a sandwich on the way, we can remove the sandwich from our problem and still obtain the desired result.
3.2 Feature importance:
This function is based on the logic followed by Decision trees. Each internal node is labelled with an input feature. Further arcs of these features are labelled with possible target values. Similar values, in case of dependent variables, are put in the same set post splitting. This process is referred to as Information Gain in case of classification (pertaining to this specific research). Feature Importance (mainly used in Random Forest) takes average of decreasing impurity over the trees.

3.3 Standard Scaler:
Standard scaler rescales the features based on Standard Normal Distribution. Hence, it reassigns the mean to a 0 and scales the entire data such that the standard deviation and variation of each sample tends towards 1 when the sample size tends towards infinity. It is commonly used in the K-Nearest Neighbours algorithm. Additionally, further scaling is done for features in the first dataset which have values with high variance. These features have been scaled by using either the maximum-score or the Z-score formula. The scaling is done to achieve normalisation of the data; keep the values from varying a lot to obtain more accurate dependencies of the target variable.

Finally, the most significant features in terms of their contribution are sorted and the top ten features from each dataset have been discussed below.

| Feature name | Mutual Information Value | Description |
|--------------|--------------------------|-------------|
| Src_bytes    | 0.5646                   | Number of bytes transferred source to destination. |
| Dst_bytes    | 0.4387                   | Number of bytes transferred from destination to source. |
| Diff_srv_rate| 0.3567                   | Percentage of connections that were made to different services. |
| Same_srv_rate| 0.3534                   | Percentage of connections that were made to same services. |
| Feature name                  | Mutual Information Value | Description                                                                 |
|------------------------------|--------------------------|-----------------------------------------------------------------------------|
| Dst_host_srv_count           | 0.3271                   | Total connections having the same port number.                               |
| Dst_host_same_srv_rate       | 0.3025                   | Connections with same host IP address and were made to same services.        |
| Dst_host_diff_srv_rate       | 0.2870                   | Connections with same host IP address but were made to different services.   |
| Logged_in                    | 0.2854                   | Is the login status 1 (successful) or 0 (not logged in).                     |
| Dst_host_serror_rate         | 0.2801                   | Percentage of connections that activated flags s0, s1, s2, s3 and have the same host IP. |
| Dst_host_srv_serror_rate     | 0.2746                   | Percentage of connections that activated flags s0, s1, s2, s3 and have the same port number. |

**Table 2. Features selected in CIC-IDS Dataset**

| Feature name                  | Mutual Information Value | Description                                                                 |
|------------------------------|--------------------------|-----------------------------------------------------------------------------|
| Init Fwd Win Byts            | 0.6694                   | Total number of bytes sent in the initial window forward.                    |
| Fwd Seg Size Min             | 0.6459                   | Min segments size observed in the forward direction.                         |
| Dst Port                     | 0.5271                   | Destination port, where the packet is sent to                               |
| Fwd Pkts/s                   | 0.5155                   | Number of forward packets per second.                                        |
| Flow IAT Mean                | 0.5131                   | Mean time between two packets sent in the flow.                              |
| Flow Duration                | 0.5122                   | Duration of the flow in microseconds.                                        |
| Flow Pkts/s                  | 0.5118                   | Number of flow packets per second.                                           |
| Init Bwd Win Byts            | 0.4947                   | Total number of forward bytes sent in the initial window.                    |
| Bwd Pkts/s                   | 0.4859                   | Number of backward packets per second.                                      |
| Flow IAT Max                 | 0.4833                   | Maximum time between two packets sent in the flow.                           |
We can thus observe that the most contributing features are ones which are transmission parameters as opposed to status parameters. Flow Packets and Error Rates determine the performance of a certain connection. This observation directly shows that the network is most vulnerable during the transmission rather than at either end i.e., source or destination.

4. Procedure

4.1 Training-Testing Split

As previously referenced, the research includes two different methods of training the data applied on each dataset. The result is accordingly reflected with respect to each. The data has been divided into training and testing sets in the proportion of 7:3 in all the instances.

4.1.1 Conventional Method. This method has been adopted in a widespread fashion by machine and deep learning communities. It involves dividing the dataset into two unequal subsets, larger one being the ‘train’ subset and smaller one being the ‘test’ subset. Each proposed model is fitted on the train subset and aims to learn from it. Test subset is used as a sort of ‘test’ for the model, i.e., only input values are given and the model is successful if the outcome is the correct target variable. In following the conventional way, the data train and test sets each contain a proportionate amount of normal and anomalous cases.

4.1.2 Quasi-Close World Assumption Method. In this approach, the model has been taught a few instances of the negative case while largely relying on the positive cases for prediction. We assumed that our system is a non-monotonous reasoning system. These systems are built on the assumption that the knowledge we currently have is insufficient and we strive to gather answers for unknowns using the existing knowledge. Anything outside of the training data is considered 1 or anomalous. The training set thus contains strictly positive cases or 0 in this instance. In our approach, major part of our training data is formed by values where the target variable is 0 and a minor part is obtained from 3% of data where the target variable is 1. Thus, devising a quasi-closed world method. The approach has been studied to draw a novel comparison between different natures of training data to build robust models.

4.2 Supervised Learning Algorithms

The research includes the implementation all the major algorithms that have given credible results in the past for binary classification problems. We compare their performance when dealing with the second form of training sets. Supervised Learning is a subset of Machine Learning and Artificial Intelligence. It makes use of a labelled dataset to train an algorithm and predict (or classify) the outcome. As the name itself suggests, it learns by ‘supervision,’ i.e., it knows the correct outcomes and makes use of them to iteratively give an improvised outcome during the prediction phase. The problem can simply be thought of as \( Y = f(X) \) where X is the input, Y is the output and f is the mapping function used by an algorithm to predict Y.

4.2.1 Decision Trees. This algorithm has a diagrammatic structure similar to a tree. Each branch represents a possible occurrence or decision, and the nodes are further split on an if-then logic in case of classification problems. For instance, someone has one flower in their hands but three choices - a red rose, a white rose, and a common daisy. The following flowchart depicts how a Decision Tree algorithm would work to find out which flower the person has.
4.2.2 Random Forest. Using the analogy given in their names, random forest algorithms are much like forests, in the sense that they cannot function without multiple decision trees. There are multiple ways and multiple features to split a decision tree and its further branches on. This algorithm *ensembles* multiple trees by training them parallely and predicts an outcome by aggravating the decision taken by all these trees. This algorithm makes use of bootstrapping to ensure there is no duplication of any tree and often successfully decreases the model variance and increases the accuracy.

4.2.3 K-Nearest Neighbours. It functions on the principle of lazy learning, i.e., it does not undergo a specialized training phase. It also works on a non-parametric learning method because it does not make assumptions on the underlying data. New data points use the principle of feature similarity; they are assigned values based on their nearest neighbours. Nearest neighbours are the points that have the least distance from the new point in feature space and K is the number of data points considered in implementation. Most common distance metric used for calculation is Euclidean, although Manhattan is also used because of its simplicity.

4.2.4 Support Vector Machines (SVM). SVM’s objective is to find the most optimized hyperplane in an N-dimensional hyperplane (where N is the number of features considered) to classify the data points. There can be many planes possible to separate the two classes of data points, but SVM strives to find the one that separates the two classes most distinctly i.e., it maximizes the margin points as much as possible. Hyperplane in an $R^2$ plane (binary classification) is a line.

4.2.5 Logistic Regression. Linear Regression has been around since the early twentieth century and although the function has several advantages, its basis of threshold can predict wrong outcomes. For instance, the predicted value of a person’s tumour being benign or malignant is 0.49, Linear regression would report the tumour to be benign because the model supports a threshold of 0.5. In reality, the person’s tumour is indeed cancerous, and the wrong prediction would have cost him his life. To overcome this problem of classification, a Logit

![Figure 3. Flowchart explaining the functioning of Decision Tree algorithm.](image-url)
function is used instead of a linear function. It works on the concept of Maximum Likelihood estimation, which makes it a better fit for classification problems.

4.2.6 Xgboost (eXtreme Gradient Boosting). Gradient Boosting is an ensemble technique, much like Random Forest. However, instead of working parallelly as is the case with Random Forest, they learn sequentially. In short, they have an ability to learn from their past mistakes and improve while in the training phase itself. It also gives a higher priority to functional space to reduce a model’s cost, instead of focusing on hyperparameters.

4.2.7 Naïve Bayes. Bayes theorem calculates the probability of occurrence of an event, based on initial information of conditions it has related to the event. Naive Bayes classifier makes use of Bayes theorem to predict whether a network is anomalous or normal. They are called ‘Naive’ because they assume that the features are strongly independent of each other.

4.3 Evaluation Metrics

4.3.1 Accuracy Score. Simply put, it is the ratio of outcomes predicted correctly over the total number of outcomes. It is obtained by the simple formula:

\[
\text{Accuracy} = \frac{TP + TN}{\text{Total Samples Taken}}
\]

where TP stands for True Positive, i.e., values which were truly predicted for the positive case and TN stands for True Negative, i.e., values truly predicted for the negative case.

4.3.2 Confusion Matrix. This matrix visually summarises the predicted results. It gives a direct comparison of TP, TN, FP, and FN values. FN values are the ones which are falsely predicted negative (normal) and FP values are falsely predicted positive (anomalous).

![Confusion Matrix](image)

**Figure 4.** Visual representation of a confusion matrix

4.3.3 Classification Report. It is a summary of precision (how many of the outcomes were predicted correctly), recall (True positives out of a total of True Positives and False positives), F1-score (the percentage of positive outcomes which turned out to be correct) and support (occurrences of a particular class in our dataset). The formulae for the four metrics are as follows:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]
where FP stands for False Positive i.e., values which were falsely predicted for the positive case.

\[
\text{Recall} = \frac{TP}{(TP + FN)}
\]

Where FN stands for False Negative i.e., values which were falsely predicted for the negative case.

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

4.3.4 ROC (Receiver Operating Characteristic) Curve. The receiver operating characteristic curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR). It, thus, reflects an overall visualisation of the performance of a certain classification model. The area under the curve is commensurate with the accuracy of the model, or specifically reflects the aggregate measure of the performance and is a decimal between 0 and 1. The formula of TPR and FPR are as follows:

\[
\text{TPR} = \frac{TP}{(TP + FN)} \quad \text{and} \quad \text{FPR} = \frac{FP}{(FP + TN)}
\]

5. Results and Discussion

Table 3. Accuracy scores obtained for dataset 1, i.e., for NSL-KDD

| Algorithm name       | Train Test Split (in %) | Closed World Approach (in %) |
|----------------------|-------------------------|------------------------------|
| Decision Trees       | 99.378                  | 81.02                        |
| Random Forest        | 100                     | 80.66                        |
| K-NN                 | 95.647                  | 78.53                        |
| SVM                  | 93.107                  | 52.63                        |
| Logistic Regression  | 89.641                  | 77.10                        |
| Xgboost              | 98.968                  | 79.28                        |
| Naïve Bayes          | 88.343                  | 52.63                        |

Table 4. Accuracy scores obtained for dataset 2, i.e., for CIC-IDS

| Algorithm name       | Train Test Split (in %) | Closed World Approach (in %) |
|----------------------|-------------------------|------------------------------|
| Decision Trees       | 99.99                   | 59.63                        |
| Random Forest        | 99.99                   | 50.87                        |
| K-NN                 | 99.82                   | 50.87                        |
| SVM                  | 92.46                   | 50.87                        |
| Logistic Regression  | 90.67                   | 50.82                        |
| Xgboost              | 98.75                   | 50.87                        |
| Naïve Bayes          | 89.82                   | 50.87                        |
Figure 5. ROC curves for best and least performing algorithms for NSL-KDD dataset. Top left and right curves show the Naïve Bayes (worst performing) and Random Forest (best performing) result for conventional train test split. Bottom left and right curves show the Naive Bayes (worst performing) and Decision Tree (best performing) results for closed world approach.

Figure 6. ROC curves for best and least performing algorithms for CIC-IDS dataset. Top left and right curves show the Naïve Bayes (worst performing) and Random Forest (best performing) result for conventional train test split. Bottom left and right curves show the Logistic Regression (worst performing) and Decision Tree (best performing) results for closed world approach.

In both datasets, conventional train-test split performed better than quasi-closed world assumption. This observation proves that the information gathered is sufficient to conduct the analysis and build models. Non-monotonic approaches cannot be relied on completely on their own. In a practical scenario, this knowledge is of a great advantage because it ascertains that we do not need to have a huge dataset to build Intrusion Detection models, hence one can make efficient use of their time and efforts in areas where they are more required.

It can also be observed that in all the four cases, the Random Forest method produced the best results. With the knowledge that we have today about these algorithms, we cannot say with 100% certainty why some models perform better under a set of conditions when compared with their counterparts which have a history of performing equally as good in other similar classification problems. We can only speculate on the reasons:
1. Random Forest makes use of bagging; hence it significantly reduces the variance and prevents a model from overfitting by building multiple sets of the original data.
2. It works well with high dimensional data.

It can also be observed that Naive Bayes in general gave a very low accuracy score in all the four cases, reasons could be:
1. It assumes that the features are ‘independent’ of each other, but our metric of feature selection heavily relies on feature ‘dependency.’
2. If it encounters any new categorical value during prediction, it takes its probability as 0, hence it tends to show the phenomenon of ‘Zero Frequency.’

It is observed that Logistic Regression did not perform well in conventional method but gave results comparable to other features in the closed world approach, reasons could be:
1. It assumes that the dependent and independent variables work linearly, which is a big limitation because linear separation is almost never found in real world scenarios.

SVM did not perform well in the Closed World Approach of NSL-KDD dataset. Our speculation is that it failed to get a clear separation margin between the two target classes.

A big coincidence that can clearly be noticed in the Quasi-Closed World Approach for CIC-IDS dataset is that all seven algorithms performed equally badly. This is a downside of the closed world approach that has been observed many times in the recent past. This implies that the models can predict 0s (benign cases) perfectly but since they did not train on malicious cases, it is unable to predict the 1s. The closed world assumption can be implemented in a better manner in the future by using algorithms to determine a similarity factor between test set features and the model. The purpose would be to obtain completely true normal cases as many malicious attackers tend to impersonate normal behaviour to attack. While the accuracy scores of the quasi-closed world approach are relatively poor, some of the algorithms like Random Forest and Decision Tree still individually perform well. This further shows, the approach requires an additional layer of insight. Although the CIC-IDS dataset is one with more data, the Quasi-Closed World accuracy scores are roughly 50%. The confusion matrices further indicate that while they achieve complete success in predicting 0s they fail terribly in predicting 1s. On a comparison, the same approach when applied to the NSL-KDD dataset acquires more success.

6. Conclusion
We made use of multiple data sources and optimised the volume of data and number of features taken by analysing the mutual information of the features. Various supervised machine learning models, namely Random Forest, Decision Trees, Logistic Regression, SVM, Naïve Bayes, KNN and Xgboost were used to determine the target variable. Results for accuracy, recall, precision and F1 scores for each model were compared. A visualization of this comparison was achieved with the help of ROC curves. The results and the discussion conclude that the conventional method of training proves superior in terms of binary classification of anomaly detection as compared to taking a Closed-World Approach. Recursion partitioning algorithms, mainly Random Forest and Decision Tree, work relatively well for such defined data with over different datasets and their implementation should be further encouraged in future research in similar domains.

References
[1] Lane T and Brodley C E 1997  *Proc. Conf. on 20th NIST-NCSC National Information Systems Security Conference* 366-381

[2] Bhuyan M H, Bhattacharyya D K and Kalita J K 2014  *IEEE Comm. Surv. & Tutor.* 16 303-336

[3] Jyothsna V and Prasad K M 2019  *Computer and Network Security* ed J Sen

[4] Fowder T P, Baulum B N and Beeharry Y 2020  *Int. J. Inf. Tecnol.* 12(3) 805-824

[5] Iglesias F and Zseby T 2015  *Mach. Learn.* 101(1) 59-84

[6] Lai Y, Zhang J and Liu Z 2019  *Secur. Commun. Netw.* 1-11

[7] Tavallaee M, Bagheri E, Lu W, and Ghorbani A 2009  *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defense Appl.* 1-6

[8] Ravi Kumar G, Venkata Seshanna K, Rahamat Basha S and Anjan Babu G 2021  *Mobile Computing and Sustainable Informatics* 68 249-258

[9] Alshammari A and Aldribi A 2021  *J Big Data* 8(90)

[10] Wu F, Li T, Wu Z, Wu S L and Xiao C Q 2021  *Int J Wireless Inf Networks* 28 262-275