REPORTS:
End-to-end project delivery on cyber-security data analytics

Student: Khoa Anh Do

ID: 223188874

Table of Content

Introduction

In this assignment, we will perform an end-to-end data analytics project on “cyber-attack classification in the network traffic database”. The goal is to understand the two given datasets and classify the correct classification of attacks utilizing different machine learning algorithms.

The report structure includes

- Description of the two datasets
- Theory of the algorithms used and performance metrics measured
- Results report and discussion
- Reference

Datasets

This section will give an overview of two datasets for cyber-attack classification: NSL-KDD and TON_IoT

NSL-KDD

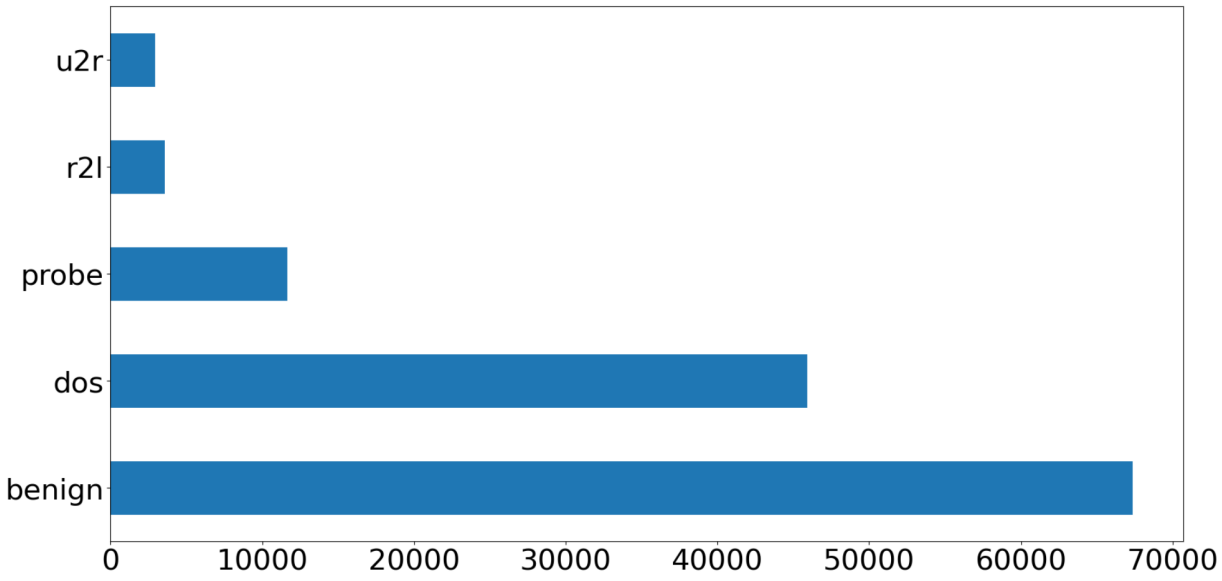
Organized by the ACM SIGKDD (Association for Computing Machinery's Special Interest Group on Knowledge Discovery and Data Mining), KDD is a competition for data mining and analytical discoveries which includes participants across the globe to promote innovation solutions. KDD’99 is the standard dataset that includes multiple network traffic recordings, and from there, the University of New Brunswick produced a refined version called NSL-KDD.

The dataset contains four different types of network attacks: DoS, R2L, Probe, and U2R. In details:

- Denial of Service (DoS): This type of attack attempts to make a system unavailable to the actual users by flooding the system traffic, causing it to shut down or crash.
- Remote to Local (R2L): This type of attack includes the intruder finding a way to enter local resources and network through remote access
- User to Root (U2R): By utilizing the vulnerability in the system implementation, the attack may gain root access functionality as a normal user

- Probe or surveillance attacks attempt to steal as much information after scanning and gaining access to the network

In addition, we also have to identify network records that do not have any suspicious activity, namely benign or normal class. In the below image, we plot the histogram for each class



As we we see, the number of instances in each class is not equally distributed. The majority class is benign or has no sign of maliciousness, followed by DoS attacks. User to Root and Remote to Local has the lowest number of instances.

TON_IoT

With the advanced progression of the Internet of Things in recent years, there are concerns regarding the cyber-security aspect of its real-world application. To address this problem and build effective Intrusion Detection Systems, a public dataset is compiled and published so data analyst experts can examine and share their solutions to the problem.

Even though the dataset contains detailed attack types in the **types**, for the scope of this assignment, we only need to detect whether an instance's attributes determine whether it is an attack or not. In details:

- Label 0: The network attributes indicate normal activity or no attack
- Label 1: The network attributes indicate there is an attack

Theories

This section will discuss the algorithms that are used for data analytics in the task as well as the performance metrics used to measure the efficacy of each method. In the next section, actual performance metrics results are entailed and discussed.

Algorithm: K-nearest neighbors

K-nearest neighbors is a non-parametric supervised machine learning algorithm. At a basic level, the algorithm assumes that similar instances (based on distance calculation) have a tendency to belong to the same class. There are many distance calculation methods, although the most common method is Euclidean distance.

More specifically, the algorithm classifies new instances by locating the K nearest instances (or K nearest neighbors). Then, the most common class observed from those neighbors will be selected as the predicted category. Here, k is the hyperparameter chosen by the engineer, to which we can perform hyperparameter tuning to select the optimal value.

Algorithm: Random Forests

Random forest belongs to a class of ensemble learning in machine learning, which aims to combine the results of multiple single machine learning algorithms and produce the best prediction. Specifically, this algorithm aims to draw outcomes from a number of random decision trees, hence the name random forest.

Decision Tree has the same concept as a binary tree in data structure. Each internal node of a tree represents a “decision”; for example, for a variable height, a decision would be *“if height is larger than 150 centimeters, proceed to the right child, or else proceed to the left child”*. By constructing a tree of “decisions”, at each leaf node, we end up with the classified data instances. A Machine Learning engineer has the option to tune many hyperparameters to achieve the optimal tree. Below are some listed options for hyperparameters in scikit-learn:

- max_depth: the max depth of the tree. If this height is reached, no more splitting is done.
- min_samples_split: the minimum number of samples required to split a node

- min_samples_leaf: the minimum number of samples to be considered a leaf node
- max_features: When considering making a decision node, how many features we have to keep in mind
- criterion: the function used to consider the best decision split

Decision tree is non-parametric, easy to understand and visualize, works well with both categorical and continuous variables, and is fairly good in terms of computation cost. However, decision trees suffer some drawbacks such as overfitting due to not capturing the general idea, and may create a bias due to an imbalanced dataset. Random Forest addresses some of these issues. In theory, a Random Forest should improve the result of a single decision tree since it trains multiple decision trees on random features, and by averaging all the outcomes, it should reduce the bias that we see in a single decision tree. Another advantage of Random Forest is that we can decide the number of trees in the dataset, which is another hyperparameter.

Algorithm: Logistic Regression

Traditionally, methods like linear regression can also be used for classification, to examine whether the predicted output passes a threshold; however, this model drawback is the interpretability: if the model predicts negative probability, it cannot be understood intuitively. The core concept of Logistic Regression utilizes the logistic function, whose unique attribute can produce values interpretable probabilities in the range of 0 and 1, effectively resolving linear regression's classification drawback:

$$p(x) = \frac{1}{1+e^{-(\beta 0 + \beta 1 x)}}$$

In scikit-learn, Logistic Regression is conveniently implemented with a built-in hyperparameter tuning mechanism, called LogisticRegressionCV. The built-in cross-validation iterates through different values of penalty strength for some regularized logistic regression implementation to prevent overfitting.

Algorithm: Support Vector Machine

The core idea of a support vector machine is to find optimal hyperplanes that can ideally separate data into different regions depending on the classification. For patterns that are not linearly separable by transformations, a kernel function is needed that in the abstract can map the data into new space. In general, support vector machines can be divided into linear SVM, where the dataset is linearly separable,

and non-linear SVM, where the dataset is not linearly separable. Below are the benefits of support vector machine:

- Effective in small, but complex datasets
- Effective in high dimensional datasets
- Variety: different kernel functions can be used for specified decision functions

However, support vector machines also suffer some drawbacks:

- Choice of Kernel: Selecting the right kernel function and its parameters can be challenging and might require domain expertise.
- Sensitivity to Noise: SVM can be sensitive to noisy data, as noisy points near the decision boundary can impact the model's performance.
- Computational Complexity: Training an SVM can be computationally intensive, especially with large datasets.

Algorithm: Multilayer Perceptrons

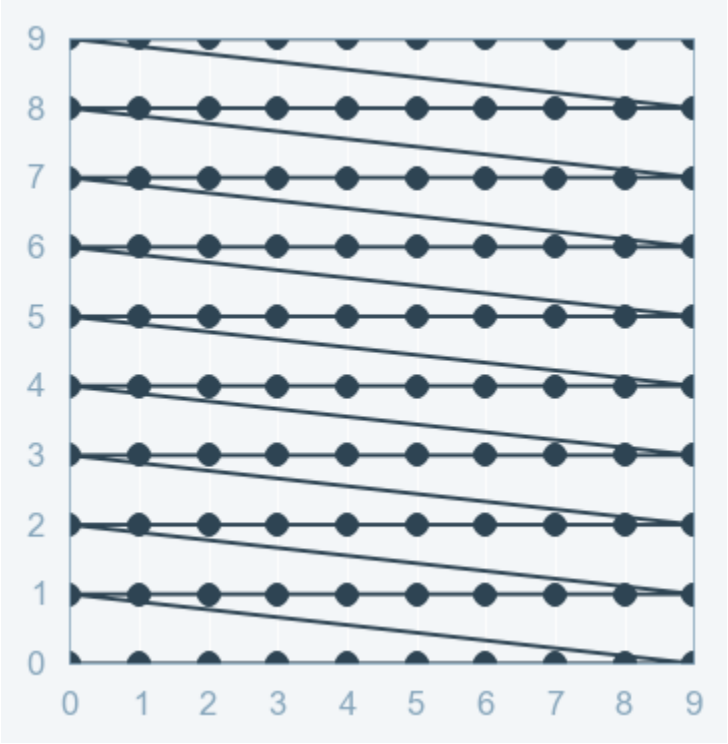
Multilayer Perceptron is a concept that precedes the current Neural Networks. In essence, this is a Machine Learning field that borrows the concept of a neuron from neuroscience (biomimicry). The neural networks may contain a hierarchical structure of neuron layers, varying a different number of neurons, and feed its output to the next layer. At each neuron, the input is weighted, and passes through an activation function onto the next set of neurons. There are many advances in Deep Learning as of 2023, such as Recurrent Neural Nets which use a stream of input to predict its outcome depending on the task, including question-answering, classification, etc.

The training of Neural networks surrounds back-propagation: in general, it involves calculating the error between predicted and actual outputs and adjusting the weights of the connections to minimize this error.

In this project, we limit ourselves to 2 layers, the first one containing 100 neurons, with RELU activation function and Adam solver to learn the pattern of the dataset.

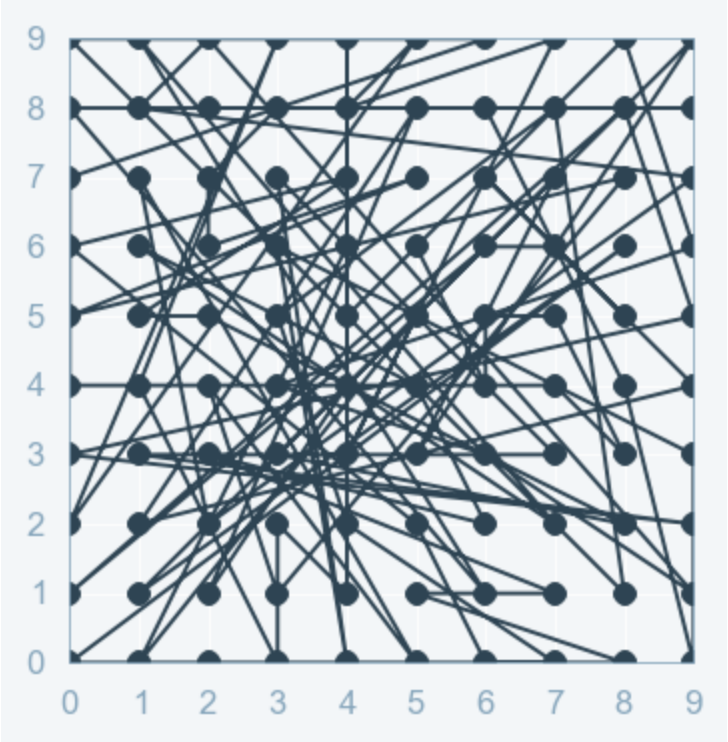
Hyperparameter tuning: GridSearchCV and RandomSearchCV

In the above algorithms, we can see that hyperparameters are important when building a Machine Learning model. Although there is no one formula for success, Machine Learning engineers can still determine the best possible outcome by trying out different values for each hyperparameter. In scikit-learn, GridSearchCV enables this process: it finds the optimal set of hyperparameters from all possible combinations that we have given the algorithm. For example, we can set a range of k values from 1 to 100, and use grid search to find the optimal k-value. The scikit-learn algorithm does this by performing an exhaustive search, in combination with cross-validation, which divides the training set into further training-validating subsets.



Visualization of Grid Search

One drawback of GridSearchCV is trying out all possible outcomes which requires tremendous computing power and possible parallel computing must be carried out. RandomSearchCV is an alternative that addresses this problem. Instead of iterating through every single combination, RandomSearchCV only selects a few sets of hyperparameters to tune the model. Cross-validation is also used in this method.



Visualization of Random Search

Metrics: Confusion matrix

For a binary classification, we can compare and categorize the outcome to one of the four type:

- True positive: The positive labeled instance is actually positive
- True negative: The negative labeled instance is actually negative
- False positive: The positive labeled instance is actually negative
- False negative: The negative labeled instance is actually positive

This can be extended to multi-label classification problem by further classify the outcome to multiple different False positive

Metrics: Accuracy

Accuracy measures the percentage of labeled instances that is actually correct. In binary classification, this can be expressed in a formula as

$$accuracy = \frac{True\ Positive + True\ Negative}{Test\ Size}$$

And in multi-label case, this can be extended as the number of test instances that are correctly matched with their actual label.

$$accuracy = \frac{Actual\ Match}{Test\ Size}$$

Accuracy suffers from some drawbacks since it does not explain the proportion of corrected samples from all positive class instances. Therefore, many more metrics are used for classification reports

Metrics: Precision, Recall, F-scores

Precision measures the proportion of labeled instances that are actually correct. In mathematics, we describe this as

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

This metric is easy to understand and interpret in binary classification. However, in multi-label cases, there are better ways to determine precision scores, namely **macro** and **weighted** calculation of precision.

- **Macro precision** is more useful in multi-labeling cases than precision. It measures the arithmetic means of precision score for all classes.

$$macro\ precision = \frac{Precision\_1 + Precision2 + Precision3 + \ldots + Precision\ N}{N}$$

- **Weighted precision** refers to the weighted mean of the precision score, where the weight refers to the percentage of occurrence in each class.

Recall answers to the question of what proportion of a class is correctly identified. In statistical terminology, this is the True Positive rate and it is formulated as

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

- Similar to precision, recall can have macro and weighted version calculations which is very useful in multi-label cases.

Many classification methods suffer from a precision and recall tradeoff: as more instances are correctly labeled, more mislabelling towards other classes is made. Therefore, there should be a new way to balance out and give a good estimate of the dataset. F1 score resolves this problem by combining both metrics:

$$fscore = \frac{precision * recall}{precision + recall}$$

Results and Discussion

This section touches on the result achieved by the 5 above algorithms on two different datasets as well as the discussion.

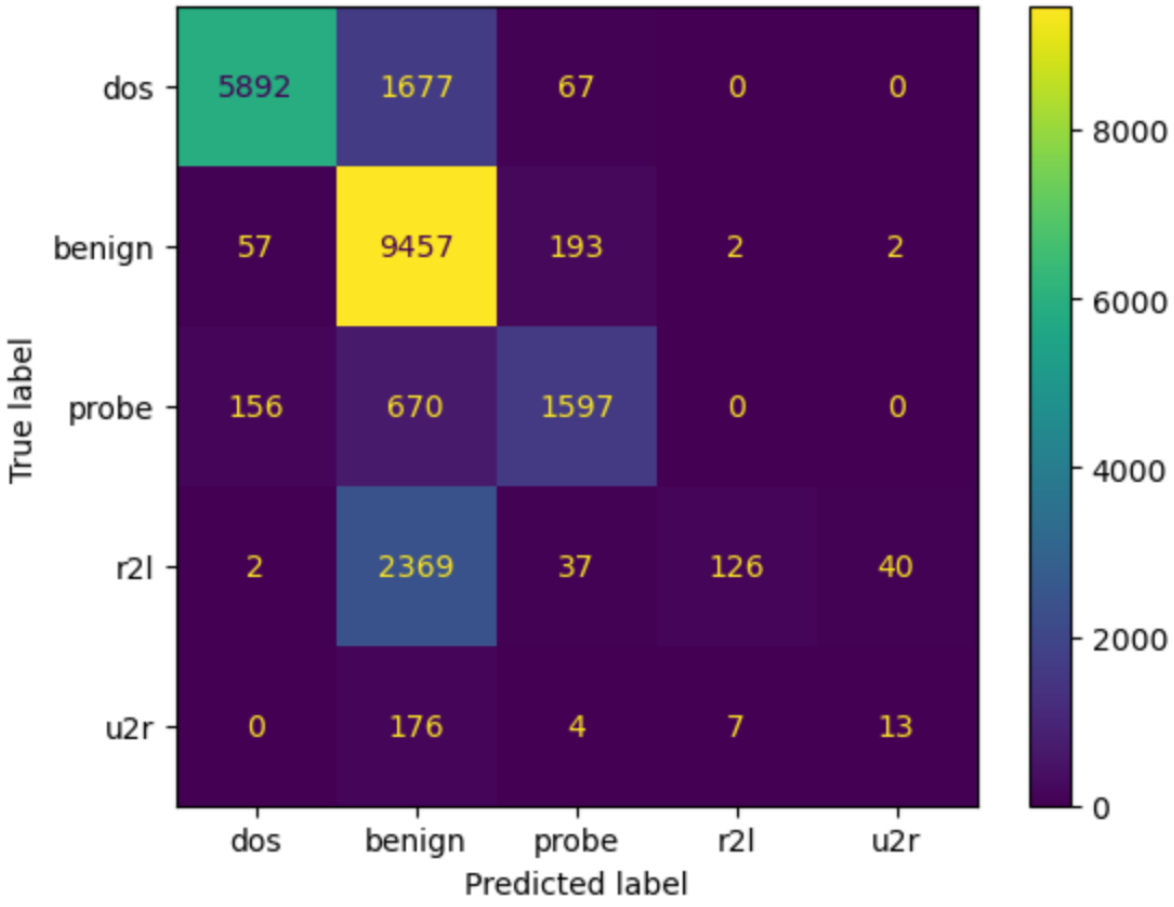
NSL-KDD

	Accuracy	Precision		Recall		F1 Score		Train time	Test time
		Macro	Weighted	Macro	Weighted	Macro	Weighted		
KNN	0.76	0.73	0.81	0.50	0.76	0.52	0.72	0.913s	10.632s
RF	0.74	0.89	0.81	0.47	0.74	0.47	0.70	15.161s	0.390s
LR	0.75	0.65	0.77	0.53	0.75	0.53	0.72	1074.22s	0.017s
SVM	0.77	0.81	0.81	0.51	0.77	0.53	0.73	26.845s	8.562s
MLP	0.766	0.79	0.79	0.52	0.76	0.53	0.73	90.374s	0.039s

Summary table

In the below sections, we present the confusion matrix for each of the method, as well as the precision score, recall score, f-score for each of the classification

K-nearest neighbors

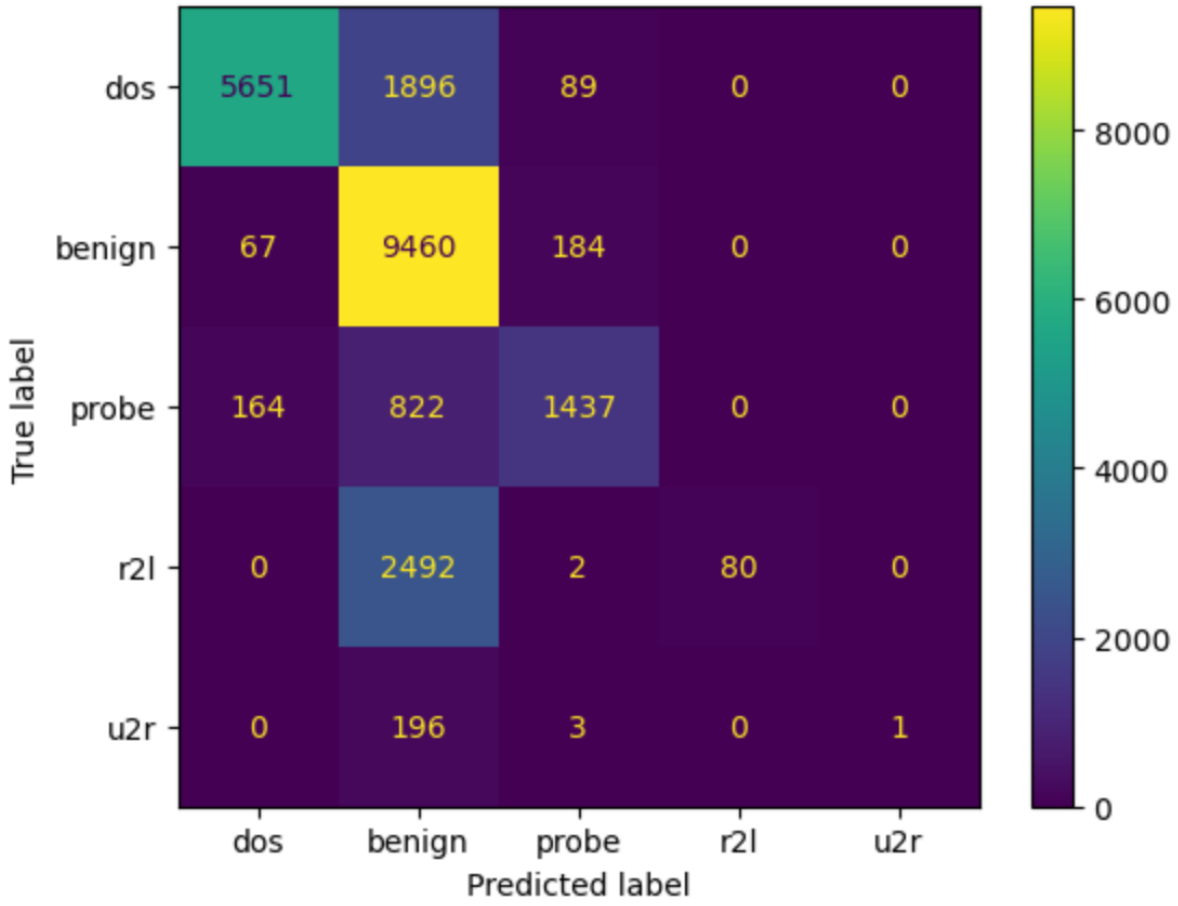


Confusion Matrix

	precision	recall	f1-score	support
benign	0.66	0.97	0.79	9711
dos	0.96	0.77	0.86	7636
probe	0.84	9,66	0.74	2423
r2l	0.93	0.05	0.09	2574
u2r	0.24	0.07	0.10	200

Classification Report

Random Forest

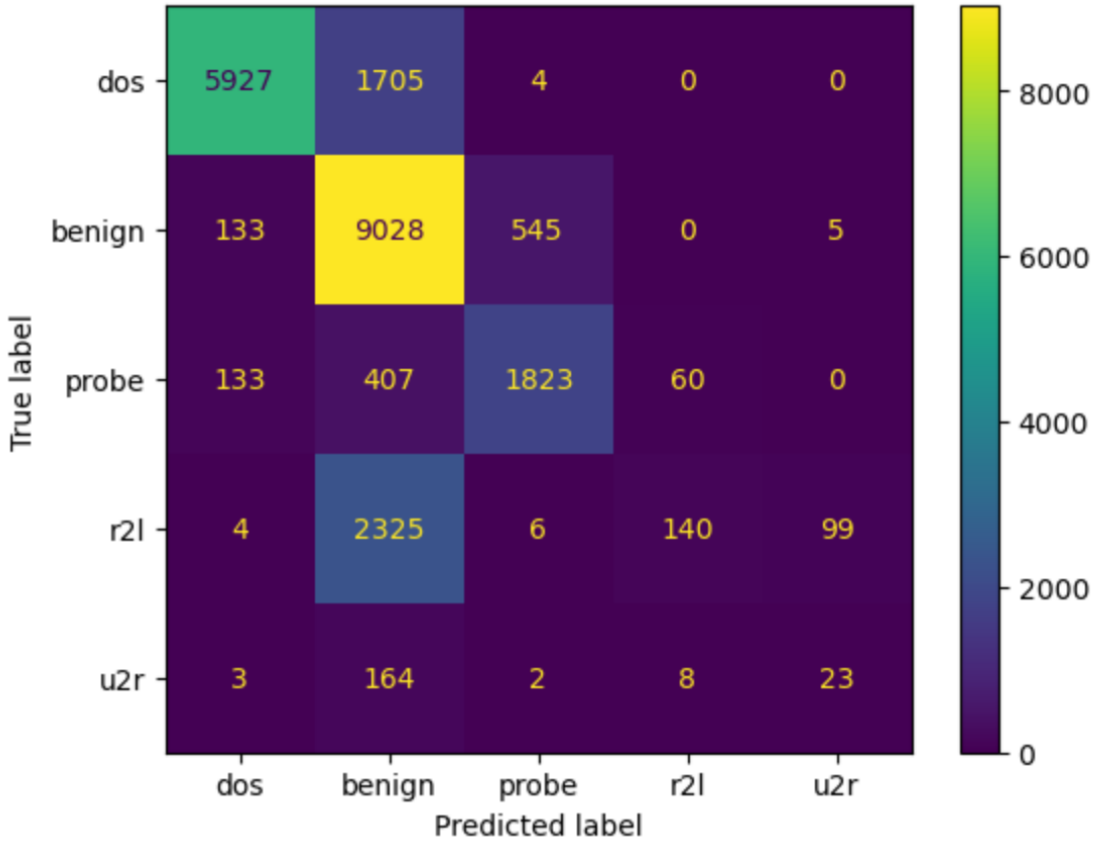


Confusion Matrix

	precision	recall	f1-score	support
benign	0.64	0.97	0.77	9711
dos	0.96	0.74	0.84	7636
probe	0.84	0.59	0.69	2423
r2l	1.0	0.03	0.06	2574
u2r	1.0	0.01	0.01	200

Classification Report

Logistic Regression



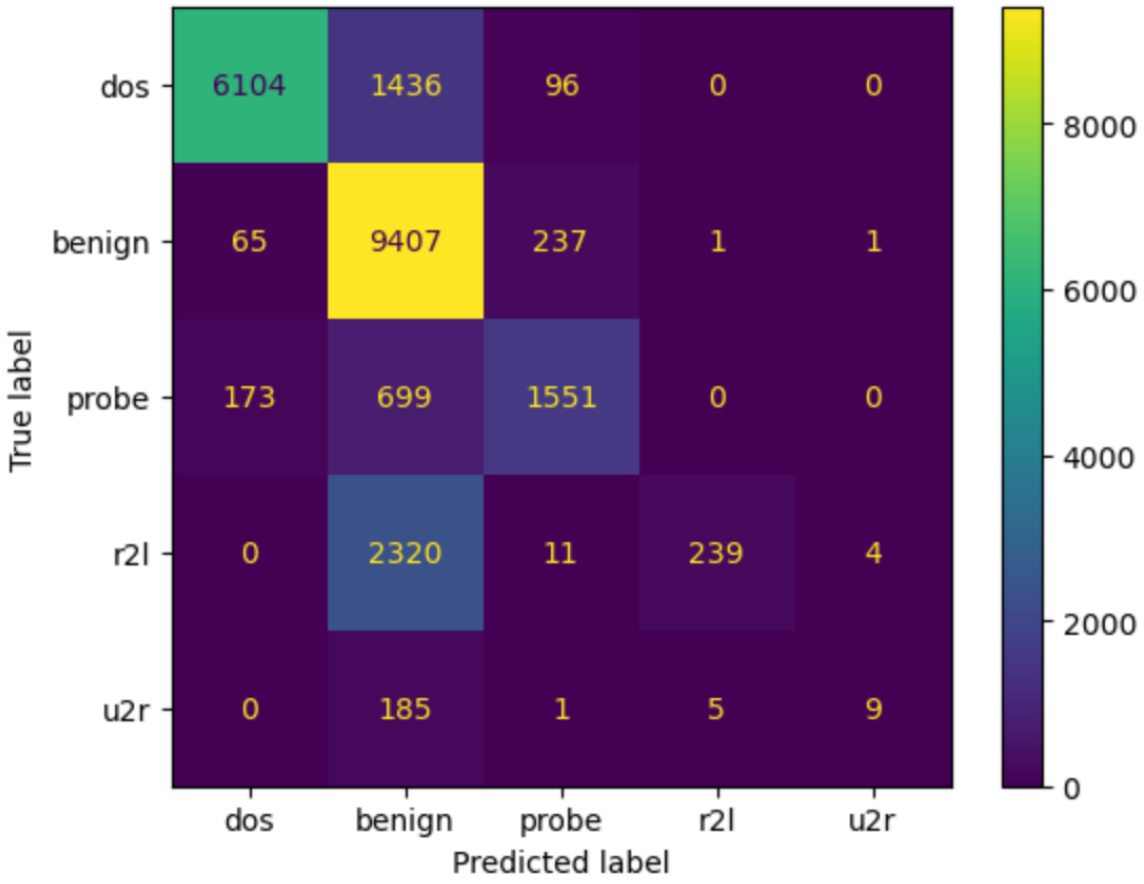
Confusion Matrix

	precision	recall	f1-score	support
benign	0.66	0.93	0.77	9711
dos	0.96	0.78	0.86	7636
probe	0.77	0.75	0.76	2423
r2l	0.67	0.05	0.10	2574

u2r	0.18	0.12	0.14	200

Classification Report

Support Vector Machine



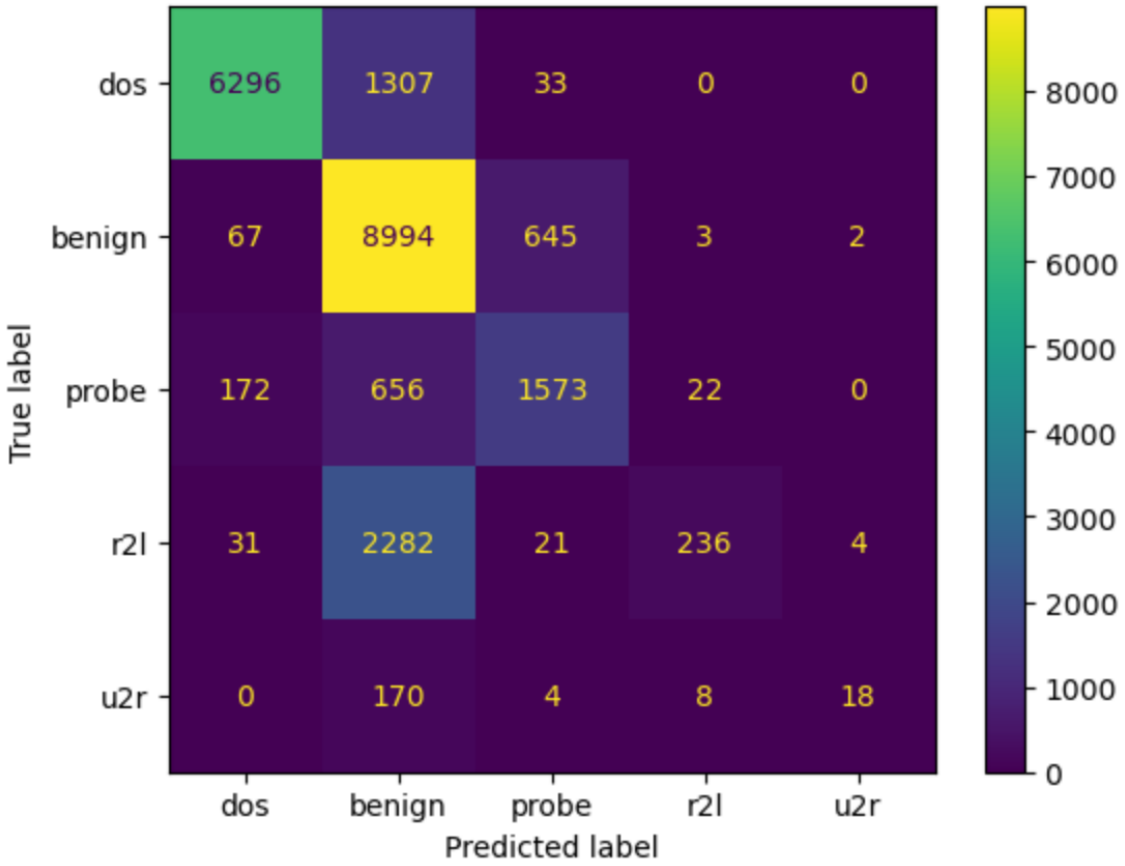
Confusion Matrix

	precision	recall	f1-score	support
benign	0.67	0.97	0.79	9711
dos	0.96	0.80	0.87	7636
probe	0.82	0.64	0.72	2423
r2l	0.98	0.09	0.17	2574

u2r	0.64	0.04	0.08	200

Classification Report

Multi-layer perceptron



Confusion Matrix

	precision	recall	f1-score	support
benign	0.67	0.93	0.78	9711
dos	0.96	0.82	0.89	7636
probe	0.69	0.65	0.67	2423
r2l	0.88	0.09	0.17	2574
u2r	0.75	0.09	0.16	200

Classification Report

Discussion

- Out of the five algorithms, Random Forest scores the highest in both precision scores, with 0.89 for macro precision and 0.81 for weighted precision. However, we can see this also results in a low recall score, at 0.47 and 0.74, the lowest, respectively. This means that Random Forest detected attack has high precision, however, it does not cover much of the whole dataset.
- In terms of macro recall, which is the arithmetic means of each class, logistic regression scores the highest at 0.53, followed by multi-layer perceptron. Both of these classifiers are better at predicting attacks than the other algorithms, but the trade is that precision is not high.
- In terms of f-score, we can observe that both Support Vector Machine and Multi-layer perceptrons perform the best at 0.53 and 0.73 for macro and weighted f-score, respectively. The worst performing algorithm is logistic regression, at 0.47 for macro f1 and 0.70 for weighted score.
- Logistic regression has the longest fit time at 1074.22 second, and this is due to opting for LogisticRegressionCV in scikit-learn, which performs built-in hyperparameter tuning with cross-validation. In the experiment, a 5-fold cross-validation was carried out. Due to its simplicity, the algorithm only took 0.017 seconds to predict based on the test set. Surprisingly, the K-nearest neighbor has the longest predicting time at 10.632 seconds although it has the lowest fitting time at 0.913 seconds
- All of the above algorithms as one characteristic in common which is the false positive instances are almost always from the benign class. This is due to the imbalance of data with the benign majority, hence the trained model assumes that it is the distribution of the population.

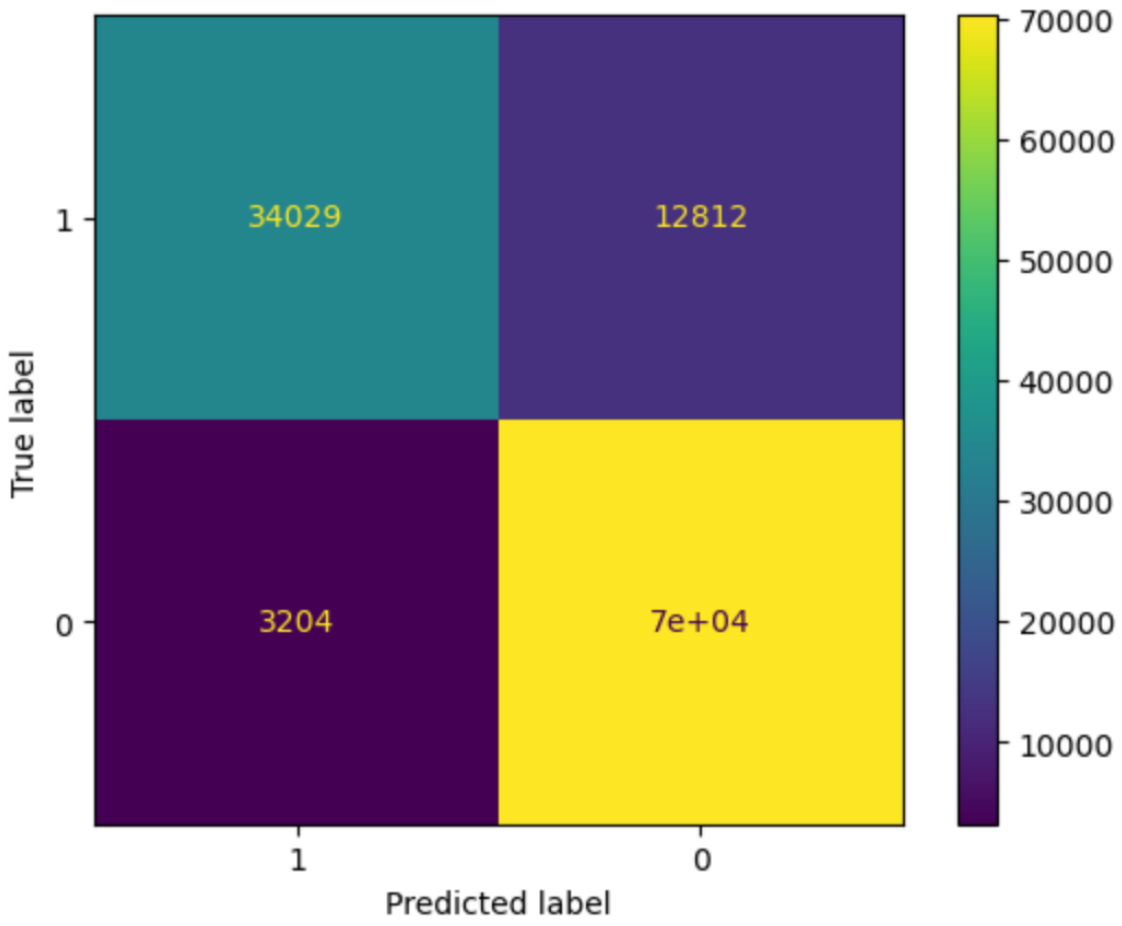
TON_IoT

	Accuracy	Precision	Recall	F1 Score	MAE	ROC AUC	Train time	Test time
KNN	0.86	0.91	0.73	0.81	0.13	0.84	0.046s	35.108s
RF	0.87	0.92	0.73	0.81	0.12	0.85	298.422s	18.063s
LR	0.69	0.87	0.23	0.36	0.31	0.61	15.35s	0.017s
SVM	0.70	0.87	0.27	0.41	0.3	0.62	1193.848s	15.065s
MLP	0.81	0.90	0.56	0.69	0.19	0.76	198.703s	0.101s

Summary Table

In the below sections, we present the confusion matrix for each of the method, as well as the precision score, recall score, and f-score for each of the classification

K-nearest neighbors

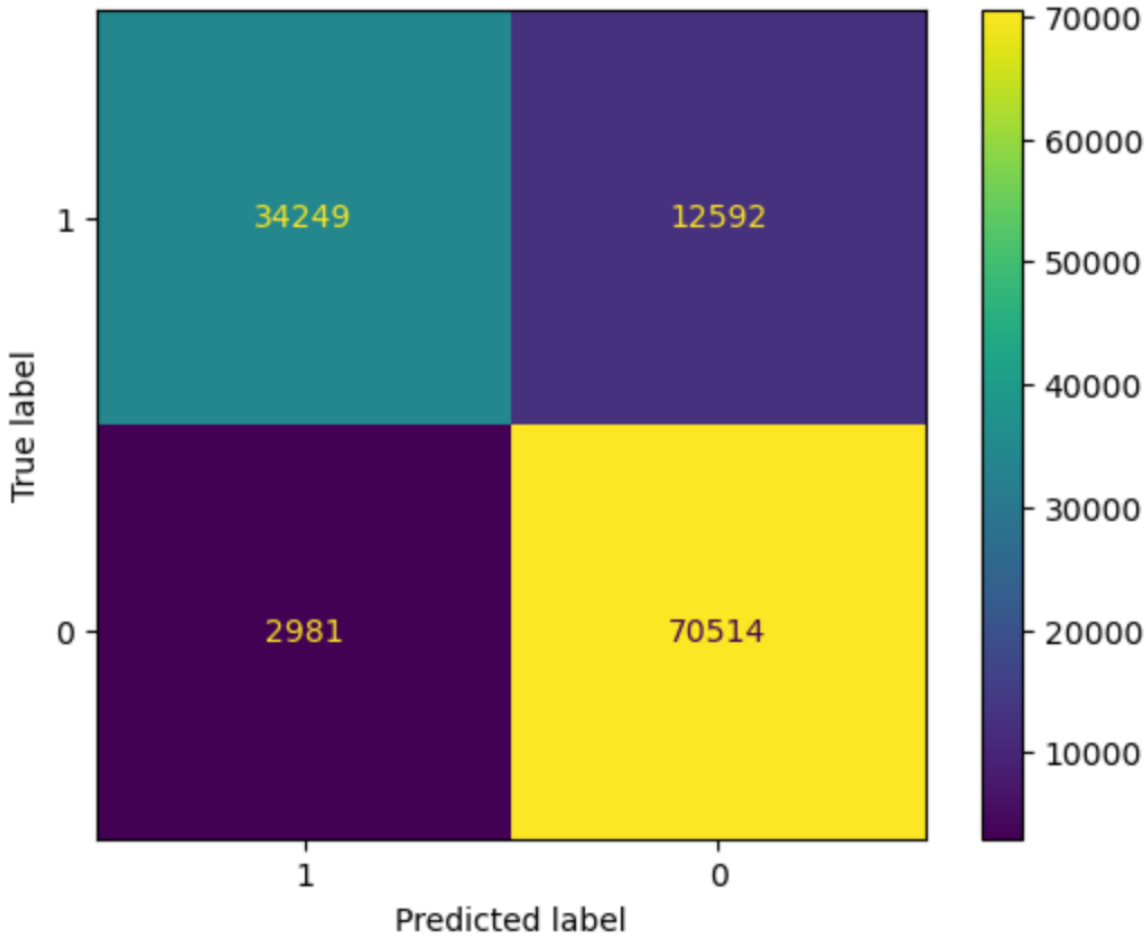


Confusion Matrix

	precision	recall	f1-score	support
0	0.85	0.96	0.90	73495
1	0.91	0.73	0.81	46841

Classification Report

Random Forest

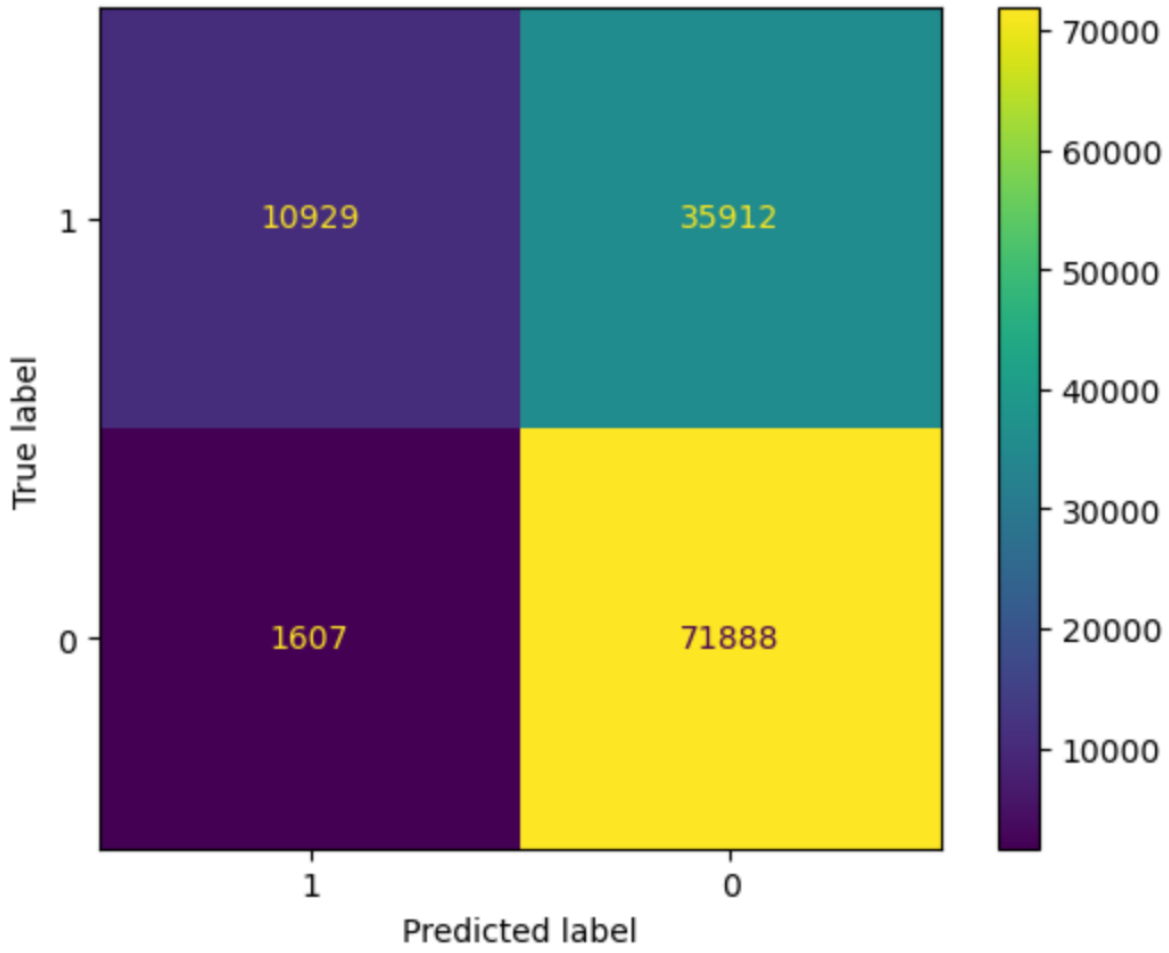


Confusion Matrix

	precision	recall	f1-score	support
0	0.85	0.96	0.90	73495
1	0.92	0.73	0.81	46841

Classification Report

Logistic Regression

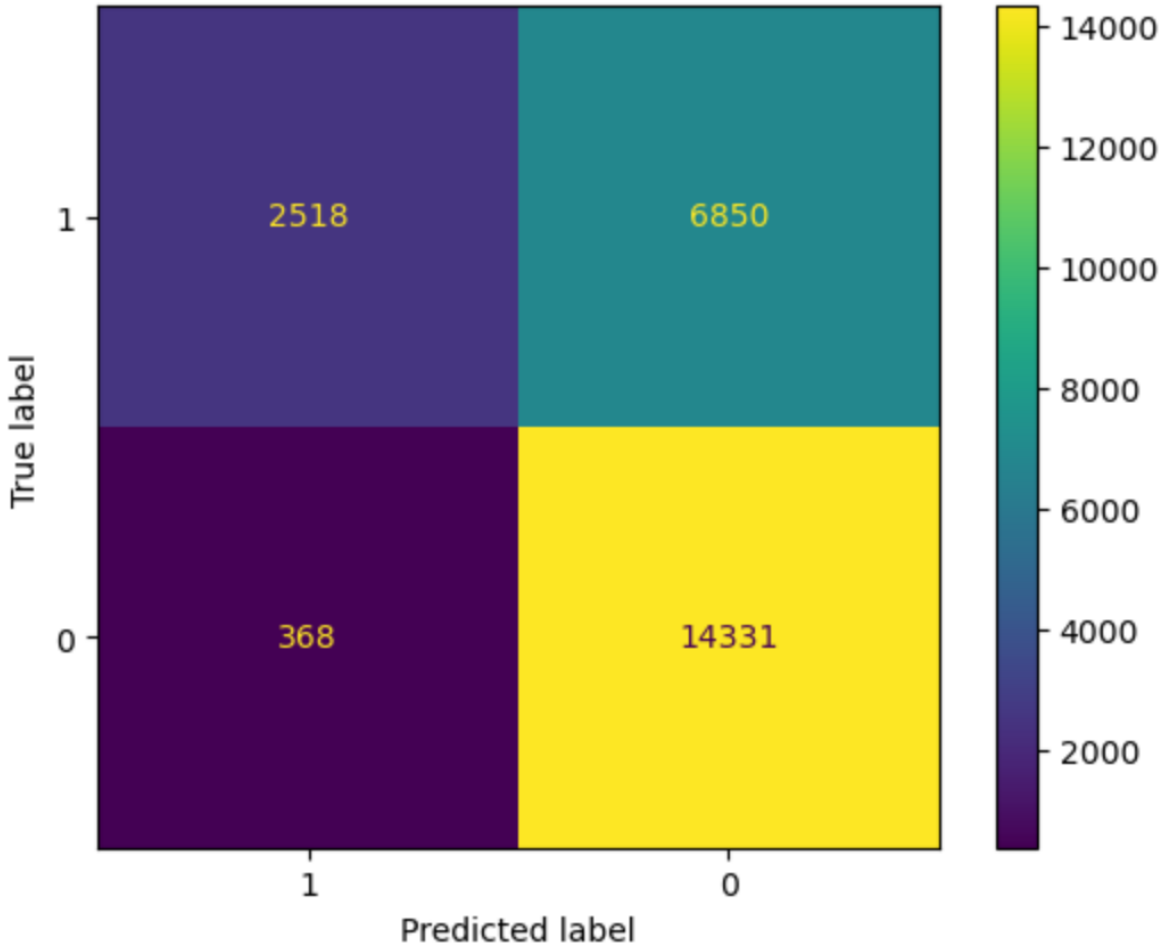


Confusion Matrix

	precision	recall	f1-score	support
0	0.67	0.98	0.79	73495
1	0.87	0.23	0.37	46841

Classification Report

Support Vector Machine

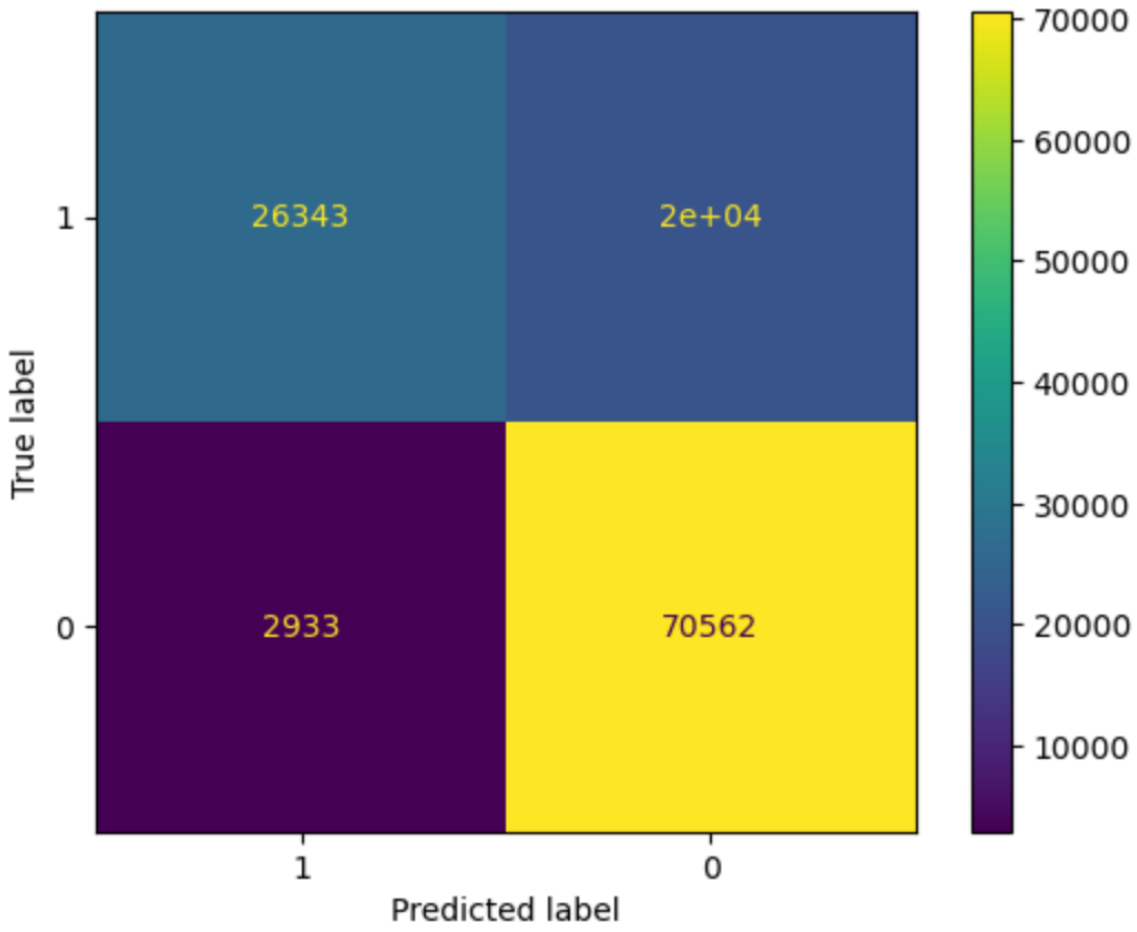


Confusion Matrix

	precision	recall	f1-score	support
0	0.68	0.97	0.80	14699
1	0.87	0.27	0.41	9368

Classification Report

Multi-layer perceptron



Confusion Matrix

	precision	recall	f1-score	support
0	0.77	0.96	0.86	73495
1	0.90	0.56	0.69	46841

Classification Report

Discussion

- For the TON_IoT dataset, Random Forest scored the highest precision score at 0.92, followed by KNearest Neighbor at 0.91. The lowest precision score is Logistic Regression and Support Vector Machine, at 0.87.
- For the true positive rate or Recall, which measures how well positive instances were identified, Random Forest and KNearest Neighbor once again scored the highest out of five methods, at 0.73, and this is reflected through their f-score at 0.81.
- Logistic Regression identified 23 percent of the total attack, which is the lowest among all five methods, followed by support vector machines at 0.27. For this dataset, both of these methods perform the lowest. However, it should be noted that due to limited resources and the high number of data points, resampling was done to support vector machines, which cuts off 80 percent of the actual dataset. Still, we see SVM taking the longest fitting time at 1193 seconds
- ROC Area under the Curve, which assesses the model's ability to discriminate between positive and negative instances, once again saw Random and KNearest neighbor achieved the best score.
- Therefore, it is safe to assume that KNearest Neighbor and Random Forest are the most suitable algorithms for this task. Random Forest takes significantly longer than KNearest Neighbor, however, its inference time is twice as fast. Logistic Regression and support vector machines are unsuitable to analyze this dataset.

References

Aicrowd author (2023), "Amazon kdd cup 23 multilingual recommendation challenge", retrieved August 2023. Available at:
https://www.aicrowd.com/challenges/amazon-kdd-cup-23-multilingual-recommendation-challenge

scikit-learn author, "Decision Trees", retrieved August 2023. Available at
https://scikit-learn.org/stable/modules/tree.html#classification

Deepak Senapati, "GridSearch vs RandomSearch", retrieved August 2023. Available at:
https://medium.com/@senapati.dipak97/grid-search-vs-random-search-d34c92946318