The School of Mathematics

THE UNIVERSITY *of* EDINBURGH

Cyber Security: Autoencoders and Variational Autoencoders for Anomaly Detection

by

Jialong He

Dissertation Presented for the Degree of
MSc in Statistics with Data Science

June 2023

Supervised by
Dr Miguel de Carvalho and Dr Amanda Lenzi

Executive Summary

To realise the anomaly detection approaches on the dataset UNSW-NB15_1, five machine learning models are adopted, including simple/deep Autoencoders, Variational Autoencoders, One-Class Support Vector Machine and Isolation Forest. According to several metrics, especially recalls for the anomalies in these models, compare and derive the simple AE as the optimal model with nearly 100% recall and only 32% precision for anomalies and 96% accuracy. This demonstrates the exploitability of using AEs for anomaly detection in the field of cyber security. Last, with the assistance of SHAP explainability framework, the interpretation of the results from the simple AE is attainable with the most influential feature "swin", namely source TCP window advertisement value.

Acknowledgments

I am grateful to the supervisors Dr Miguel de Carvalho and Dr Amanda Lenzi and the programming helper James Chok during the one-month period of this challenging project. Also, I want to thank experts from Lloyds Banking Group: George Deskas, Callum Scott Hodgkinson and Dimitrios Ntakoulas for giving the introduction of this project and lots of explanations within this month. Finally, I appreciate some of my dear classmates who accompany with me throughout the project. We are making progress together!

University of Edinburgh – Own Work Declaration

This sheet must be filled in, signed and dated - your work will not be marked unless this is done.

Name: Jialong He

Matriculation Number: s2281875

Title of work: Cyber Security: Autoencoders and Variational Autoencoders for Anomaly Detection

I confirm that all this work is my own except where indicated, and that I have:

- Clearly referenced/listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Not sought or used the help of any external professional academic agencies for the work
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Complied with any other plagiarism criteria specified in the Course handbook

I understand that any false claim for this work will be penalised in accordance with the University regulations (`https://teaching.maths.ed.ac.uk/main/msc-students/msc-programmes/statistics/data-science/assessment/academic-misconduct`).

Signature Jialong He

Date 29/06/2023

Contents

1 Introduction **1**

2 Data **1**

2.1 About the dataset UNSW-NB15 1

2.2 Exploratory data analysis 1

2.3 Pre-processing 3

3 Methods and Models **4**

3.1 Performance metrics 4

3.2 Autoencoders 5

3.2.1 Reconstruction error 5

3.2.2 Simple autoencoders 5

3.2.3 Deep Autoencoders 7

3.3 Variational Autoencoders 8

3.4 One-Class Support Vector Machine 10

3.5 Isolation Forest 11

4 Results **12**

4.1 Model comparison 12

4.2 Model explanation 12

5 Conclusions **14**

A Code **16**

B Dataset **16**

List of Tables

1 Metrics comparison between different models 12

List of Figures

1 Proportions of Labels. 2

2 Distributions of some features. 2

3 The architecture of a simple AE with one hidden layer. 6

4 The progress of a simple AE's model fitting and the metrics and its confusion matrix. 6

5 The architecture of a deep AE with three hidden layers. 7

6 The progress of a deep AE's model fitting and the metrics and its confusion matrix. . 7

7 A simple illustration of the architecture of a VAE. 8

8 The progress of a VAE's model fitting and the metrics and its confusion matrix. . . . 9

9 An illustration of how OC-SVM (with a RBF kernel) works. 10

10 The confusion matrix of the predication on the testing set. 10

11 Outliers detection in Isolation Forest. 11

12 The confusion matrix of the predication on the testing set and the average path lengths. 11

13 The SHAP explanation on one single prediction (two examples). 12

14 The SHAP explanation on a bunch of predictions. 13

15 Top ten explainable features with their absolute SHAP values. 14

1 Introduction

Anomaly detection (also outlier detection or novelty detection) is a process of identifying rare events or observations which deviates significantly from their group in the field of data analysis. One of the popular applications of anomaly detection is cyber security in the banking system since it can detect those malicious activity, events or records.

There are multiple anomaly detection methods and we are using traditional statistical models, like One-Class SVM and Isolation Forest as well as deep learning neural network models, like Autoencoders and Variational Autoencoders which are recently prominent in this field so as to perform semi-supervised approaches given the dataset UNSW-NB15 in this report. This report also aims to compare the models' performance metrics such as recall and investigate the interpretability of the models based on SHAP.

2 Data

Before applying anomaly detection methods to the target dataset, it is necessary to explore the dataset, understand its structure, composition, and distributions, and perform essential preprocessing steps.

2.1 About the dataset UNSW-NB15

The dataset UNSW-NB15 was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours. There are nine types of attack in UNSW-NB15, such as Fuzzers, Analysis, Backdoors, DoS and so on. However, we do not care about identifying which type of an attack is but whether a network interaction is an attack or not in this project, namely, the goal is binary with 0 for benign and 1 for anomalous (attack).

The dataset used in this project is a subset of UNSW-NB15 called UNSW-NB15_1.

2.2 Exploratory data analysis

There are 700,001 rows and 49 columns in the dataset UNSW-NB15_1 with each row is a data point or record and each column is a feature.

Among these records, there are 22,215 attacks which is approximately 3.17% of the total. The proportion of two types of all records is shown in the below Figure 1.

Next we shall have a look at the distribution of some features' values in Figure 2. Note that since there are above 40 features in the dataset so it is hard for us to look at them all here.

From the top left density plot of feature "sport", it seems that vast majority values are uniformly distributed from 0 to 65,000 while from "sttl" and "swin" (top right and bottom left), their values are extremely skewed and concentrated. In the bottom right, we can see the count plot of different types of service used with a great number of "-" (which stands for not much used service).

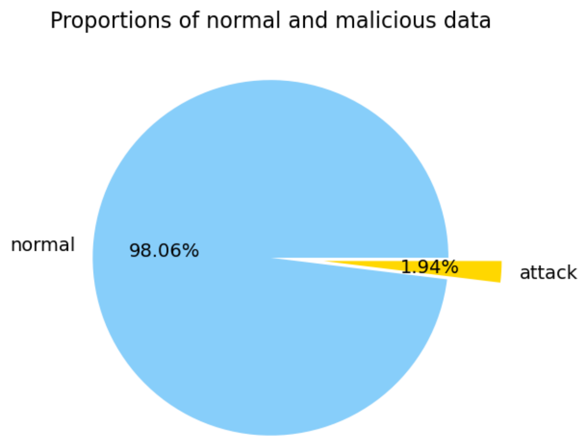

Figure 1: Proportions of Labels.

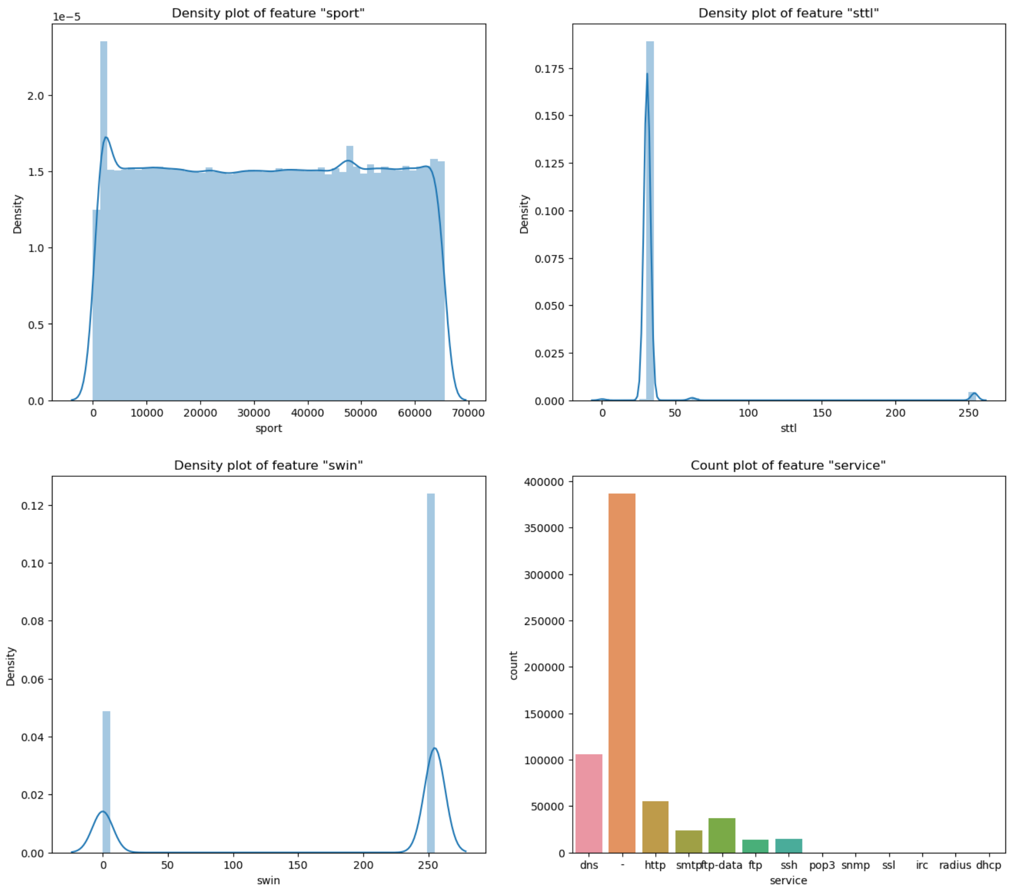

Figure 2: Distributions of some features.

2.3 Pre-processing

It is of great significance to conduct some pre-processing on the dataset in the very beginning. Firstly, since we are more interested in whether a record is an attack or not and uninterested in the exact type of attacks, the feature “attack_cat” was deleted. In addition, two IP address columns called “srcip” and “dstip” and two time columns called “Stime” and “Ltime” are deleted. Next, 63,188 duplicated records were found in the dataset and they are deleted because duplicates will definitely influence the performance of models and it seems like it is impossible that so many duplicates exist for an explainable reason. Then some problematic records with illegible characters in column “sport” and “dsport” were deleted. Besides, there are no empty or NA values in the dataset.

Now there are 636,766 records in the dataset of which 12,350 are attacks.

Several categorical columns, such as “proto”, “state” and “service”, need to be converted from string to numbers.

Since the distribution of many features are imbalanced with extremely low and high values, we are trying to apply Min-Max scaler to transform the dataset. Assume the dataset X has m columns, so the transformed ith column of X is derived as

$$X_j std = \frac{X_j - min(X_j)}{max(X_j) - min(X_j)},$$

where j is 1,2,...,m.

Finally, it is worth noting that we extract another subset (150,000 records) from the processed dataset UNSW-NB15_1 when we are performing training and fitting on OC-SVM model mainly because training and predicating on a large data size using OC-SVM is extremely time consuming (one hour or longer) and not worthwhile.

3 Methods and Models

The choice of models and the methods used during the modelling are introduced in this section. Besides, we will obtain the performance of different models on the dataset.

3.1 Performance metrics

Considering the label in the dataset consists of 0 for normals and 1 for attacks, we define that

- True positive (TP) is the case in which both the prediction label and the true label are 1;
- True negative (TN) is the case in which both the prediction label and the true label are 0;
- False positive (FP) is the case in which the prediction label is 1 but the true label is 0;
- False negative (FN) is the case in which the prediction label is 0 but the true label is 1.

We refer to the following four metrics in assessing our models' performance in this report.

1. Accuracy: accuracy measures the overall correctness of the model's predictions. It calculates the ratio of correctly classified instances (both true positives-anomalies and true negatives-normals) to the total number of instances. Accuracy focuses on the overall performance of the model's prediction.
$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$
2. Precision: precision measures the accuracy of positive predictions made by the model. It calculates the ratio of true positive predictions (correctly predicted positives) to the total number of positive predictions (true positives plus false positives). Precision focuses on the quality of positive predictions.
$$\text{Precision} = \frac{TP}{TP+FP}$$
3. Recall: recall, also known as sensitivity or true positive rate, measures the model's ability to identify all positive instances. It calculates the ratio of true positive predictions to the total number of actual positive instances (true positives plus false negatives). Recall focuses on the completeness of positive predictions. In the process of evaluating the model's performance, we will pay more attention to recall compared to other metrics. The reason is that in the field of cyber security, for example, considering whether a banking account is under attack or not, we would rather misjudge a normal account as an attack than misjudge an attack as normal.
$$\text{True positive rate(TPR)} = \text{Recall} = \frac{TP}{TP+FN}$$
4. F-beta Score: the F-beta score is a generalized version of the F1 score that allows control over the balance between precision and recall. It uses a positive real factor beta, where beta is chosen such that recall is considered beta times as important as precision. Since we favour recall over the other metrics, a large beta=2 is considered in our case, that is to say, we reckon that the importance of recall is double of precision.
$$\text{F}_\beta = (1+\beta^2)\frac{precision * recall}{\beta^2 * precision + recall},$$
where $\beta = 2$ is the constant factor.

3.2 Autoencoders

3.2.1 Reconstruction error

Before we delve into the introduction of autoencoders, there is something important and elementary we have to know, which is reconstruction errors.

The reconstruction error is a general notion which actually measures the discrepancy between the generated data and original ones and it is the key especially when someone wants to perform autoencoders. So how we define a threshold based on those reconstruction errors when facing the testing set? We select mean absolute error (MAE) to decide our threshold in this project. Alternatives include mean squared error (MSE) loss, binary cross-entropy (BCE) loss, Kullback-Leibler divergence (KLD) loss and so on.

MAE measures the discrepancy between the predications and original values. It is chosen to define the reconstruction error here once the model is trained. Assume the dataset consists of n records and $x_i, (i = 1, 2, ..., n)$ is the ith record, so the MAE of records can be calculated by the mean of these absolute differences

$$e_{\text{mae}} = \frac{\sum_{i=1}^{n} |\hat{x_i} - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n}$$

Then we create an interval which starts from MAE and ends at MAE plus multiple standard deviation (of those absolute values above), e.g., a hundred times of standard deviation and let the threshold be equal to different values in this interval. For each threshold, we obtain its metrics and grab the optimal one after comparison between these different thresholds.

3.2.2 Simple autoencoders

Autoencoders (AEs) are a type of neural networks in machine learning used to learn high-dimensional data pattern and reduce the dimensionality efficiently. A simple autoencoder (AE) consists of two functions: an encoding function that transforms the input data, and a decoding function that recreates the input data from the encoded representation. In detail, the encoder reduces the dimensionality of a dataset while the decoder essentially recover the reduced data from low dimension to the original dimension. The goal of such a process is to try to reconstruct the original input and see if AEs learn the data's pattern well enough.

One popular application for AEs is anomaly detection. By learning most salient features in the training dataset which only have normal data, the model is able to reproduce the accurate pattern of the training data, also called reconstruction. The difference between original training data and the reconstructed one is called reconstruction errors. Then a threshold is defined by the reconstruction errors. When confronted with a dataset with potential anomalies, AEs could also obtain a reconstruction of those anomalies but this time there will exist a greater reconstruction error. We classify those which beyond the threshold as anomalies and vice versa. The following Figure 3 illustrates how a simple AE works[1].

After applying a simple AE with optimal hyper-parameters (learning rate of Adam optimizer equal to 0.01; sigmoid activation function of the output layer) in Python based on TensorFlow, we obtain the optimal metrics which are shown in following Figure 4. From the top two sub-figures in Figure 4, we can see the AE model is converging as the training and validation accuracy and their MAE loss

[1]Figure source: https://commons.wikimedia.org/wiki/File:Autoencoder$_{s}$*chema.png*

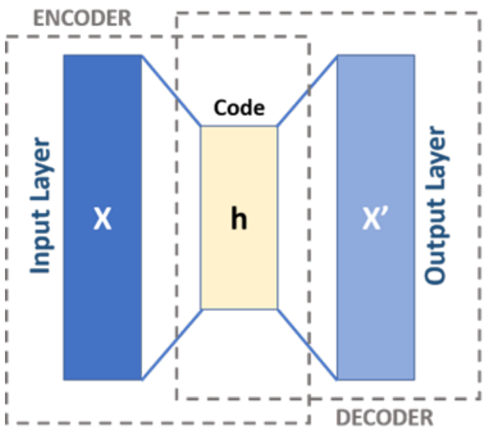

Figure 3: The architecture of a simple AE with one hidden layer.

are moving towards stability. It is worth noting that since we are training only on the normal dataset (which only consists of non-anomalous data) but validating on the mixed dataset (which consist of both anomalous and normal data), it is reasonable that the accuracy of validation is a bit lower while its loss is a bit higher than training's. The bottom left one shows us the optimal threshold is chosen when the F2-Score reaches the maximum. The range of thresholds is from MAE minus one times the standard deviation to MAE plus five hundred times the standard deviation and we will stick to this range of potential thresholds in the subsequent models. The last one (bottom right) shows the corresponding numbers of TN, FN, FP and TN from top to bottom and left to right which consist of the confusion matrix of the testing set. It can be easily calculated that the recall is nearly 1.

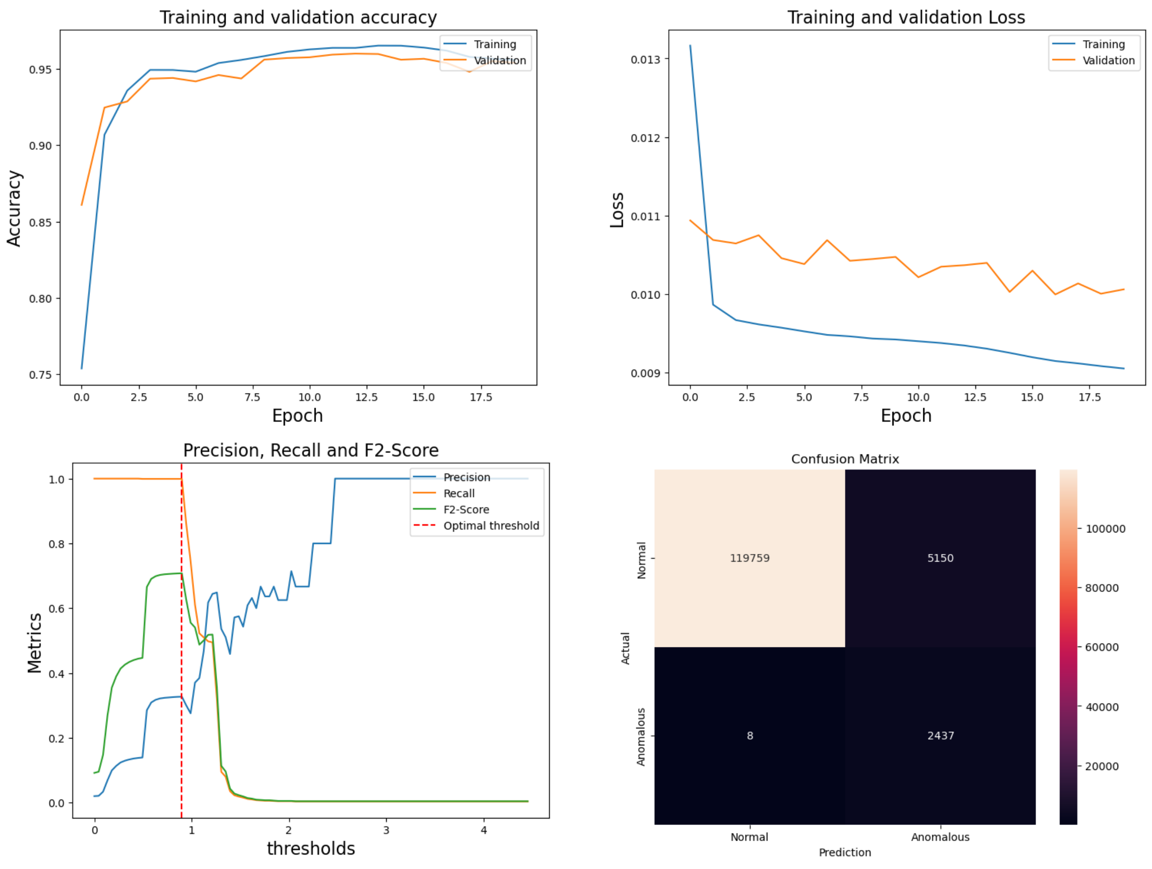

Figure 4: The progress of a simple AE's model fitting and the metrics and its confusion matrix.

3.2.3 Deep Autoencoders

A deep AE is nothing but just a simple AE with several hidden layers. Figure 5 shows the architecture of a deep AE model with 3 hidden layers.

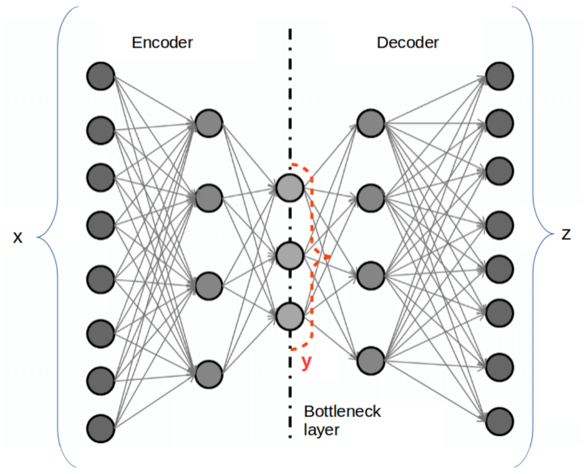

Figure 5: The architecture of a deep AE with three hidden layers.

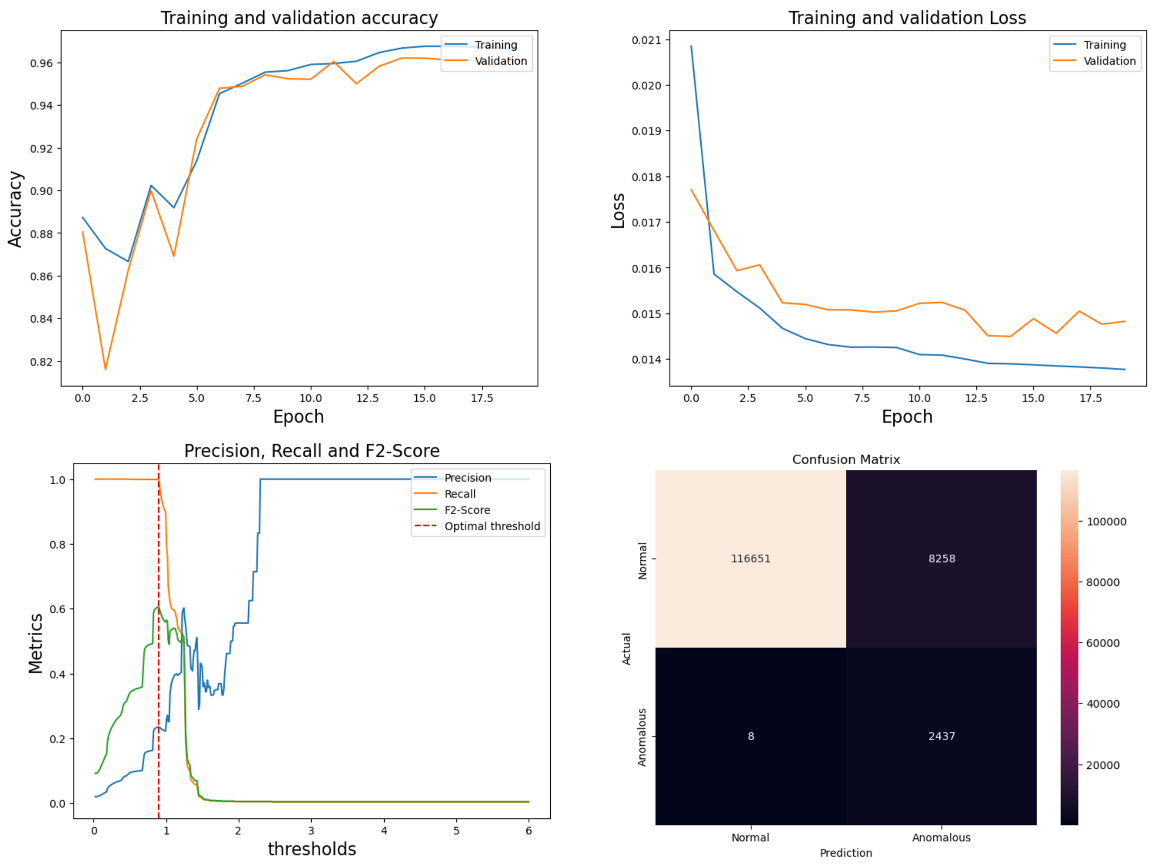

Figure 6: The progress of a deep AE's model fitting and the metrics and its confusion matrix.

When performing the deep AE model, we maintain using previous hyper-parameters (learning rate; activation function). In Figure 6, we can see that during the fitting stage it takes the training and validation accuracy more epochs to converge; the loss plot looks fine as before; the optimal threshold is found as F2-Score reaches its maximum but along with a notably lower recall which implies the deep AE is worse than previous simple AE; the number of FN increases by a few while the number of FP surges apparently in the confusion matrix which also indicates the great performance falling of the deep AE.

3.3 Variational Autoencoders

Variational Autoencoders (VAEs) are more generative models on the basis of AEs. The structure of a VAE is basically similar to AEs which has an encoded layer taking the input and a decoder layer producing the output. The basic idea is that, instead of taking the original dataset as input, a VAE actually learns the distribution of the dataset over the latent space and then generates new data samples from that distribution and finally decodes the sampled data with reconstruction errors.

To make the progress a bit detailed, we are going to introduce a bit more about how a VAE works. Consider a hidden variable z from the latent space which follows a prior distribution $p(z)$. A data point x_i can be sampled from the conditional distribution $p(x|z)$. Now we would like to know the posterior distribution of z given the data x. By Bayes theorem,

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

However, the calculation of $p(x)$ in the denominator is extremely difficult which turns out to be an intractable distribution. Hence, variational inference (VI) is applied to tackle with this problem. VI is a technique which aims to approximate complex distributions but we cannot illustrate its principle since it is beyond the scope of this report. Finally, the goal becomes finding another Gaussian distribution $q(z|x)$ with some mean and covariance. Kullback–Leibler (KL) divergence is used to ensure the similarity between $p(x|z)$ and $q(z|x)$.

All in all, we can use the original data to infer the hidden variables from $q(z|x)$ during the encoding part and regenerate the data from $p(x|z)$. The following Figure 7 simply shows how a VAE works[2].

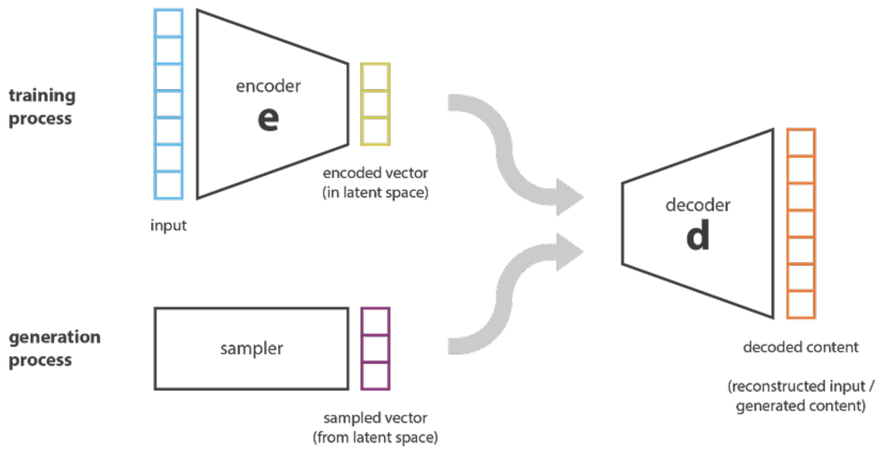
training process
encoder
e
encoded vector (in latent space)
input
decoder
d
decoded content
(reconstructed input / generated content)
generation process
sampler
sampled vector (from latent space)

Figure 7: A simple illustration of the architecture of a VAE.

Similarly, Figure 8 shows the related process and outcomes. The validation accuracy is not converged; training and validation loss increase sharply compared to previous two models (this is mainly due to the methods difference since VAEs involve variational inference); optimal threshold reaches at a relatively low F2-Score; based on the values in the confusion matrix (both higher FP and FN compared with the deep AE), the calculated metrics in Table 1 indicates that the VAE's performance is even worse than the deep AE.

[2]Figure source: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

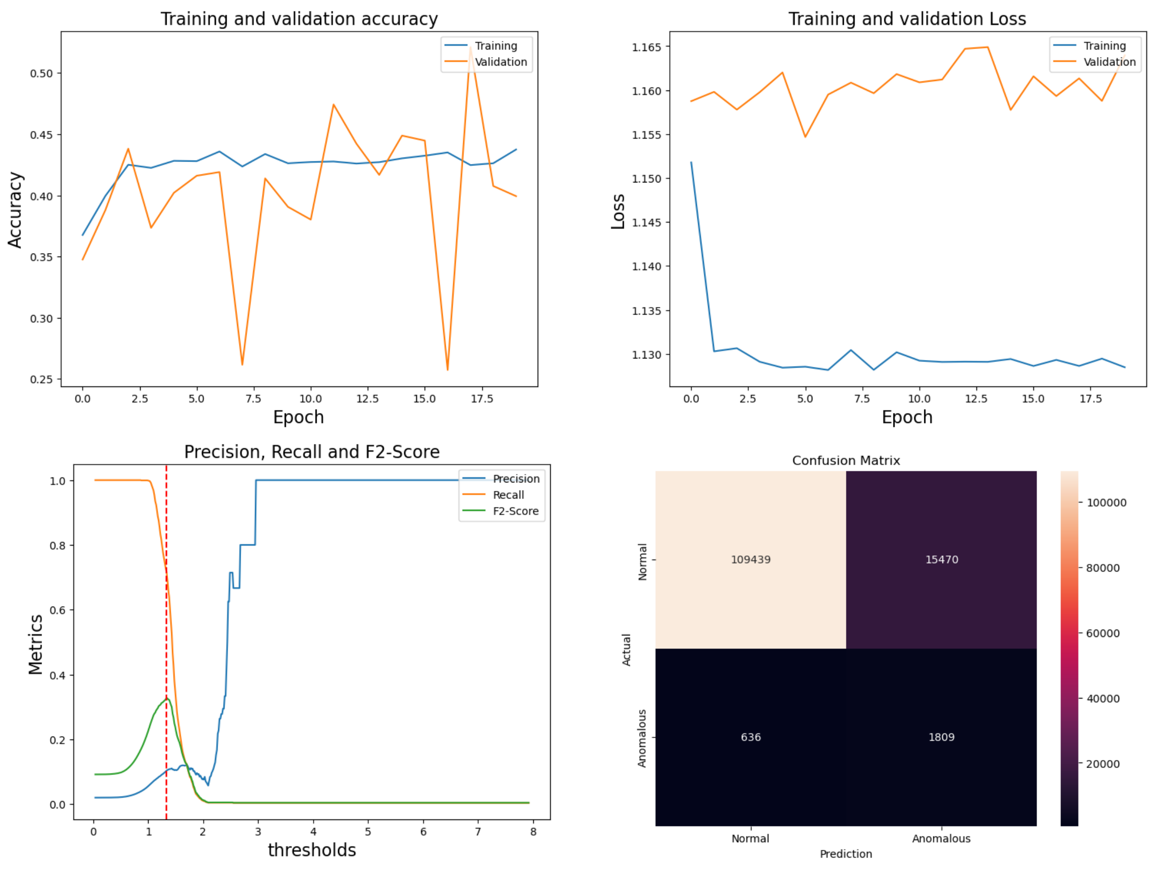

Figure 8: The progress of a VAE's model fitting and the metrics and its confusion matrix.

3.4 One-Class Support Vector Machine

One-Class Support Vector Machine (OC-SVM) is one kind of traditional support vector machine models that is specific for novelty detection of unsupervised or semi-supervised learning problems. The basic idea is that the model trains on a normal dataset and is used to detect novelties on any new dataset.

The following Figure 9 shows how OC-SVM partitions data and distinguish normal data from anomalous data [3].

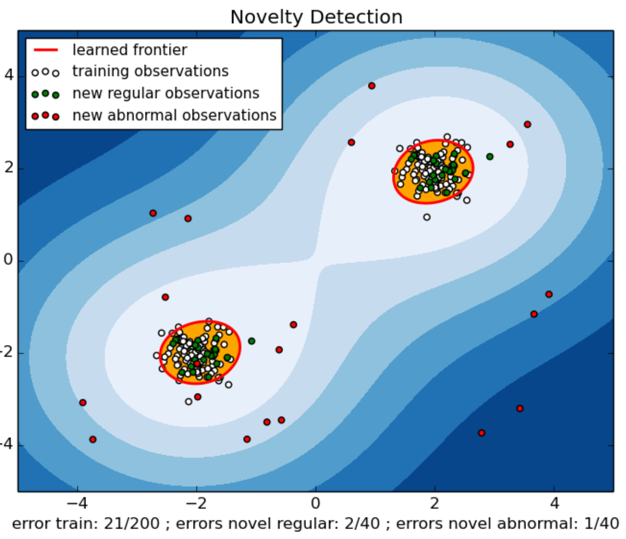

Figure 9: An illustration of how OC-SVM (with a RBF kernel) works.

Through comparing the metrics of different combinations of hyper-parameters in OC-SVM, it is found that the model performs best when the hyper-parameters “gamma” and “nu” equal to 0.5 and 0.1 correspondingly. The confusion matrix of the predication on the testing set is shown below (Figure10) with calculated accuracy 0.95, precision 0.27, recall 0.87 and F2-score 0.6.

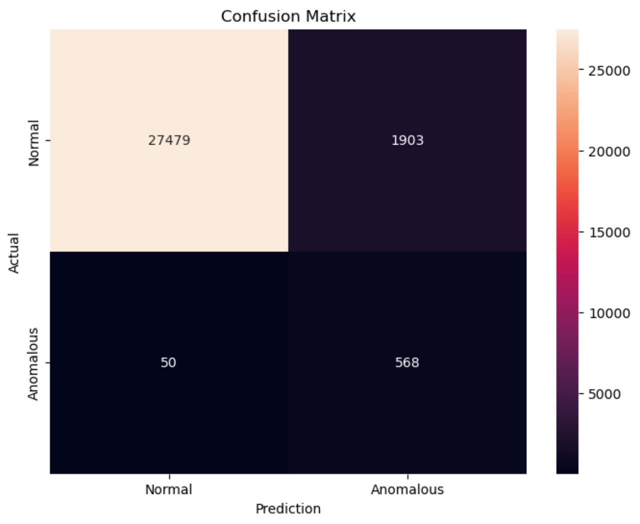

Figure 10: The confusion matrix of the predication on the testing set.

[3]Figure source: https://scikit-learn.org/stable/auto$_e$$xamples/svm/plot_oneclass.html$

3.5 Isolation Forest

Isolation forest is a machine learning algorithm especially for anomaly detection problems. The basic idea is that it recursively partition the dataset until it forms a tree structure (starting from the root node and terminating at the leaf node). Each time it isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature. Anomalies are those observations which obtain higher anomaly scores [4], or simply put, have a shorter path from the root to the leaf (Figure11).

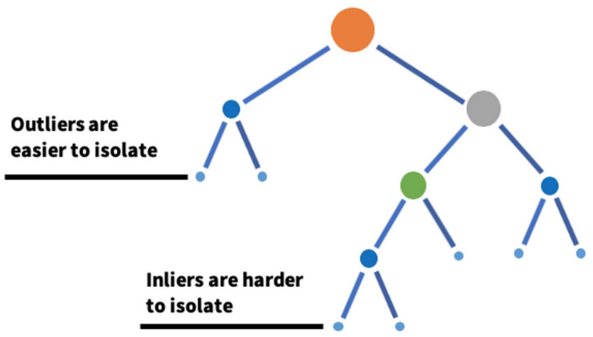

Figure 11: Outliers detection in Isolation Forest.

The best combination of the hyper-parameters in Isolation Forest ("n_estimators"=50, "max_sample"=256 and "contamination"=0.02) obtains the metrics of accuracy 0.95, precision 0.28, recall 0.98 and F2-Score 0.66. The following Figure 12 shows the confusion matrix of the predication on the testing set and the average path lengths (where one will be classified as an anomaly if its average path length is below a threshold, say 0.05 in the left plot, otherwise it will be classified as a normal point).

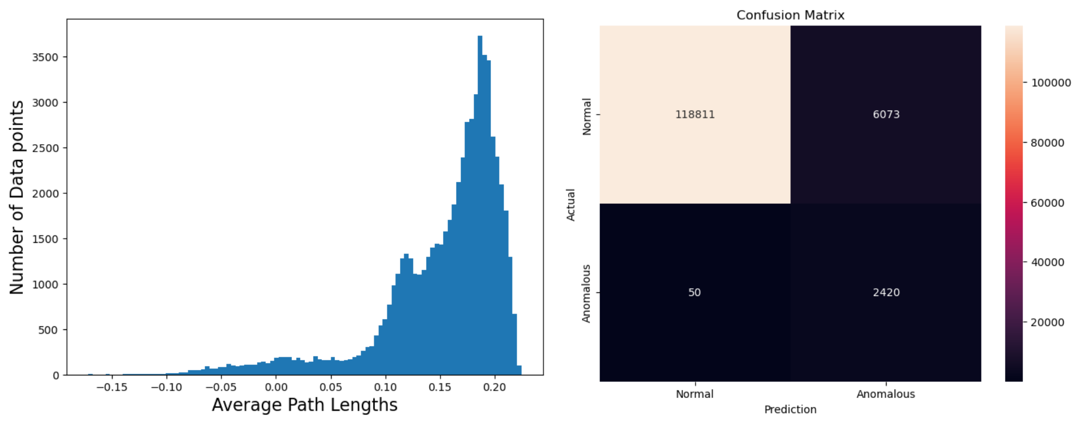

Figure 12: The confusion matrix of the predication on the testing set and the average path lengths.

[4]Figure source: https://engineering.linkedin.com/blog/2019/isolation-forest

4 Results

In this section, we bring everything we yielded together and compare the performance between different models and try to interpret the optimal model.

4.1 Model comparison

Until now, we obtained the outcomes of the testing set from five models (simple AE, deep AE, VAE, OC-SVM and Isolation Forest). We aggregate these outcomes together in Table 1.

Models	Accuracy	Precision	Recall	F2-Score
Simple AE	0.96	0.32	1.0	0.7
Deep AE	0.94	0.23	1.0	0.6
VAE	0.89	0.12	0.72	0.35
OC-SVM	0.95	0.27	0.87	0.6
IForest	0.95	0.28	0.98	0.66

Table 1: Metrics comparison between different models

As we can see, the accuracy of five models are quite close; all five precisions are not high which implies all of models derived a large number of FP (predict the normals as attacks).

Consider both recalls and F2-Scores as our priority, the simple AE outperforms the other models and eventually becomes the optimal model. However, Isolation Forest as the traditional methods is fairly outstanding among these models with regard to its fast modelling running speed although its recall and F2-Score are a bit lower than the simple AE's.

4.2 Model explanation

Shapley Additive exPlanations (SHAP) is used as the explainability framework for the interpretation of the results from above models. SHAP provides the explanation of the prediction $\hat{x}_i$ of an instance x_i by computing the contribution of each feature to the prediction. That is to say, after we obtained the predicted value $\hat{x}_i$, with the help of SHAP, we can know how the predicted value comes from those features and what proportion of each feature contributes to the predicated value.

We only use SHAP to explain the optimal model: simple AE in this part. Figure 13 shows how the features contribute to a single prediction.

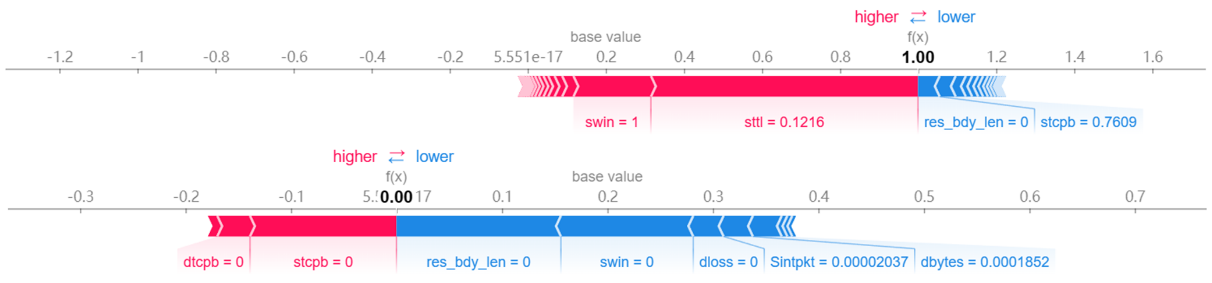

Figure 13: The SHAP explanation on one single prediction (two examples).

In Figure 13, there are two different cases with one prediction equal to 1 (the top case) and the other equal to 0 (the bottom case). As for the top case, red features dominated by "sttl" and "swin" are pushing the predicted value to the right along the axis while blue features, such as "res_bdy_len" and “stcpb” are pushing in the opposite direction. In other words, the red features are positively influencing the prediction while the blue features are negatively influencing it and there exists offset between these features and they finally contribute to a prediction of 1, namely an attack. Similarly in the bottom case, this time these features finally contributes to a prediction of 0, namely a normal one. In addition, we can also find the strength of explainability of each feature on certain prediction. Following these two cases, feature “sttl”=0.1216 has the strongest positive influence for the top case while “stcpb”=0, “res_bdy_len”=0 and “swin”=0 has roughly the same influence no matter positive or negative (Note the feature values are already Min-Max scaled).

Figure 14 shows the overall interpretability of features on 100 predictions. As for the vast majority of cases in which predictions are 0, the most prominent negative feature and its corresponding value is “res_bdy_len”=0 and the most prominent positive feature and its corresponding value is “swin”=0. The exact meanings of these two features are the lower (close or equal to zero) source TCP window advertisement value (“swin”) is, the more likely a record is going to be anomalous since the feature’s influence is positive whereas the lower (close or equal to zero) actual uncompressed content size of the data transferred from the server’s http service (“res_bdy_len”) is, the more likely a record is going to be normal since its influence is negative. Note that these features and their corresponding values somewhere have the opposite effect on the prediction for the possible reason that the predictions are affected by the joint effect from dozens of features, so one feature which has negative impact in one record this time may have positive impact in another record next time.

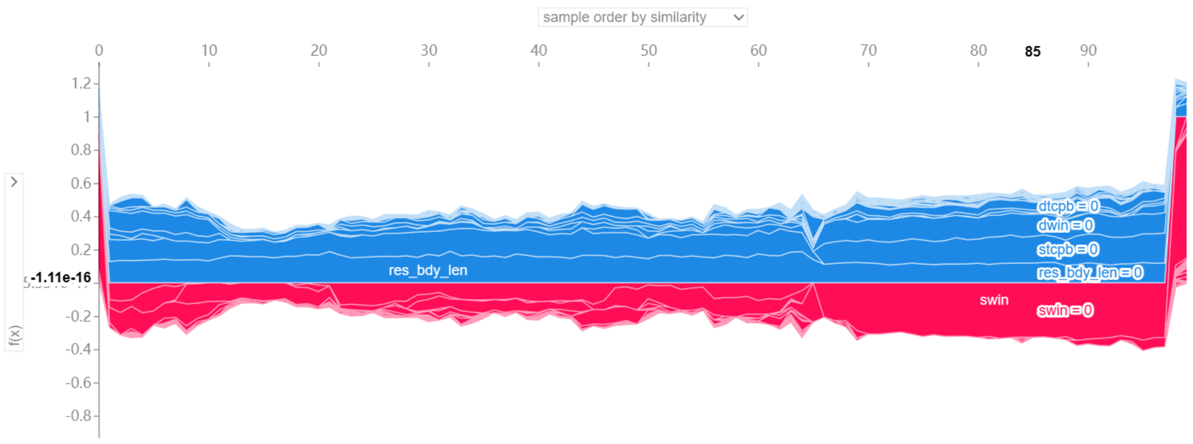

Figure 14: The SHAP explanation on a bunch of predictions.

Figure 15 is a summary bar plot which demonstrates top ten features with highest absolute explanation power regardless of the direction (negative or positive). We can tell that feature “swin” is the most powerful explainer among these features with an absolute SHAP value around 0.19 which means source TCP window advertisement value is the most significant factor in predicting whether a record is normal or anomalous.

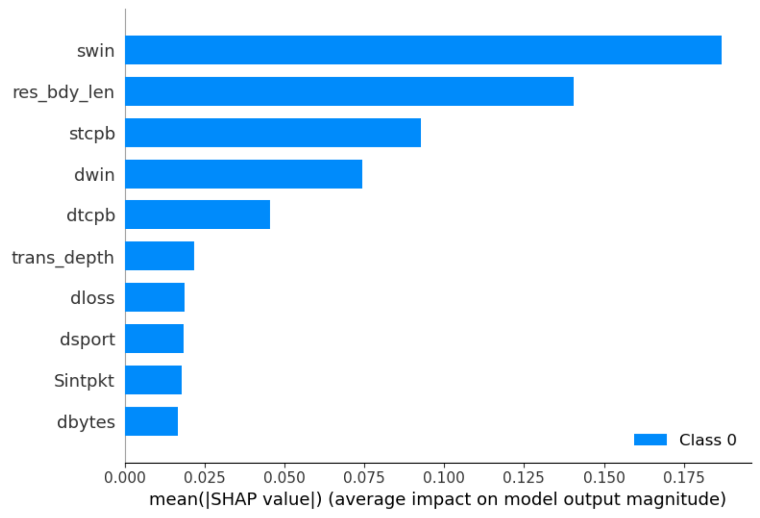

Figure 15: Top ten explainable features with their absolute SHAP values.

5 Conclusions

We performed anomaly detection approaches on UNSW-NB15_1 using simple AE, deep AE, VAE, OC-SVM and Isolation Forest. By comparing models' metrics, especially their recalls, we obtained the simple AE as the optimal model with accuracy of 0.96, precision of 0.32, recall of nearly 1.0 and F2-Score of 0.7 based on predictions of testing set. The low precision results from the high number of false positive but this is tolerable since misjudging a normal record as anomalous is less risky than misjudging an anomalous one as normal. The significantly high recall demonstrates the exploitability of using AEs for anomaly detection in the field of cyber security, e.g. in this case detecting anomalous records. With the assistance of SHAP, we successfully interpreted and explained the results from the simple AE. Some of the features are strongly useful while some are not. The most influential feature is the source TCP window advertisement value regarding the absolute SHAP value.

References

[1] S. Alla and S. K. Adari. Beginning anomaly detection using python-based deep learning: With keras and pytorch. *Beginning Anomaly Detection Using Python-Based Deep Learning*, 2019.

[2] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach. Explaining anomalies detected by autoencoders using shap, 2020.

[3] D. P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[4] U. Michelucci. An introduction to autoencoders, 2022.

[5] C. Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[6] N. Moustafa and J. Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6, 2015.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

Appendices

A Code

The whole Jupyter notebook code can be found at
https://github.com/Yancy0517/autoendoers-for-anomaly-detection.git

B Dataset

The dataset used in this project can be found at
https://cloudstor.aarnet.edu.au/plus/index.php/s/2DhnLGDdEECo4ys?path=