# COMPARISON OF MACHINE LEARNING BASED INTRUSION DETECTION MODELS FOR DATA BREACHES

Claire Brownell, Liu Cui
West Chester University of Pennsylvania
cb946801@wcupa.edu, lcui@wcupa.edu

**ABSTRACT**
A data breach is the intentional or inadvertent exposure of confidential information to unauthorized parties. Data breach poses serious threats to organizations, including significant reputational damage and financial losses. One of the defense strategies is Intrusion Detection System (IDS), which aims at detecting anomaly quickly by using machine learning models. In this paper, we present analysis of seven machine learning models when applied to the KDD dataset. These models include Logistic Regression, K-Nearest Neighbor, Gaussian Naïve Bayes, Linear SVC, Decision Tree, and Random Forest. The analysis is tested by using statistical measurements, such as accuracy, precision, detection rate, and false alarm rate of each model.

**KEY WORDS**
Machine Learning, Intrusion Detection, Data Breach

## 1. Introduction

As technology and the Internet of Things evolves, it is inevitable that security risks come with this growth. Incidents can include, but are not limited to, unauthorized access to systems, denial of service (DoS) attacks that designed to disrupt service availability, phishing scams that aimed at deceiving individuals into revealing sensitive information, and both accidental and intentional acts leading to the deletion, damage, or corruption of data. Each instance of a data breach introduces new challenges for cybersecurity professionals to navigate during investigations.

One of the most pressing concerns in the cybersecurity domain today revolves around network security. This issue has become increasingly prominent with the widespread adoption of mobile devices such as smartphones, tablets, and laptops. These handheld devices, while enhancing connectivity and convenience, also serve as potential gateways for cyber threats.

Effective cybersecurity measures are essential not only for protecting sensitive information but also for ensuring the integrity and availability of services in an increasingly interconnected world. The goal of cybersecurity measures is to create a secure digital environment where users can trust in the protection of their data and the resilience of the systems they depend upon daily.

One of the effective cybersecurity measures is IDS, which is a device or software application that monitors a computer network or systems, analyzes data from several key points in a computer system to detect malicious activity or policy violations [1]. There are three main types of IDS, namely signature-based IDS, anomaly-based IDS, and hybrid IDS. The signature-based IDS efficiently processes a high volume of network traffic to match predefined string, pattern (such as sequences used by malware), or rules that correspond to a known attack. The biggest challenge of signature-based IDS is that it is very difficult to detect new attacks or unseen attacks. Anomaly-based IDS tried to overcome this limitation by following behavior-oriented detection. It examines network traffic and finds dynamic patterns, then compares them with normal behavior to detect deviations in the case of any anomalies. The hybrid detection approach considers both signature-based IDS and anomaly-based techniques, which tries to achieve high efficiency and identify zero-day attacks [1].

Due to the large amount and high speed of malware infection, fast reactionary capability is a must for the IDS [5]. Data science that is good at digesting large amounts of data quickly and identifying patterns makes it possible. Therefore, we will apply seven different machine learning models on one of the most widely used cybersecurity dataset, NSL-KDD to evaluate compare their intrusion detection performance.

The rest of the paper is organized as follows. Section 2 introduces the experiment setup. Section 3 describes the dataset, which is NSL-KDD. Section 4 outlines the seven machine learning models. Section 5 shows all statistic comparisons. Section 6 concludes the paper.

## 2. Experiment

The steps followed as part of the research methodology are as follows:

- Dataset**:** NSL-KDD data set is applied (section 3)

```
x, y = make_classification(n_samples=125973,
n_features=20, n_classes=2, random_state=42)
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=42)
```

- Seven classification models are applied for IDS (section 4)

```
# Define your models in a dictionary
models = {
   "Logistic Regression":
LogisticRegression(max_iter=100, n_jobs=-1),
   "KNN": KNeighborsClassifier(n_jobs=-1),
   "Gaussian Naive Bayes": GaussianNB(),
   "Linear SVC": LinearSVC(max_iter=1000, dual=False),
   "Decision Tree":
DecisionTreeClassifier(max_depth=10),
   "Random Forest":
RandomForestClassifier(n_estimators=100,
max_depth=10, n_jobs=-1),
   "PCA + Random Forest":
make_pipeline(PCA(n_components=0.95),
RandomForestClassifier(n_estimators=100,
max_depth=10, n_jobs=-1))
}
```

- All seven classification will be compared according to statistic results (section 5)

```
# Function to calculate all metrics
def calculate_metrics(y_true, y_pred):
    tn, fp, fn, tp = confusion_matrix(y_true, y_pred).ravel()
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred,
zero_division=0)
    f1 = f1_score(y_true, y_pred)
    metrics = {
        'Accuracy': accuracy,
        'Precision': precision,
        'F1 Score': f1,
        'TP': tp,
        'FP': fp,
        'TN': tn,
        'FN': fn
    }
    return metrics
# Dictionary to store results
all_results = {}

# Train, predict, and calculate metrics for each model
for name, model in models.items():
    model.fit(X_train, y_train)  # Train model
    predictions = model.predict(X_test)  # Make predictions
    metrics = calculate_metrics(y_test, predictions)  #
Calculate metrics
    all_results[name] = metrics  # Store results
```

## 3. Data Set

The NSL-KDD dataset is a refined version of its predecessor, the KDD'99 dataset. It is widely used in the cybersecurity field for training and evaluating IDS. By analyzing the patterns in the dataset, machine learning models differentiate between normal network behavior and various types of malicious activities [2,3].

The NSL-KDD data set has 42 attributes with a total number of 125,973 instances. Among the 42 attributes, 41 of them represent the characteristic attributes of the data, and 1 attribute represents the type of the attack. 20% of the data is used as training and the rest of it is used in test data set. Here are some sample features that were provided in the dataset.

***Connection features:*** Each entry starts with network connection features such as protocol type (tcp, udp), service (ftp_data, http, private, etc.), and status (SF, REJ, S0, etc.). These initial fields are categorical and describe the basic context of the network connection.

***Numerical features****:* Following the initial categorical data are various numerical features that may include, but are not limited to, the number of bytes sent from source to destination (Src Bytes), the number of bytes sent from destination to source (Dst Bytes), and other statistics derived from the network traffic such as duration, error rates, number of connections to the same host in a specified time window, and more. These features are essential for identifying patterns indicative of normal or malicious activity.

***Labels:*** Each record concludes with a label indicating whether the connection is normal or corresponds to a specific type of attack (e.g., neptune, warezclient). This label is critical for supervised learning tasks, where the goal is to train a model to accurately classify unseen connections based on learned patterns.

***Difficulty level****:* The very last number in each record (e.g., 20, 15, 21) represents the difficulty level assigned to each record, indicating how challenging it might be for a machine learning model to correctly classify the connection.

The dataset contains a set of network connections, each represented by various features derived from the network traffic and labeled as either normal or an attack, with specific attack types identified. [2, 3]. Four types of attacks exist in the dataset: Denial of Services (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). Table 1 shows the count of how many attack instances there were for each type. In Table 2-5, the samples get separated again into which specific attack it was identified as.

| Type of Attacks | Number of Samples |
|---|---|
| DoS | 45927 |
| Probe | 10163 |
| U2R | 1545 |
| R2L | 942 |
| Normal | 67343 |

*Table 1:* Attack instances in four categories

| Type of Attacks | Number of Samples |
|---|---|
| neptune attack | 41214 |
| smurf attack | 2646 |
| back attack | 956 |
| teardrop | 892 |
| pod attack | 201 |
| land attack | 18 |

*Table 2:* Attack instances in DoS

| Type of Attacks | Number of Samples |
|---|---|
| satan attack | 3633 |
| ipsweep attack | 3599 |
| portsweep attack | 2931 |

*Table 3:* Attack instances in Probe

| Type of Attacks | Number of Samples |
|---|---|
| nmap attack | 1493 |
| buffer_overflow attack | 30 |
| rootkit attack | 10 |
| loadmodule attack | 9 |
| perl attack | 3 |

*Table 4:* Attack instances in U2R

| Type of Attacks | Number of Samples |
|---|---|
| warezclient attack | 890 |
| warezmaster attack | 20 |
| imap attack | 11 |
| ftp_write attack | 8 |
| multihop attack | 7 |
| phf attack | 4 |
| spy attack | 2 |

*Table 5:* Attack instances in R2L

## 4. Machine Learning Models Used in IDS

There are two broad categories of machine learning models, supervised learning, and unsupervised learning. Supervised learning techniques learns from the input data known as the training data set and predict on testing data set [2]. Classification and regression methods are popular supervised learning techniques since they can be used to classify cybersecurity problems, such as denial-of-service attack (yes, no), spoofing (yes, no). In the following sub-sections, seven classification models that applied in this paper will be introduced.

### 4.1 Logistic Regression

Logistic Regression is a statistical method used for analyzing a dataset in which there are one or more independent variables that determine an outcome. It is used to predict the likelihood of an event occurring based on prior observations of a dataset. In the context of machine learning, Logistic Regression is often used for binary classification tasks, such as spam detection.

### 4.2 K-Nearest Neighbor (KNN)

The K-Nearest Neighbor model is a type of instance-based learning where the function is only approximated locally, and all computation is deferred until function evaluation. It's one of the simplest of all machine learning algorithms, and it's used for both classification and regression tasks, though it's more widely known for its application in classification.

### 4.3 Gaussian Naïve Bayes

Gaussian Naïve Bayes is a variant of Naïve Bayes that is specifically used when the features have a continuous distribution, and an assumption is made that the values associated with each class are distributed according to a Gaussian distribution (normal distribution). It's widely used for classification tasks since it works well with high-dimensional data. It's particularly useful in applications like spam filtering, sentiment analysis, and document classification.

### 4.4 Linear SVC

Linear SVC is a type of Support Vector Machine (SVM) used for classification tasks. It tries to find the best linear boundary that separates the classes in the feature space. By maximizing the margin between the closest points of the classes (support vectors), Linear SVC minimizes classification errors. It's effective in high-dimensional spaces and for cases where the number of dimensions exceeds the number of samples.

### 4.5 Decision Tree

A Decision Tree is a flowchart-like tree structure where an internal node represents a feature, the branch represents a decision rule, and each leaf node represents the outcome. It is a type of supervised learning algorithm that is mostly used in classification problems and works for both categorical and continuous input and output variables. It's intuitive and easy to visualize but can be prone to overfitting if not pruned correctly.

### 4.6 Random Forest

Random Forest is an ensemble learning method for classification (and regression) that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set, providing a more robust and accurate prediction by averaging multiple trees.

### 4.7 PCA with Random Forest

PCA with Random Forest combines the dimensionality reduction technique of PCA with the classification power of Random Forest. PCA is used to reduce the number of variables in a dataset while preserving as much information as possible. The transformed dataset, with reduced dimensions, is then used to train a Random Forest model. This approach can lead to improved model performance by reducing overfitting and decreasing training time, as Random Forest has fewer dimensions to consider. It's particularly useful when dealing with high-dimensional data.

## 5. Results

In the statistic analysis, the True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) are determined first.

- TP: Correctly predicted positive outcomes.
- FP: Incorrectly predicted positive outcomes.
- TN: Correctly predicted negative outcomes.
- FN: Incorrectly predicted negative outcomes.

Then, we use these values to calculate following criteria.

- Accuracy measures probability of the correct classification. So, it can be determined as (TP+TN)/(TP+TN+FP+FN).
- Precision measures the proportion of true positive results in all positive predictions as TP/(TP+FP).
- False alarm rate is the opposite of precision, which measures the proportion of false positive rate in all positive predications as FP/(TP+FP)
- Detection rate, or Recall measures the proportion of true positive results in all actual positives as TP/(TP+FN).

- F1 score measures the mean of the precision and detection rate as (2*TP)/(2*TP+FP+FN).

The bar graphs below present a comparative analysis of different machine learning models across various performance metrics. Figure 1 shows the TP, FP, TN, and FN for all models. which are the four fundamental categories in evaluating the performance of classification models. Each model is listed along the horizontal axis. Generally, high TP and TN bars would be indicative of better model performance, whereas high FP and FN bars would indicate areas where the model may be lacking. Among all seven models, Random Forest has the highest TP, TN, and lowest FP, and FN. Gaussian Naïve Bayes has the lowest FP, FN, and highest FP, and FN.

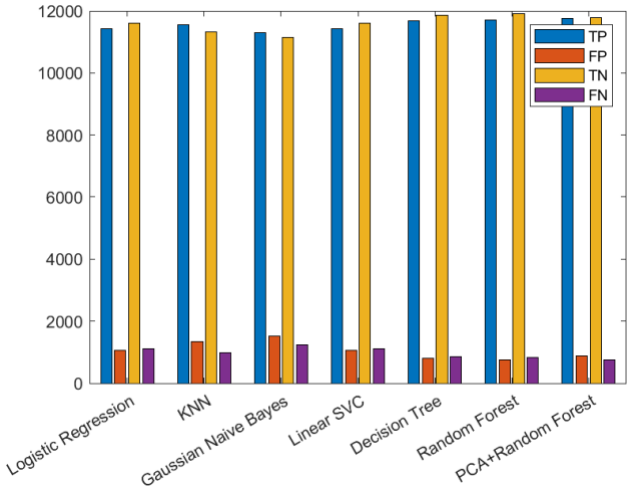


Figure 1. Comparions of TP, FP, TN, and FN

,

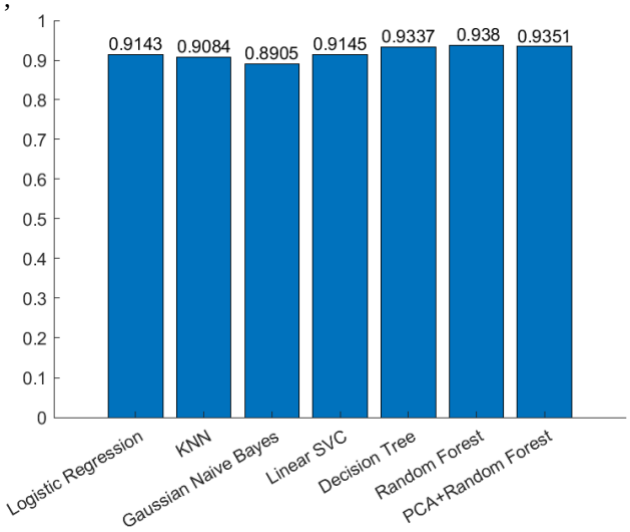


Figure 2: Accuracy

Figures 2 compares the accuracy of all seven models. Accuracy indicates the probability of correct classification among all predictions. All models demonstrating relatively high accuracy, indicating consistent model performance. Among them, Guassian Naïve Bayes has the lowest accuracy, which is 0.8905, and Random Forest has the highest accuracy, which is 0.938.

Figure 3 compares the precision of all seven models. Precision focuses on true prediction. It calculate the percentage of TP among all prediction that is true. Random Forest has the highest percentage and Guassian Naïve Bayes has the lower percentage.

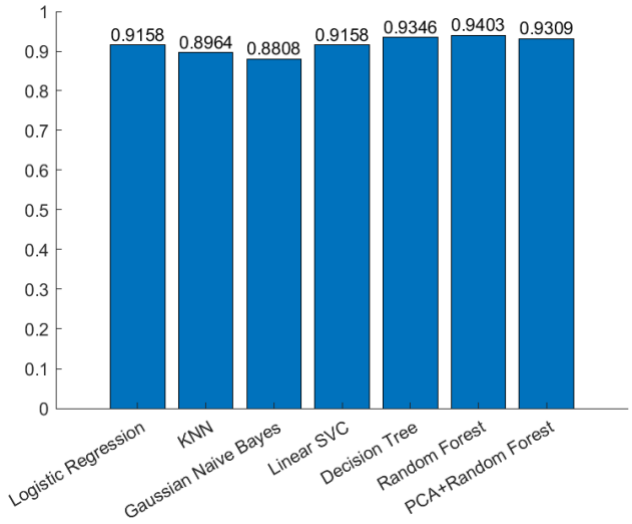


Figure 3: Precision

Figure 4 shows the false alarm rate, which is (1-precision). The false alarm rate measures the proportion of incorrect classification as positives agaist all positive prediction. For the false alarm rate, lower values are preferable as they indicate fewer incorrect positive predictions. The Random Forest model exhibits the lowest false alarm rate, which, in conjunction with a high Detection Rate, suggests a strong performance. In contrast, Guassian Naïve Bayes and KNN have higher false alarm rates compared to other models. With 0.1192 false alarm rate, it means among all true predictions 11.92% are not an attack. High false alarm rate may bring large overhead for cybersecurity measurements.

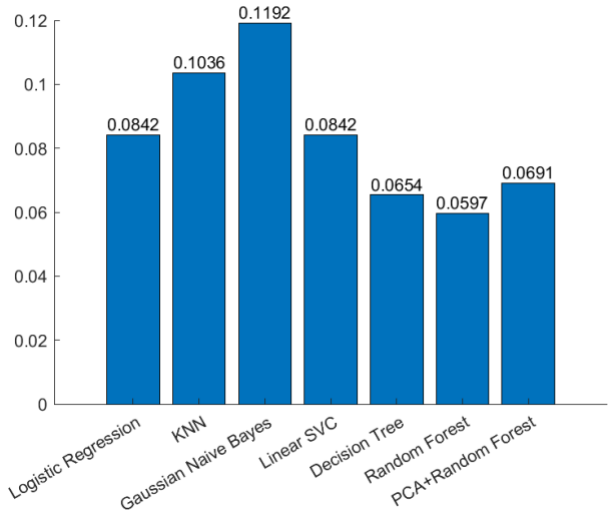


Figure 4: False Alarm Rate

Figure 5 shows the Detection Rate. The Detection Rate measures the propotion of TP against actual positive (attack). The models generally show high detection rates, with PCA with Random Forest achieving the highest detection rate, followed closely by Random Forest and Decision Tree. Gaussian Naïve Bayes appears to have the lowest detection rate among the displayed models.

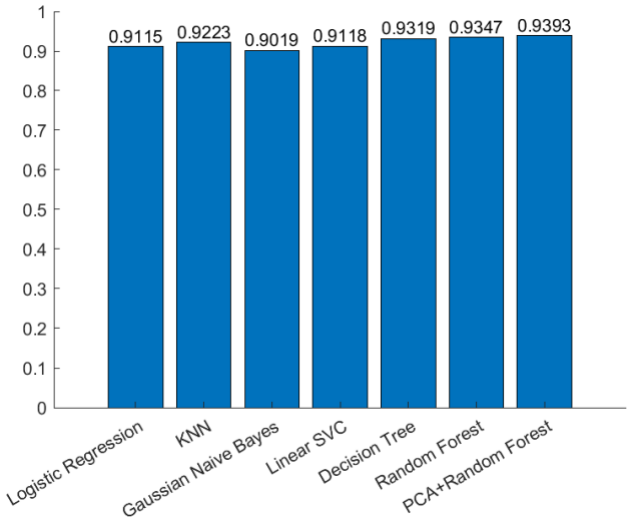


Figure 5: Detection Rate

Figure 6 illustrates F1 scores, which combine precision and detection into a single metric. These scores are also high across the board, signifying a good balance between precision and recall among the models.

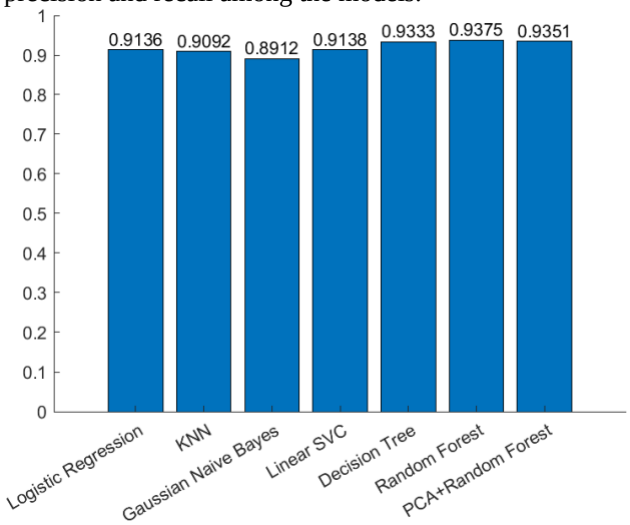


Figure 6: F1 Score

## 6. Conclusion and Future Research

To conclude, the Random Forest model emerged as the most effective in this intrusion detection scenario, balancing a high detection rate with a low false alarm rate. This suggests that ensemble methods, such as Random Forest, are particularly suitable for handling complex datasets with a mix of categorical and numerical features, like those typically found in network intrusion detection tasks. The use of PCA in conjunction with Random Forest also shows promise, potentially offering a means to improve performance further by reducing dimensionality and focusing on the most informative features.

Gaussian Naïve Bayes, KNN, and Linear SVC do not perform well since they have lower detection rate, lower precision, and higher false alarm rate.

In the future research, we will separate the dataset according to each type of attack. Analyze the seven models against each attack type and see whether we could find the best model to detect each attack.

## References:


[1] Sarker, I. H., Kayes, A. S. M., Badsha, S., Alqahtani, H., Watters, P., & Ng, A. (2020, July 1). *Cybersecurity Data Science: An overview from machine learning perspective - journal of big data*. SpringerLink. https://link.springer.com/article/10.1186/s40537-020-00318-5

[2] Aggarwal, P., Sharma, S. K. (2015, August 21). *Analysis of KDD dataset attributes - class wise for intrusion detection*. Procedia Computer Science. https://www.sciencedirect.com/science/article/pii/S1877050915020190?ref=pdf_download&fr=RR-2&rr=8597f0d8385143bc

[3] husnaa606. (2023, December 28). *NSL-KDD/datmin/kel.3*. Kaggle. https://www.kaggle.com/code/husnaa606/nsl-kdd-datmin-kel-3

[4] Mahesh1, G., Sujit, Y., Snehal, K., Jairaj, N., & Ramchandra, S. (n.d.). Data leakage detection - IRJET. https://www.irjet.net/archives/V3/i3/IRJET-V3I3227.pdf

[5] Debar, H., Hochberg, J., & Denning, D. E. (2002, May 3). Intrusion detection techniques and approaches. Computer Communications. https://www.sciencedirect.com/science/article/pii/S0140366402000373?casa_token=MhXCEmrCS6IAAAAA%3A7EJ3JA9R6JIVxVYlu9Md_PuOIwMVOthB2vRffPFwwGUbZ0DDW37G6tVCBAsvOaf57FbnArp77o