THE CATHOLIC UNIVERSITY OF AMERICA

DA516

Applications of Data Analytics and Development

Anomaly Based Intrusion Detection Project

Spring 2023

Students Name:

- NafizMahamud
- Fahad Hakami

Course Instructure:

- Dr. Chaofan Sun

Table of Contents

Introduction

Anomaly detection is a critical task in many applications, such as fraud detection, intrusion detection, and fault detection in complex systems. Traditional anomaly detection methods rely on statistical models or rule-based approaches, which may not be able to capture complex and subtle anomalies in the data. In recent years, deep learning has emerged as a promising approach for anomaly detection, as it can automatically learn complex representations of the data and detect anomalies that are difficult to detect using traditional methods.Deep learning-based anomaly detection methods typically involve training a deep neural network on a large dataset of normal data, and then using the trained model to detect anomalies in new data. The neural network can be designed to learn various types of representations of the data, such as time-series data, image data, or text data, depending on the application. The anomaly detection can be performed by comparing the output of the neural network for a new data point to the output for the corresponding normal data point. If the difference between the outputs exceeds a certain threshold, the data point is flagged as an anomaly.There are various deep learning-based anomaly detection methods, such as autoencoders, recurrent neural networks (RNNs), convolutional neural networks (CNNs), and generative adversarial networks (GANs). Each method has its strengths and weaknesses and is suited for different types of data and anomalies, but in this project, we used two different approaches. We used supervised learning with a CNN model, and semi-supervised learning using autoencoding and a couple machine learning models to compare the results.Overall, deep learning-based anomaly detection has shown promising results in various applications and has the potential to significantly improve the accuracy and efficiency of anomaly detection compared to traditional methods.

Problem Statement

Detecting intrusion in network traffic by using machine and deep learning techniques.

Understanding the dataset

Data Set Information

The NSL-KDD dataset[1] is a widely used benchmark dataset for evaluating intrusion detection systems. It was created by modifying the original KDD Cup 1999 dataset to address some of its limitations. The NSL-KDD dataset contains a more comprehensive set of attack types and traffic features, and the attack instances are better represented in the dataset.The dataset contains network traffic data for various types of attacks, including Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probe attacks, as well as data for normal network traffic. The dataset has a pre-defined train-test split, with the training set containing 125,973 instances and the testing set containing 22,544 instances. In this project we're going to use just the training dataset.

Goal

The classification goal is to predict if the connection instances an attack (it will be referred to as class 1) or it normal (it will be referred to as class 0).

Dataset Attributes

The dataset has43 variables, but for simplicity we are going to describe just the ones we used for the training[2]:

NO.	**Name**	**Typeand Values**	**Description**
1	logged_in	Numeric	Login Status :1 if successfully logged in; 0 otherwise
2	flag	Categorical:(SF, S0, REJ, RSTR, RSTO, S1, SH, S2, RSTOS0, S3, OTH)	Status of the connection
3	count	Numeric	Number of connections to the same destination host as the current connection in the past two seconds
4	serror_rate	Numeric	The percentage of connections that have activated the flag (2) s0, s1,s2 or s3,among the connections aggregated incount (3)
5	rerror_rate	Numeric	The percentage of connections that have activated the flag (2) REJ,among the connections aggregated incount (3)
6	same_srv_rate	Numeric	The percentage of connections that were to the same service, among the connections aggregated incount (3)
7	diff_srv_rate	Numeric	The percentage of connections that were to Different services, among the connections aggregated incount (3)
8	srv_diff_host_rate	Numeric	The percentage of connections that were to different destination machines among the connections aggregated in srv_count (deleted because it has a low correlation with the target column)
9	dst_host_count	Numeric	Number of connections having the same destination host IP address
10	dst_host_srv_count	Numeric	Number of connections having the same port number
11	dst_host_same_srv_rate	Numeric	The percentage of connections that were to the same service, among the connections aggregated in dst_host_count(9)
12	dst_host_diff_srv_rate	Numeric	The percentage of connections that were to different services,among the

			connections aggregated in dst_host_count(9)
13	protocol_type	Categorical:(TCP,UDP,ICMP)	Protocol used in the connection
14	service	Categorical:(http, private, domain_u, smtp, ftp_data, eco_i, other, ecr_i, telnet, finger, ftp, auth, Z39_50, uucp, courier, bgp, whois, uucp_path,iso_tsap, time, imap4, nnsp, vmnet, urp_i, domain, ctf, csnet_ns, supdup, discard, http_443, daytime, gopher, efs, systat, link, exec, hostnames, name, mtp, echo, klogin, login, ldap, netbios_dgm, sunrpc, netbios_ssn netstat, netbios_ns, ssh, kshell, nntp, pop_3, sql_net, IRC, ntp_u, rjeremote_job, pop_2, X11, printer, shell, urh_i, tim_i, red_i, pm_dump, tftp_u, http_8001, aol, harvest, http_2784)	Destination network service used
15	attack	Categorical:(Back, Land, Neptune, Pod,Smurf,Teardrop,Apache2, Udpstorm,Processtable, Worm,Satan, Ipsweep, Nmap, Portsweep, Mscan,Saint,Guess_Password, Ftp_write, Imap, Phf,Multihop, Warezmaster, Warezclient, Spy,Xlock, Xsnoop, Snmpguess,Snmpgetattack,Httptunnel, Sendmail, Named,Buffer_overflow, Loadmodule, Rootkit, Perl,Sqlattack, Xterm, Ps)	The name of the attack if any; otherwise normal

The values of the ‘flag’ column have specific meaning given below:

S0 Connection attempt seen, no reply.
S1 Connection established, not terminated.
SF Normal establishment and termination. Note that this is the
same symbol as for state S1. You can tell the two apart because
for S1 there will not be any byte counts in the summary, while
for SF there will be.
REJ Connection attempt rejected.
S2 Connection established and close attempt by originator seen
(but no reply from responder).
S3 Connection established and close attempt by responder seen
(but no reply from originator).
RSTO Connection established, originator aborted (sent a RST).
RSTR Responder sent a RST.

RSTOS0 Originator sent a SYN followed by a RST, we never saw a SYN-ACK from the responder.
RSTRH Responder sent a SYNchronize ACKnowledgement followed by a RST, we never saw a SYN from the (purported) originator.
SH Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the responder (hence the connection was "half" open).
SHR Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the originator.
OTH No SYN seen, just midstream traffic (a "partial connection" that was not later closed).

We can categorize each attack in four classes:

Attack class	Attack type
Dos	Back, Land, Neptune, Pod,Smurf,Teardrop,Apache2, Udpstorm,Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan,Saint
R2L	Guess_Password, Ftp_write, Imap, Phf,Multihop, Warezmaster, Warezclient, Spy,Xlock, Xsnoop, Snmpguess, Snmpgetattack,Httptunnel, Sendmail,Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl,Sqlattack, Xterm, Ps

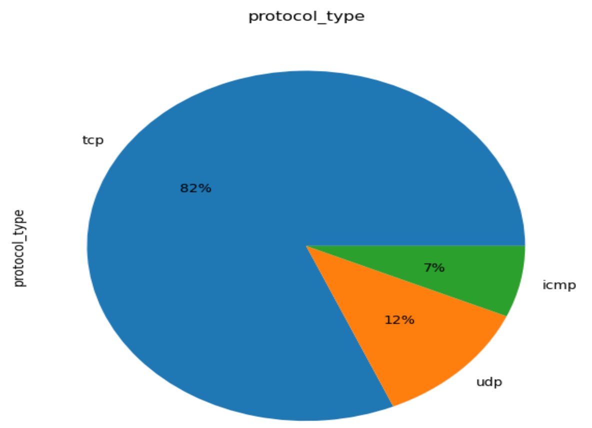
We're going to use these classes in the data analysis part, but after that the attack columns will be a binary column with either 1 for an attack or 0 for normal.

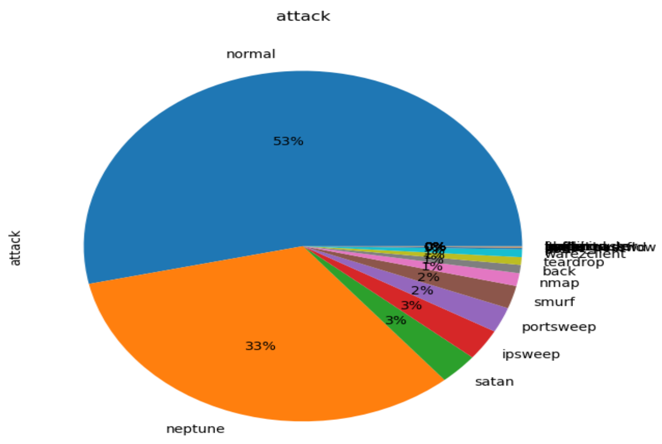
Objective

- Build a classifier Model to predict whether a request on the network is an attack or not.
- Increasing website security by stopping the request in case it was detected as an attack before any harm is done.

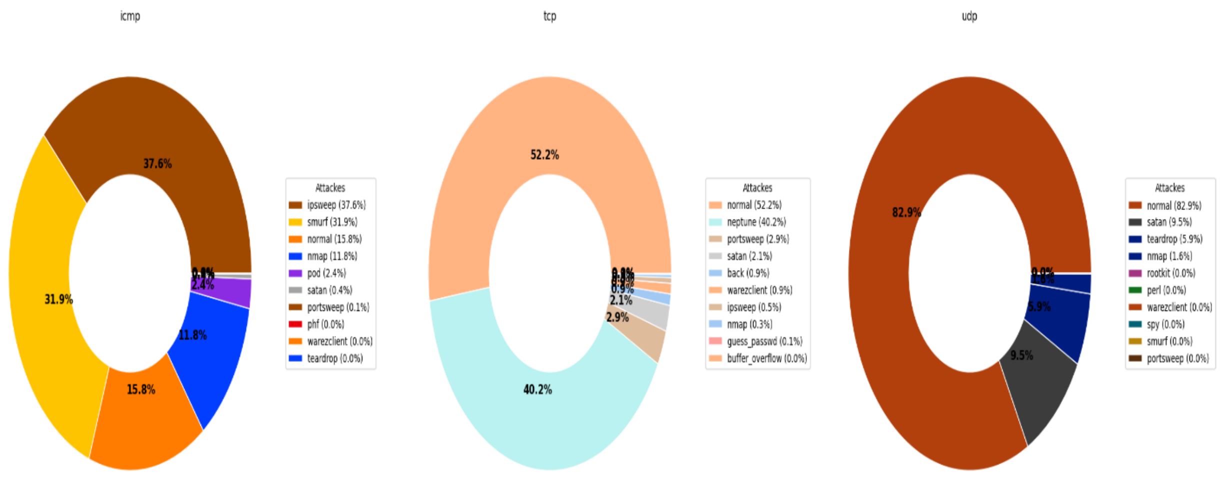
Data Analysis

The graph on the left displays that TCP protocol is used in 82% of the network traffic. The pie chart on the right reveals that out of the total traffic, 53% are classified as normal and 57% as attacks. Among the attacks, the Neptune attack type constitutes 33% of the dataset.

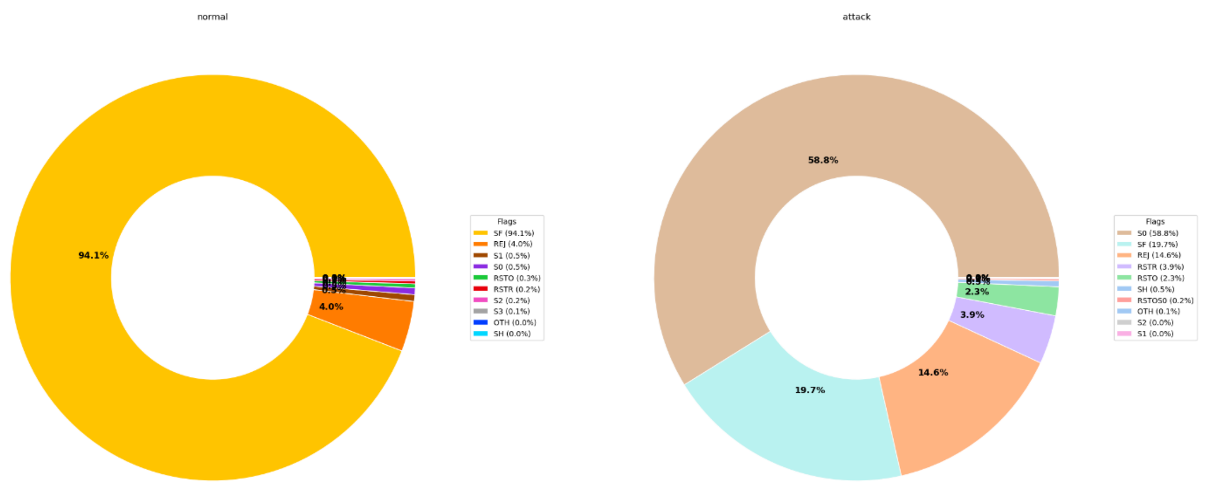




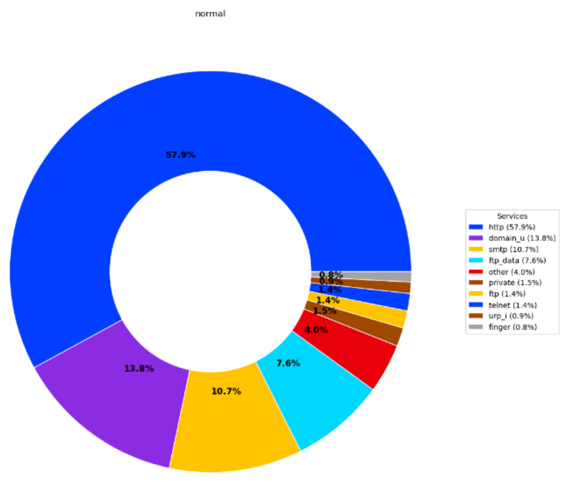
Next three charts provide insights into the characteristics of network traffic for different protocols. Specifically, it appears that traffic using the ICMP protocol is more likely to be associated with attacks, particularly IP Sweep and Smurf attacks. On the other hand, TCP protocol traffic is roughly evenly split between normal traffic and potentially malicious traffic, with Neptune attacks being the most common type of attack. Finally, traffic using UDP protocol appears to be the safest, with a majority of traffic being classified as normal, although SATAN and teardrop attacks are still more likely to occur on this protocol.

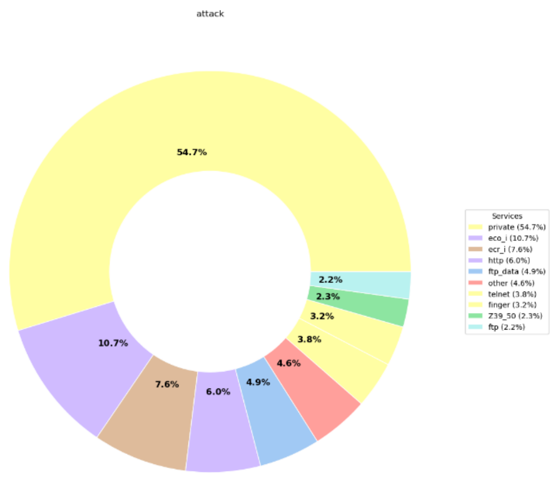


Important patterns are revealed in the next two charts related to the use of TCP flags in network traffic. Specifically, the SF flag is predominantly used in normal traffic, accounting for 94% of all such traffic. However, in the context of attacks, this flag is used in nearly 20% of all traffic. Moreover, the S0 flag is used in almost 60% of all attacks, highlighting its significance as an indicator of potentially malicious traffic.



For services, by analyzing the charts below, we can observe that HTTP service is used by 58% of the regular network traffic, but only 6% of the attacks are targeted towards it, implying it is a relatively secure choice. On the other hand, even though the private service is utilized for just 1.5% of the normal traffic, it is responsible for 55% of the attacks, indicating that it is more vulnerable to attacks.

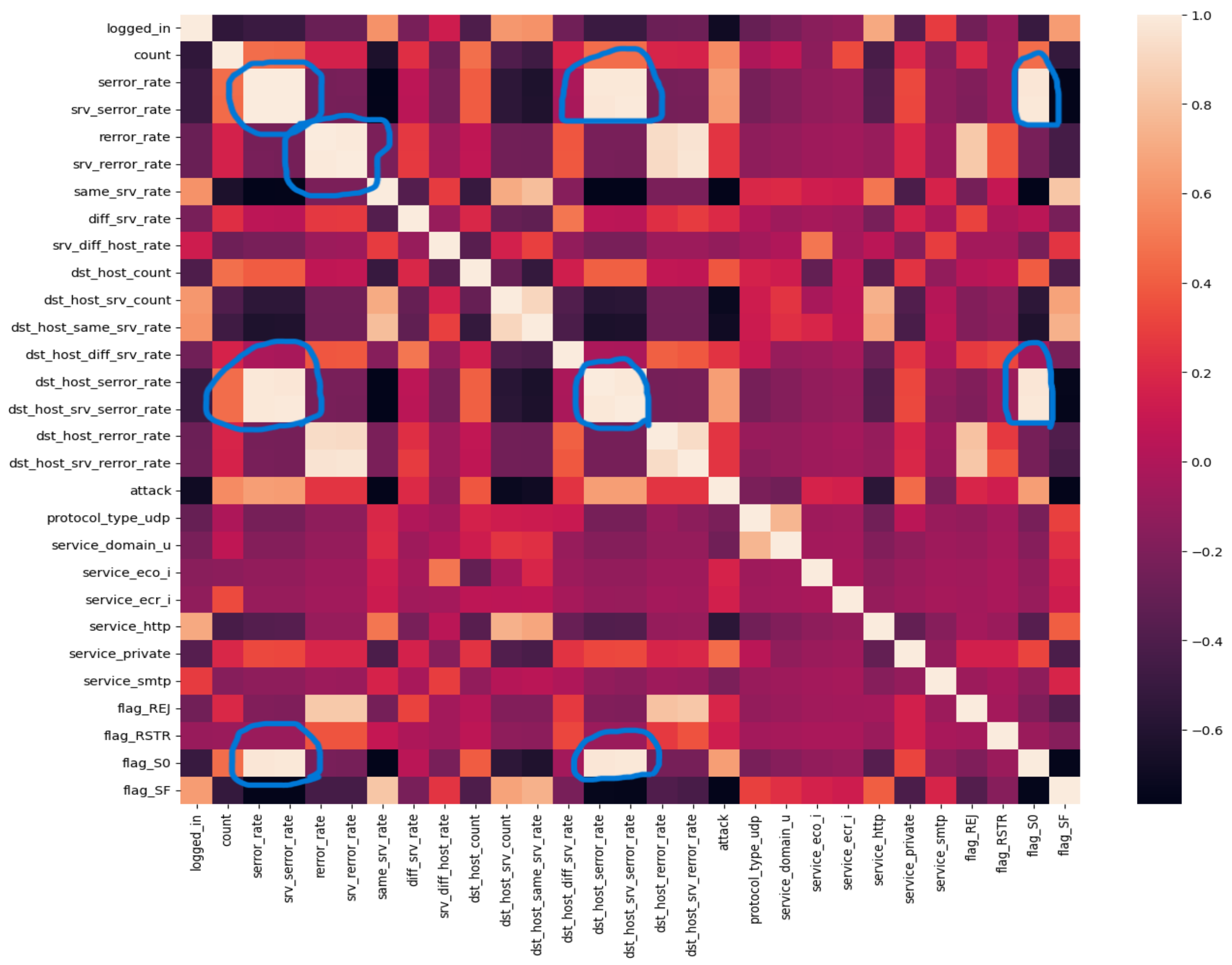




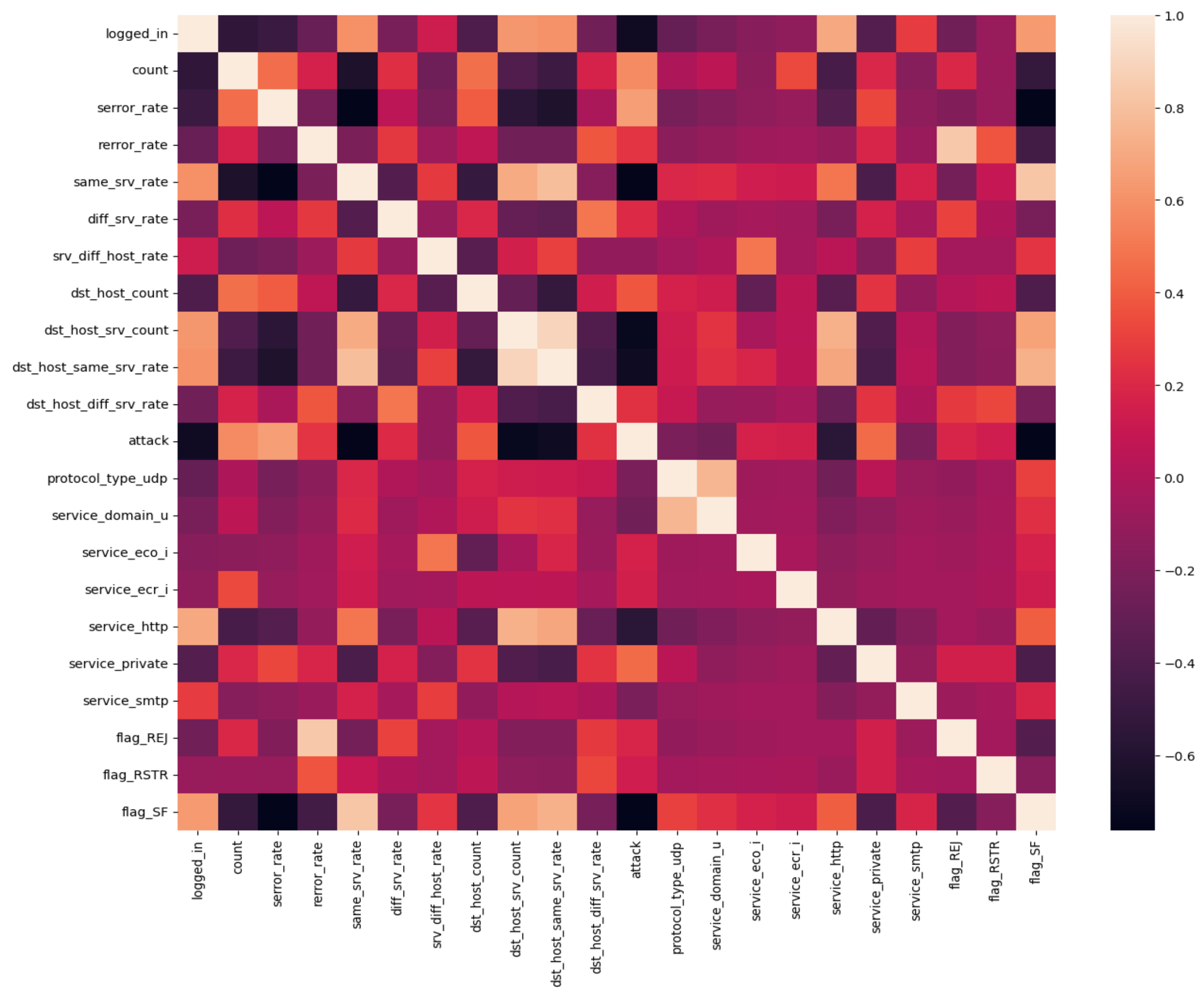
Data Preprocessing

Choosing training variables

Initially, we utilized the 1 Hot encoding technique to eliminate categorical data in order to determine the independent variables for training. Next, we evaluated the correlation between the independent variables and the dependent variable and discarded any columns with a correlation less than 1 or greater than -1. This resulted in a set of 28 independent variables. To further assess the correlation between all columns, we employed a heatmap. The heatmap revealed that certain columns had a correlation of 1 with other columns, indicating that they were identical. Consequently, we removed these duplicate columns.



Another heatmap after dropping the identical columns:



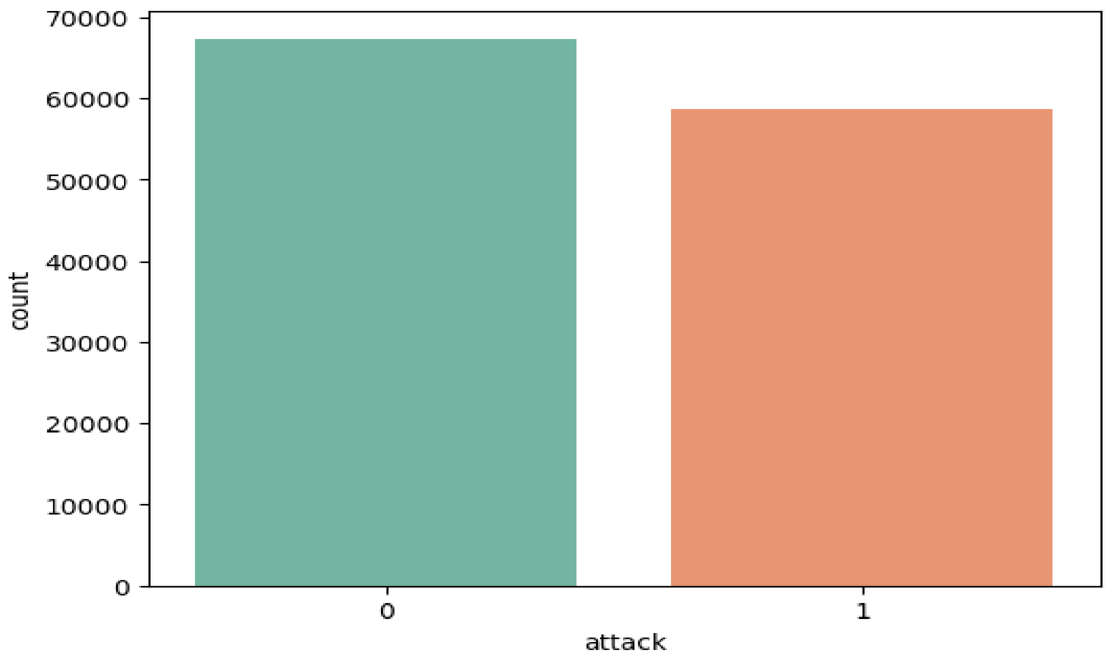
After deleting the columns, the independent columns are down to 21 columns.

Outliers

The dataset had no significant outliers.

Balancing

The dataset had more normal traffic than attacks as the graph shows below.



Therefore, we used Synthetic Minority Over-sampling Technique (SOMTE) which is a technique used in machine learning to address imbalanced datasets, where one class has fewer instances than the other. It generates synthetic samples of the minority class by creating new instances that are combinations of existing minority class instances. This helps to balance the dataset and provide a more equal representation of both classes, allowing the machine learning model to learn and generalize to new data.

Scaling

To scale the dataset, we used MinMaxScaler which scales the features to a fixed range, typically between 0 and 1. It works by subtracting the minimum value of the feature and then dividing it by the range of the feature.

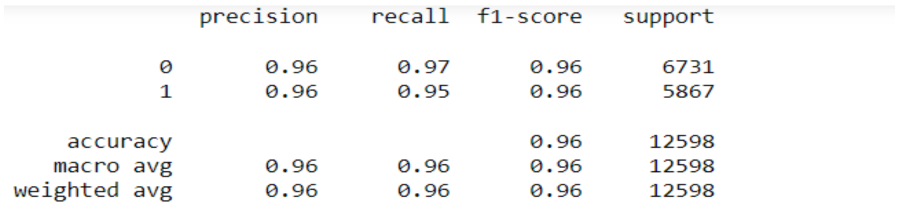
Machine Learning Models

The machine learning algorithm that was considered in this project includes.

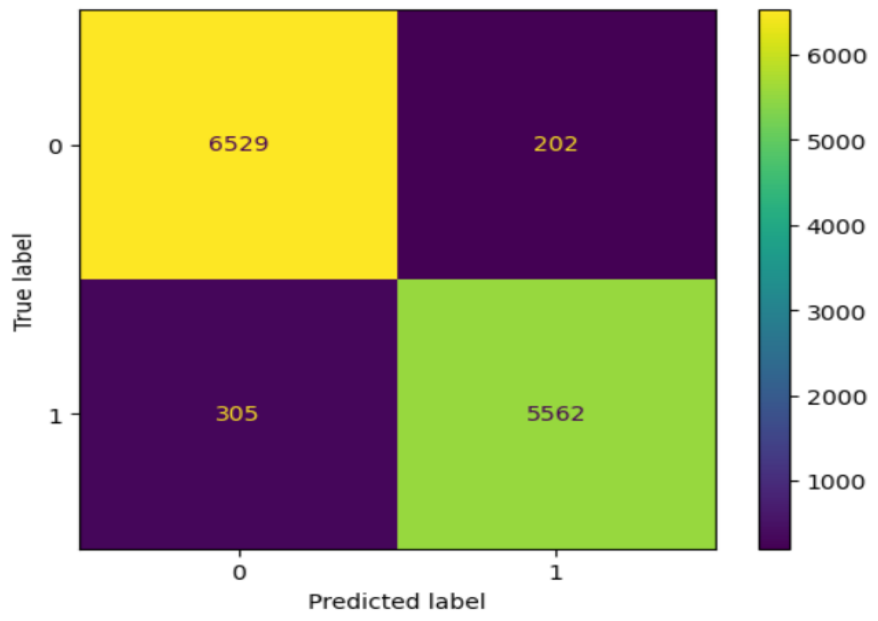
- LogisticRegression
- Random ForestClassifier (RFC)

LogisticRegression

To optimize hyperparameters in our first model, which was LogisticRegression, we usedGridSearchCV. The results of GridSearchCV indicated that the best hyperparameters for our model were C=0.1, max_iter=100, penalty=L1, and solver=saga.



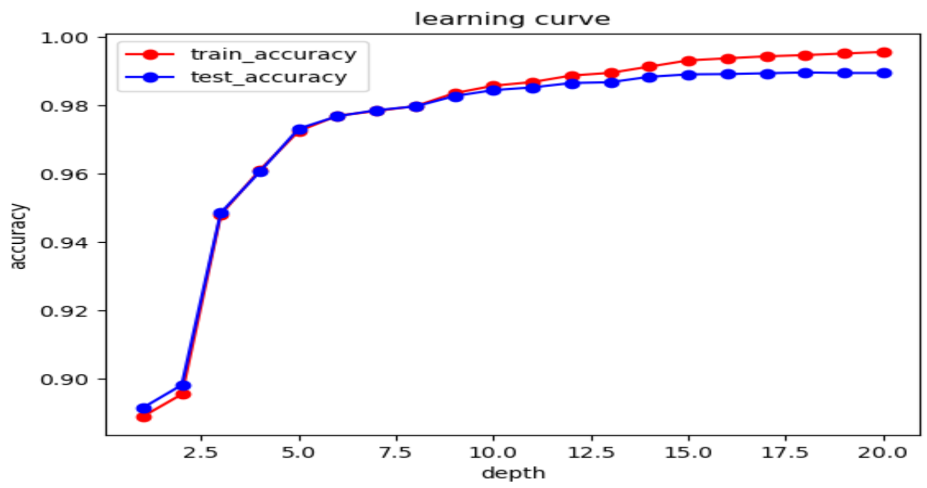
```
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      6731
           1       0.96      0.95      0.96      5867

    accuracy                           0.96     12598
   macro avg       0.96      0.96      0.96     12598
weighted avg       0.96      0.96      0.96     12598
```



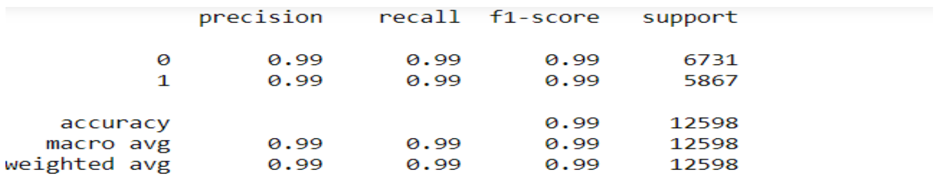
AS the confusion matrix shows above, theLogisticRegression got a 96% accuracy which is a good result, but for a secured website we need better.

Random ForestClassifier (RFC)

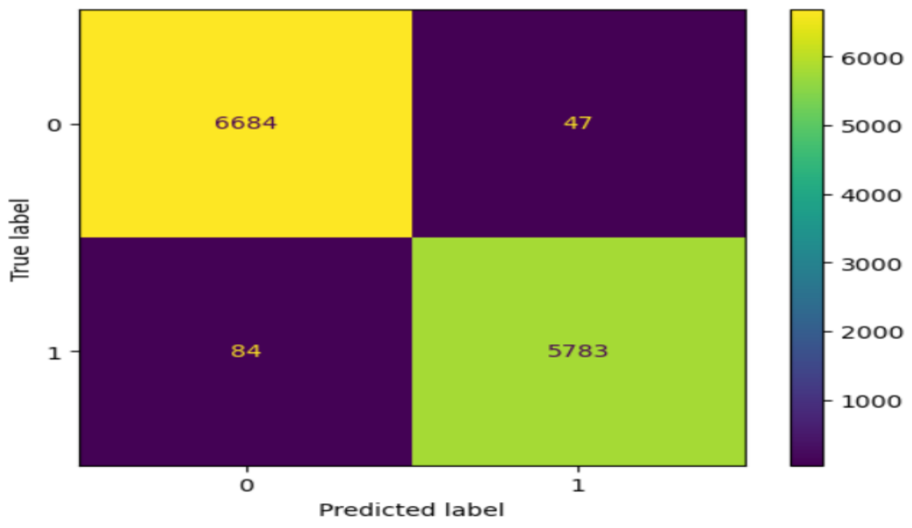
The second machine learning model we used is Random ForestClassifier (RFC) which is a very powerful model, and we used a loop to find the best depth. To avoid overfitting, we set the max depth to 20.



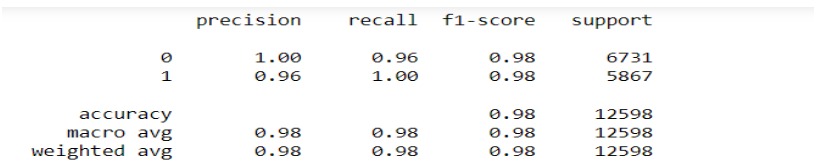
The best depth that gave us the highest testing accuracy was 18 and the results a shown on the graph below.



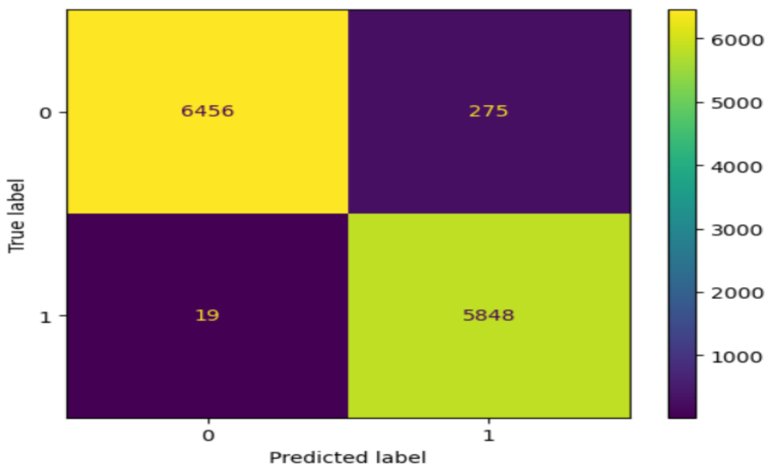
```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      6731
           1       0.99      0.99      0.99      5867

    accuracy                           0.99     12598
   macro avg       0.99      0.99      0.99     12598
weighted avg       0.99      0.99      0.99     12598
```



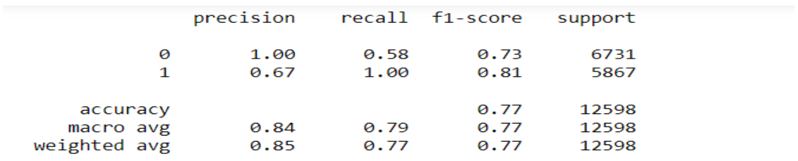
The Random ForestClassifier (RFC) model has given a very high accuracy, but the problem now is that it classified 84 attacks as normal and that can endanger any website. Therefore, we lowered the threshold to 0.1 which means if there is a higher than 10% probability for a connection instance to be an attack, classify it as an attack.



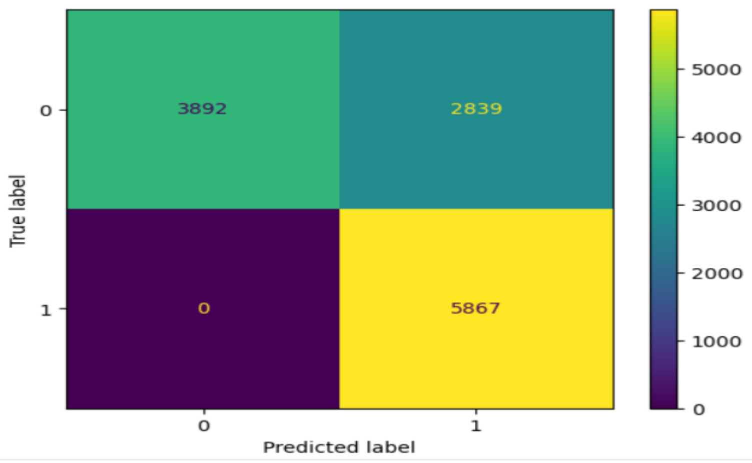
```
              precision    recall  f1-score   support

           0       1.00      0.96      0.98      6731
           1       0.96      1.00      0.98      5867

    accuracy                           0.98     12598
   macro avg       0.98      0.98      0.98     12598
weighted avg       0.98      0.98      0.98     12598
```



By lowering the threshold to 0.1 we were able to reduce the false positives(attacks classified as normal) from 84 to 19 with maintaining 96% recall for normal instances and 98%overall accuracy. That’s a much better result for security, but we are aiming at eliminating all false positiveswhile maintaining the highest recall for normal instances and overall accuracy. To achieve that we lowered the threshold to 0.0015 and by doing that we were able to eliminate all false positives, but the recall for normal instances and the overall accuracy have dropped as it’s shown on the matrix below.



```
              precision    recall  f1-score   support

           0       1.00      0.58      0.73      6731
           1       0.67      1.00      0.81      5867

    accuracy                           0.77     12598
   macro avg       0.84      0.79      0.77     12598
weighted avg       0.85      0.77      0.77     12598
```



We got 100% recall for attack with no false positives while maintaining a 77% accuracy and 58% recall for normal.

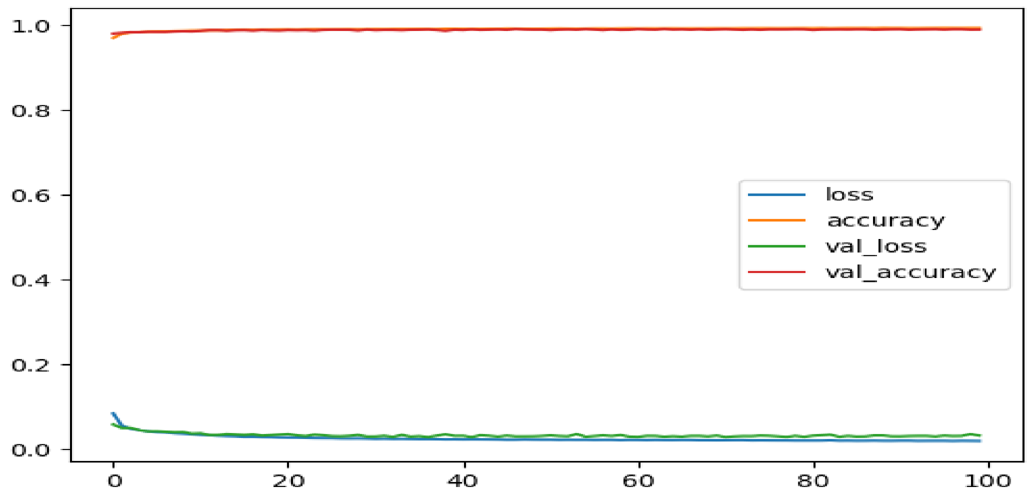
Supervised Deep Learning

CNN Model

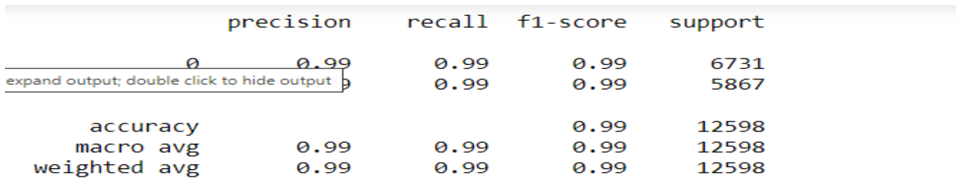
The CNN model we used for the supervised deep learning part consists of several layers of Conv1D (1D convolutional) neural network layers, followed by a Flatten layer to transform the output of the convolutional layers into a one-dimensional vector. The Flatten layer is then followed by two dense layers with Rectified Linear Unit (ReLU) activation functions, which help the network learn non-linear relationships in the data. The final dense layer uses a sigmoid activation function to produce a binary output between 0 and 1. The model is compiled using the Adam optimizer, binary cross-entropy loss function, and accuracy as the evaluation metric.

Overall, this model is designed for binary classification tasks, such as predicting whether an input sequence belongs to a certain class or not. Convolutional layers are particularly useful for analyzing sequential data, such as time series or natural language data, where the relationships between adjacent elements in the sequence are important. The ReLU activation functions help to introduce non-linearity into the model, while the sigmoid function produces a probability output for the binary classification task.

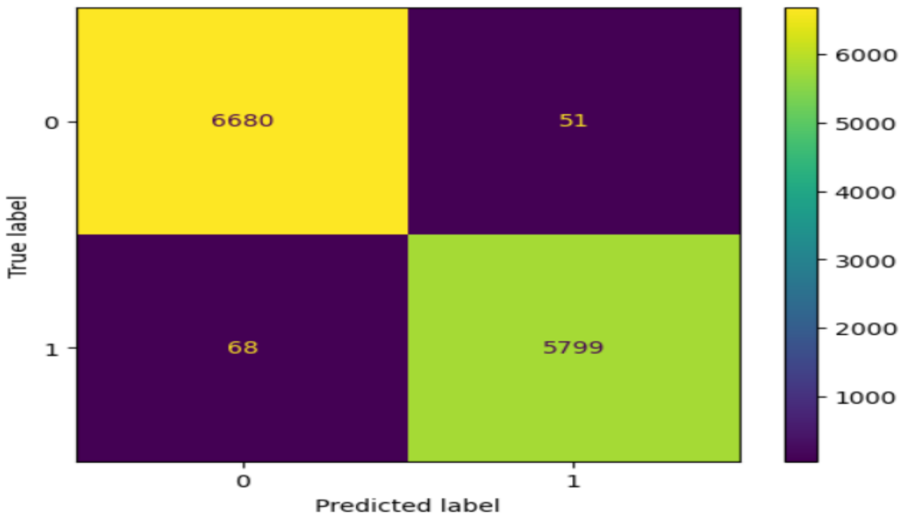
The loss and the accuracy for the CNN model:



The best training accuracy was 99.05% which is better than both machine learning models.

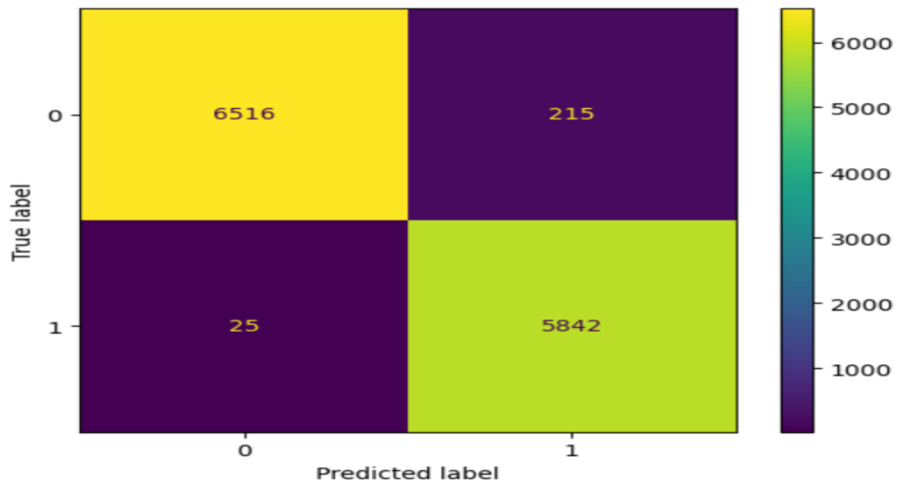


```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      6731
expand output; double click to hide output      0.99      0.99      5867

    accuracy                           0.99     12598
   macro avg       0.99      0.99      0.99     12598
weighted avg       0.99      0.99      0.99     12598
```



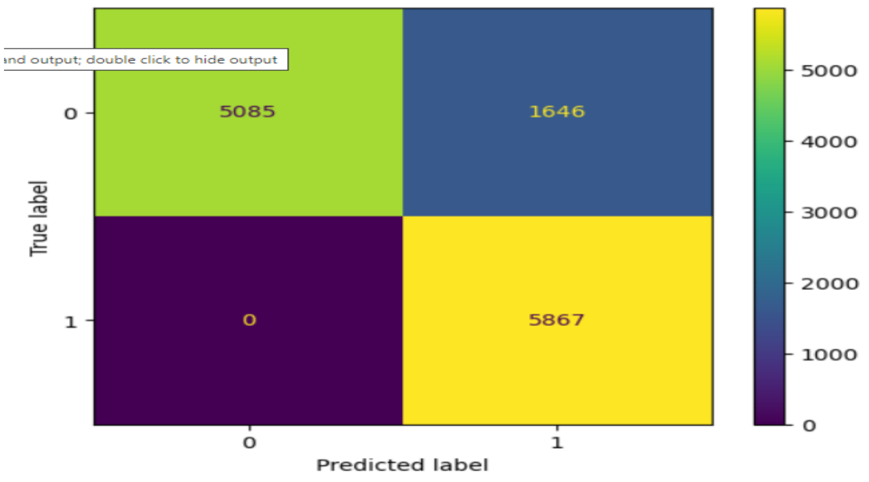
The number of false positives is lower than the one we got from the Random Forest Classifier model. Next, we tried to reduce the threshold to 0.1 reduce the number of false positives.

```
              precision    recall  f1-score   support

           0       1.00      0.97      0.98      6731
           1       0.96      1.00      0.98      5867

    accuracy                           0.98     12598
   macro avg       0.98      0.98      0.98     12598
weighted avg       0.98      0.98      0.98     12598
```



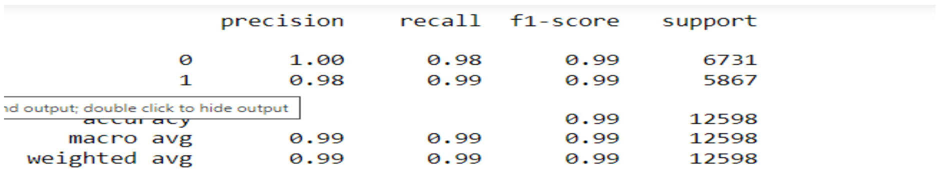
False positive cases were reduced from 68 to 25 with maintaining 97% recall for normal instances and 98% which is similar to what we got with the Random Forest Classifier model. To eliminate all false positive cases, we tried to find the best threshold as we did for the Random Forest Classifier model, and we were able to do that while maintaining 76% recall for normal instances and 87% overall accuracy with threshold equal to 0.00001.

```
              precision    recall  f1-score   support

           0       1.00      0.76      0.86      6731
           1       0.78      1.00      0.88      5867

    accuracy                           0.87     12598
   macro avg       0.89      0.88      0.87     12598
weighted avg       0.90      0.87      0.87     12598
```

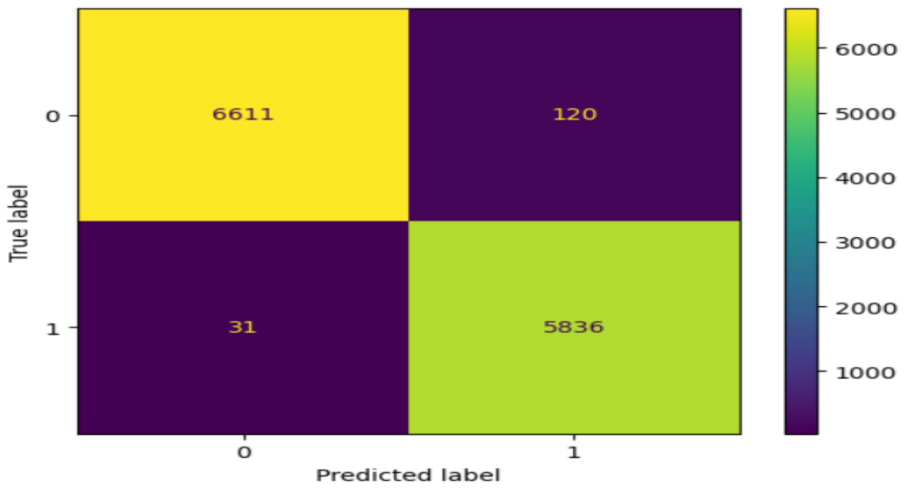


CNN Model with 1 : 7 Ratio

In order to improve the detection of attacks in the model, we adjusted the weights to put more emphasis on class 1 (attacks class) and retrained the model. The results showed an increase in the recall score for attacks, with a minor decrease in overall accuracy and recall score for normal instances, which was expected.

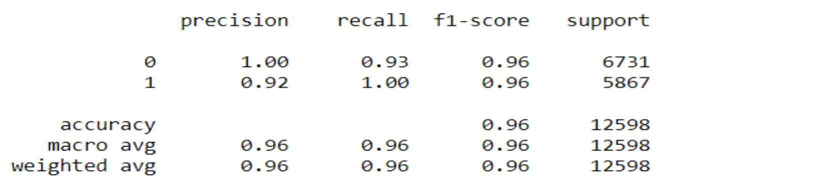


```
              precision    recall  f1-score   support

           0       1.00      0.98      0.99      6731
           1       0.98      0.99      0.99      5867

    accuracy                           0.99     12598
   macro avg       0.99      0.99      0.99     12598
weighted avg       0.99      0.99      0.99     12598
```

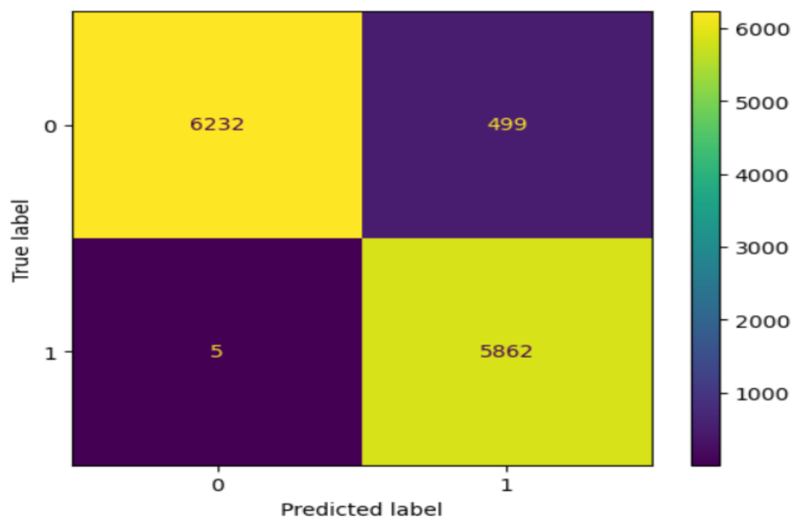


As we did previously, we tried to optimize the threshold to get no false positives while maintaining the highest recall for normal instances and overall accuracy. The graphs below show the results with threshold equal to 0.1 and 0.003.

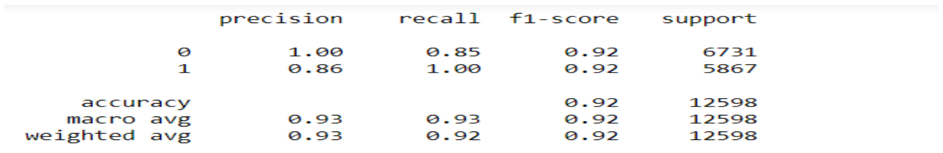
Threshold equal to 0.1:



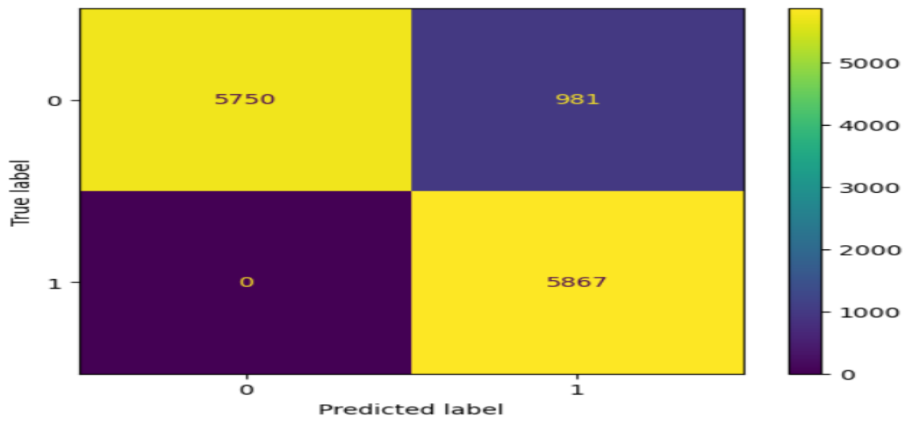
```
              precision    recall  f1-score   support

           0       1.00      0.93      0.96      6731
           1       0.92      1.00      0.96      5867

    accuracy                           0.96     12598
   macro avg       0.96      0.96      0.96     12598
weighted avg       0.96      0.96      0.96     12598
```



Threshold equal to 0.003:



```
              precision    recall  f1-score   support

           0       1.00      0.85      0.92      6731
           1       0.86      1.00      0.92      5867

    accuracy                           0.92     12598
   macro avg       0.93      0.93      0.92     12598
weighted avg       0.93      0.92      0.92     12598
```



To sum up, we succeeded in creating a deep learning model capable of detecting anomalies within a network. Although a machine learning model could potentially yield a similar accuracy score, the deep learning model's flexibility enables us to adjust the level of security the model provides more effectively. By utilizing a CNN model, wewere able to achieve an accuracy rate of up to 99% with fewer than 1% false positives, which could jeopardize website security, and fewer than 1% false negatives, which could impact user experience. This outcome is particularly beneficial for an average website that does not require a high level of security. Additionally, we accomplished the task of eliminating all false positives while still maintaining a false negatives rate of less than 15% and an accuracy rate of 92%, which is a level of performance that may be required by governmental or banking websites.Determining the optimal model and threshold can have significant implications for the system, depending on whether it is more important to have a user-friendly system with fewer normal instances being blocked due to false positives, or a more secure system where every attack is detected.

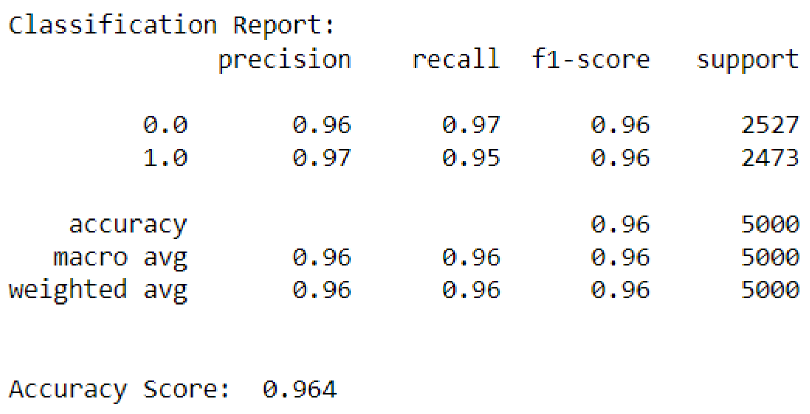
Semi-Supervised Deep Learning:

Semi-Supervised Learning is combination of supervised and unsupervised learning processes in which the unlabelled data is used for training a model as well. In this approach, the properties of unspervised learning are used to learn the best possible representation of data and the properties of supervised learning are used to learn the relationships in the representations which are then used to make predictions.

In this kernel, I have explained how to perform classification task using semi supervised learning approach. This approach makes use of autoencoders to learn the representation of the data then a simple linear classifier is trained to classify the dataset into respective classes.

Autoencoders are a special type of neural network architectures in which the output is same as the input. Autoencoders are trained in an unsupervised manner in order to learn the exteremely low level repersentations of the input data. These low level features are then deformed back to project the actual data. An autoencoder is a regression task where the network is asked to predict its input (in other words, model the identity function). These networks has a tight bottleneck of a few neurons in the middle, forcing them to create effective representations that compress the input into a low-dimensional code that can be used by the decoder to reproduce the original input.

We used only 5000 normal data to train the encoders and then obtain the latent representation which is used to predict 1000 both normal and intrusion data. Then we combine both predicted normal and intrusion (20,000) with compressed dataset and split them for train and test set. Finally, we simply apply logistic algorithm for the classification got accuracy 96%.



```
Classification Report:
               precision    recall  f1-score   support

         0.0       0.96      0.97      0.96      2527
         1.0       0.97      0.95      0.96      2473

    accuracy                           0.96      5000
   macro avg       0.96      0.96      0.96      5000
weighted avg       0.96      0.96      0.96      5000


Accuracy Score:  0.964
```

Conclusion:

Based on our experiences on this project, we have reached a conclusion that semi supervised can be a better model due to it requires very less labeled data and we can improve the performance of the model by applying different kinds of autoencoders like sprase, avriational etc. For the future research, more well suited generative autoencoders can be applied to get more insightful features.

References

[1].Dataset: NSL-KDD (https://www.kaggle.com/datasets/hassan06/nslkdd)

[2]. L.Dhanabal, Dr. S.P. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms", 2015.(https://drive.google.com/file/d/1V9-UoUGh3UZ1ZLucUq-DwRo25eb-alMN/view?usp=share_link)