# KDD CUP Dataset Analysis

## Cybercrime and Fraud Detection: Final Project Group Ullman

**Francesco Capo** **Jesper Oedegaard** **Chow Yan Pui Wendy** **Mya Tillak** **Matteo Staglioli**

**270341** **CSO00375** **E00177** **E04272** **259181**

**Abstract:** ***In today's digital age, network security is of paramount importance. Detecting network intrusions accurately is crucial for safeguarding sensitive data and ensuring the integrity of information systems. This project focuses on developing a robust machine learning-based intrusion detection system using the KDD Cup 1999 dataset. The dataset comprises various network intrusion types, including denial-of-service (DoS) attacks, unauthorized access (R2L), remote-to-local (U2R) attacks, and probing attacks. We employ advanced data cleaning techniques and sophisticated machine learning algorithms, such as K-Nearest Neighbors (KNN), XGBoost, and Neural Networks, to classify and detect these intrusions. Our goal is to enhance the accuracy and efficiency of intrusion detection systems, contributing to a more secure network environment.***

## I. Introduction

### *KDD cup*

The KDD Cup 1999 dataset is a benchmark dataset widely used for developing and testing machine learning models for network intrusion detection. The dataset was prepared for the Third International Knowledge Discovery and Data Mining Tools Competition and tries to simulate a military network environment in which various kinds of network intrusions happen. A record in the dataset corresponds to a flow passing through an instance of the network, each entry representing a network connection from two IP addresses taking place over a series of network packets. The dataset contains 41 features in each record: 40 are encapsulated features, while one is the label. A wide range of features encompasses these features; some are essential features of individual TCP connections, some are content features of each connection and traffic features that describe a two-second time window statistically.

### *Features*

Basic features contain information like how long the connection was for, what protocol it was, what service was being accessed on the destination, and what the connection's status was. Content features capture specific behaviors during the connection, such as those brought out in the number of failed login attempts or shell prompts invoked. It calculates the traffic features over the small time window, i.e., the number of incoming connections to the same host in the past two seconds.

Some specific examples of these features are described by the number of seconds since the connection was created, the type of protocol in the connection, the type of network service on the destination, the connection status, the number of bytes transferred from the source IP address, and the number of bytes transferred to the destination IP address. For instance, the count feature describes the number of connections to the same host as the current connection in the previous two seconds.

## *Goals*

The main goal of this project is to design an effective intrusion detection system based on machine learning methods. These generally include the following significant steps: data cleaning to ensure that data are accessible from missing values and duplicated data; data exploration to fathom the distribution of data points and relationships between them; preprocessing data by encoding categorical variables, scaling numerical features, and splitting the data into training and testing datasets; developing models to train numerous machine learning models for classifying network activity into five categories; and evaluating models for the performance of developed models using appropriate checking criteria. This systematic approach is designed to develop a reliable and effective intrusion detection system that may adequately identify and categorize network intrusions.

The KDD Cup 1999 dataset includes labels for each record that indicate whether the network activity is normal or an attack. These attacks are further categorized into five broader categories:

**Benign:** normal network activity without any malicious intent.

**Denial of Service (dos):** attacks, such as *smurf* and *udpstorm*, aim to overwhelm the network or server resources, rendering them unavailable to legitimate users.

**Remote to Local (r2l):** attacks, like *ftp_write* and *guess_passwd*, involve an attacker gaining unauthorized access to a local machine from a remote location, often by exploiting weaknesses in authentication mechanisms.

**User to Root (u2r):** attacks, including *buffer_overflow* and *rootkit*, are more severe, as they allow an attacker with limited access to escalate their privileges to gain root or administrative access.

**Probe:** attacks, such as *satan* and *ipsweep*, involve scanning the network to gather information about possible vulnerabilities that can be exploited for future attacks.

## II. Data Cleaning

Data cleaning is a critical step in any data analysis project. The goal is to prepare the raw data for analysis by removing inaccuracies, addressing inconsistencies, and transforming it into a format suitable for machine learning models. In this project, the data cleaning process involved several steps: identifying and handling missing values, removing unnecessary columns, encoding categorical variables, standardizing numerical data, addressing class imbalance, and performing feature selection.

### 1. Identifying and Handling Missing Values and Duplicates

Missing values can severely impact the performance of machine learning models. They can lead to biased estimations, reducing the accuracy of the model. Therefore, the first step in the data cleaning process was to identify any missing values in the dataset and decide on an appropriate strategy to handle them.

To detect missing values, we utilized the *isnull* and *sum* functions from pandas. These functions help us count the number of missing values in each column.

In our dataset, we found that there were no missing values, which simplified the cleaning process.

Since there was no unique identifier, the presence of duplicates couldn't be estimated.

### 2. Removing Unnecessary Columns

Not all columns in a dataset contribute to the analysis or model building. Some columns may contain redundant or irrelevant information that can be removed. In our dataset, we identified the *num_outbound_cmds* column as candidate for removal because it contained only zero values across all records.

Columns with constant values do not provide any useful information for differentiating between classes in a machine learning model. Therefore, removing such columns can help in reducing the dimensionality of the dataset and improving the model's performance (Han et al., 2011).

### 3. Encoding Categorical Variables

Machine learning algorithms require numerical input to perform computations. Therefore, categorical variables need to be converted into a numerical format. This process is known as encoding. There are various methods for encoding categorical variables, including one-hot encoding and label encoding. In our case, we used label encoding for simplicity and efficiency (Han et al., 2011).

Label encoding assigns a unique integer to each category in the categorical variable. We applied label encoding to the **"protocol_type"**, **"service"**, **"flag"** and **"label"** columns in our dataset.

### 4. Standardising the Data

Standardisation is a crucial step in data preprocessing, especially for algorithms that rely on distance calculations, such as k-nearest neighbours (K-NN) and support vector machines (SVM). Standardisation transforms the data to have a mean of zero and a standard deviation of one, ensuring that all features contribute equally to the analysis (Buczak & Guven, 2015).

We applied standardisation to all numerical columns in the dataset. The **“StandardScaler”** from scikit-learn was used for this purpose.

The data cleaning ensured, through a systematic approach, that the dataset was free from inconsistencies and was in a suitable format for machine learning models. By addressing missing values, removing unnecessary columns, encoding categorical variables, standardising numerical data, handling class imbalance, and selecting key features, we enhanced the quality of the dataset.

## 5. Summary

This comprehensive data cleaning process laid a strong foundation for building robust and accurate machine learning models, ultimately contributing to the success of our project. The careful preparation of the data ensures that the models trained on this data are reliable and capable of making accurate predictions, thereby meeting the project's objectives effectively.

## III. Exploratory Data Analysis

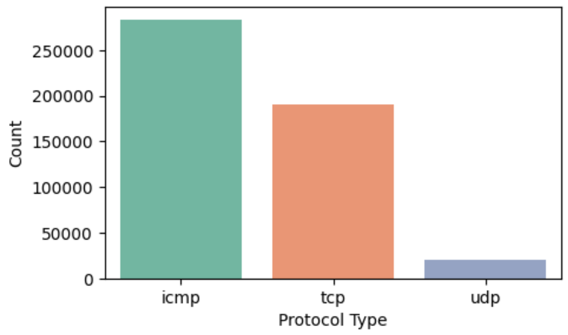
In the domain of network intrusion analysis, Exploratory Data Analysis (EDA) is a vital step when developing a machine learning-based intrusion detection module using the KDD Cup 1999 Data.

EDA helps achieve this goal by understanding the data's structure, patterns, and relationships. It involves detecting anomalies such as outliers and missing values, generating hypotheses about the data, informing data cleaning decisions, and choosing appropriate models.
Identifying and addressing data quality issues, analysing correlations between features, and performing feature engineering are crucial steps in ensuring a comprehensive understanding of the dataset. This exploration assists in developing an effective intrusion detection classifier that accurately categorises network activities into the predefined classes.

### *Explanation of the categorical variables*

*Sorting Protocol*

The "protocol_type" variable in the dataset refers to the network protocol used, for example, TCP, UDP and ICMP. Knowing the distribution of protocol types helps understand the dataset's composition, which is essential in building a robust intrusion detection model as certain attacks may be more prevalent in specific protocols .

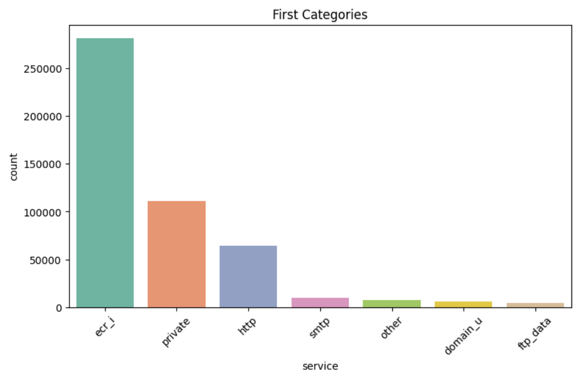


The above bar chart creates a visual count of each protocol type in the training data. From the graph it is evident that *ICMP* has the highest count, *TCP* still has a significant count but less than *ICMP* and *UDP* has the lowest count. This distribution indicates an imbalance that could influence the detection model. For instance, many DoS attacks might utilise *ICMP* or *TCP* protocols (Khraisat et al., 2019). Therefore, this initial analysis helps guide subsequent pre-processing and feature engineering steps, ensuring that the model accurately captures and learns from the patterns associated with each protocol type.

*Service*

The 'service' variable represents the network service on the destination, whether it is HTTP, FTP, or so on. These services can indicate the kind of communication happening over the network, and is crucial for understanding whether the activity is a potential security threat or patterns of ordinary behaviour.

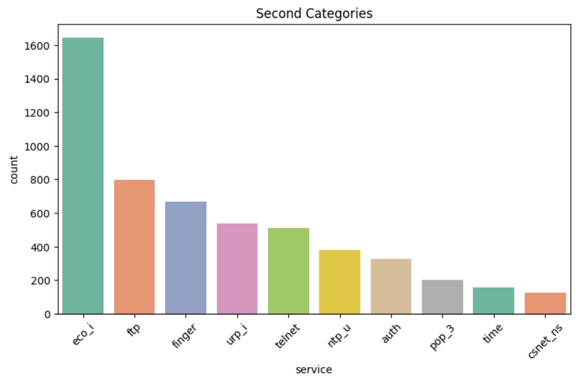
Due to the large dataset of service variables, for ease of visualisation the data was split into three groups based on their frequency – large, medium and small. The first subplot created depicts services with high frequency and uses the 'order' parameter to ensure the services are plotted in descending order of their frequencies.



This plot represents services with more than 2,100 occurrences, with the network service *‘ecr_i’* occurring significantly higher than the rest of the network service variables. ‘*ecr_1*’ is a service related to the TCP/IP protocol that manages network congestion (Lim, 2020). Frequent occurrences of *‘ecr_i’* might indicate routine network operations involving congestion management, or on the other hand, the unusual patterns may suggest issues or targeted attacks (Lim, 2020). Hence, the importance of the EDA to ensure that datasets with imbalances and anomalies are adequately represented in the model training process.

This is then followed by the network service “*private*”, which concerns internal or proprietary network services that are not publicly defined or standardised (Ayub, et al., 2022). High occurrences of these services can indicate strong use of internal applications and may cause concern for potential internal attacks or exploitation of proprietary protocols. The third most frequent network service is “*http*”, this service is used for transmitting web pages and web-based applications and therefore is a common target for web-based attacks such as SQL injection, cross-site scripting (XSS) and distributed denial-of-service (DDoS) attacks (Ramamoorthy, et al., 2023). Close monitoring of these services are vital as different services are associated with specific vulnerabilities and attack vectors. For instance, web-based attacks, such as SQL injection and cross-site scripting (XSS), are more likely to target HTTP services due to the nature of web traffic and applications (Ramamoorthy, et al., 2023 and Naresh, et al., 2022). Similarly, FTP services are particularly vulnerable to brute-force attacks because they often involve authentication processes that can be exploited by attackers repeatedly trying different password combinations (Hamza & Al-Janabi, 2024).

The remaining four networks “*smtp*”, “*other*”, “*domain_u*” and “*ftp_data*” were distributed less frequently in comparison to the first three mentioned, however they still occur more frequently than the services depicted in the bar plots measuring the second and third categories. This suggests that these services are frequently targeted and should be well-represented in the model training process to ensure robustness against common attack vectors (Andelic 7 Segota, 2024).



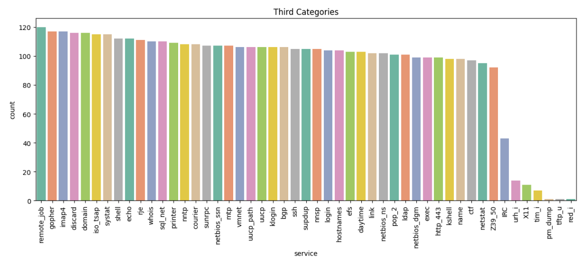
The second part of Categories bar plot represents medium frequency services with occurrences between 120 and 2100. The highest frequent services of this category include:

- *‘Eco_i’* service network measures round-trip times in IP networks and verify that network paths are operational. These services can be misused for network reconnaissance or as part of amplification attacks in denial-of-service (DoS) scenarios. Proper monitoring and securing of this service are essential to prevent exploitation (Vishnu & Praveen, 2022).

- *'FTP'* is a standard protocol used to transfer files between a client and a server over a network, update web servers and manage website content. This server is extremely vulnerable to interception and eavesdropping making it likely targeted for brute-force attacks and unauthorised data access.

  - 'Finger' service protocol provides information about users on an isolated system, such as login name, real name and terminal name. While this service can be used to monitor user activity and manage system resources, the finger service is a susceptible security risk as it exposes sensitive user data that can be used for reconnaissance or social engineering attacks (Vishnu & Praveen, 2022).

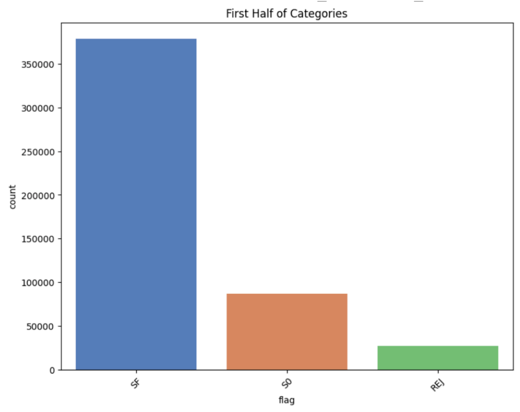
Understanding the network services and their occurrences in the dataset assist in distinguishing between typical network operations and potential areas of concern for security. Whilst the use of these systems is vital for maintaining network stability and performance, their security considerations are evident, especially for services such as *'FTP'* and *'Telnet'* that use plain text protocols.



This third plot complements the previous two by focusing on infrequent services, with 120 or fewer occurrences. This long tail of services with very low frequency, indicating a diverse but sparse set of service types (Addai, et al., 2023). The sparse distribution might lead to potential underrepresentation of these services in the model. Despite their infrequent services, the networks services in this category are still susceptible to be targeted by specific attacks (Addai, et al., 2023).

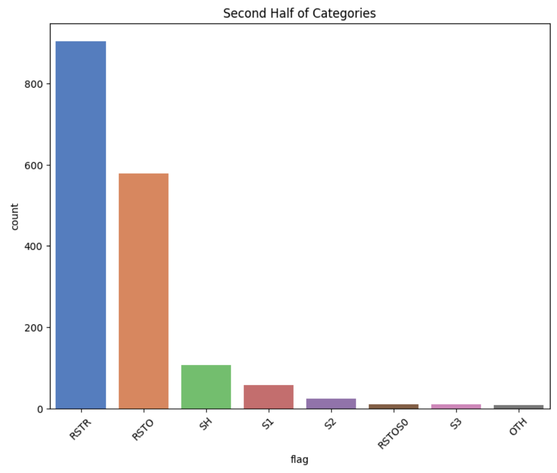
*Flag*

The 'flag' variable represents the status of the connection and contains both frequent and rare flags, which can provide insights into normal and anomalous network behaviours. The code calculates the frequency of each *'flag'* category and then splits the categories in two groups, *'flag_large'* for categories with more than 25,000 occurrences, and *'flag_small'* for those with 25,000 or fewer occurrences.



The first subplot presented above, shows that the *'SF'* (successful connections) flag is the most common connection with 378, 440 connections. This is a normal functioning network as most connections are successfully completed (Ko et al., 2021). The second flag *'S0'* (connections started but were not completed) has a noticeable drop in status connection with an approximate count of 58, 300, and is followed closer by 'REJ' (rejected connections), with 14, 993 connections. The *'S0'* and *'REJ'* flags indicate a sizable number of incomplete or rejected connections (Ko et al., 2021). Whilst some level of rejection is normal, it also may ultimately be due to network-level security measures or firewalls blocking unauthorised access (e.g. DoS attacks) (Tariq, et al., 2023).

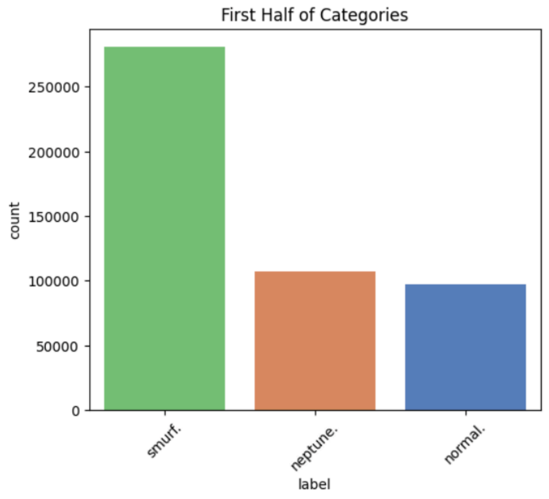
For example, flags such as *'S0'* could be due to network reconnaissance activities. Therefore, it is important to ensure that monitoring of spikes in this flag type is a part of the machine learning-based intrusion detection model (Addai et al., 2023). As previously discussed in relation to the aforementioned variables, there is a noticeable imbalance in the distribution connection states. This suggests that despite most network traffic connecting successfully, there are notable rejected attempts and connections without replies.



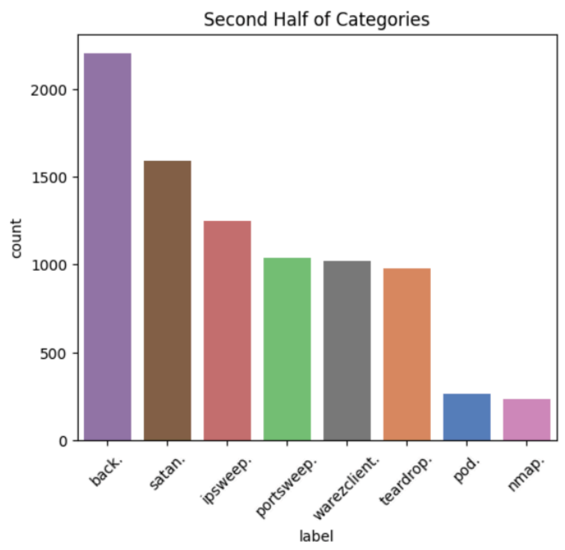
The low-frequency flags represented in the second bar plot, include *'RSTO'*, *'RSTR'*, and *'SH'*, whilst are less common, highlight rare states that may suggest specific attack vectors or anomalies (Dastres & Soori, et al., 2021). Therefore, a high number of reset connections of 'RSTO' and 'RSTR' may indicate attempts of port scanning or attempts to evade detection by terminating connections prematurely (Abdulganiyu, et al., 2023). Further, flags such as *'SH'* (half-opened connections) could indicate SYN flood attacks, which are a type of DoS attack that exhausts server resources by initiating many half-open connections.

## *Label*

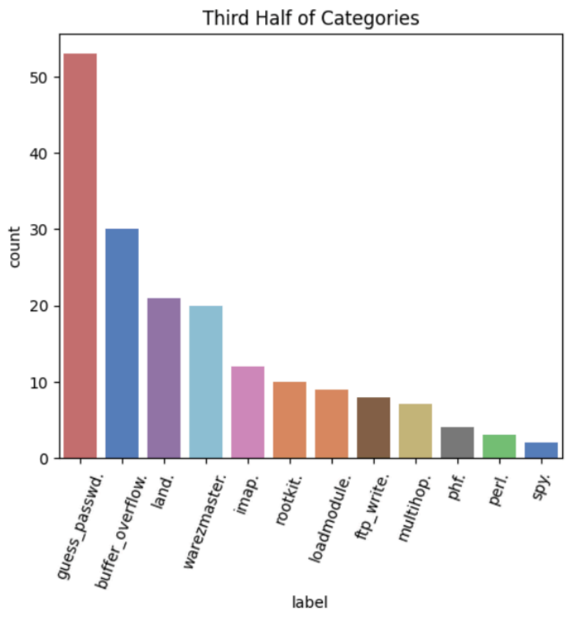
To gain insights into the distribution of different labels in the dataset, we performed a detailed exploratory data analysis of the label variable. The label variable categorizes each network connection or activity, which is crucial for understanding the dataset composition and preparing for model training. The analysis involved counting the occurrences of each label and dividing them into three categories: highly frequent labels, moderately frequent labels, and rare labels.



- **First Half of Categories Labels**: The first subplot displays labels with more than 50,000 occurrences. These labels are dominant in the dataset and could potentially bias.



- **Second Half of Categories Labels**: The second subplot shows labels with frequencies between 200 and 50,000. These labels provide a balanced representation and are crucial for model generalisation.



- **Third Half of Categories Labels**: The third subplot focuses on labels with 200 or less occurrences. The graph provides valuable insights into the distribution of rare attack types in the dataset that are important to understand in order to build a comprehensive intrusion detection system.

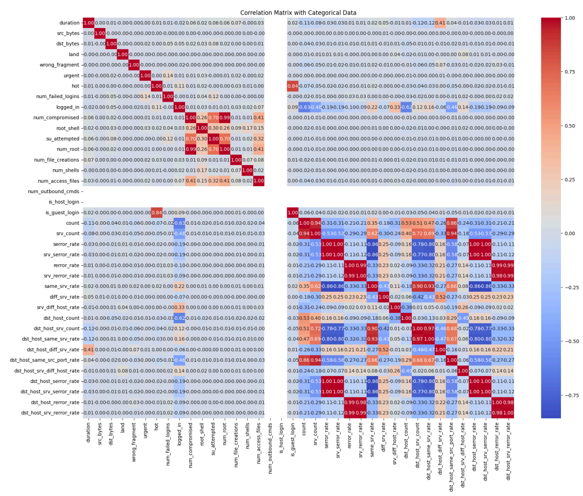
The mean occurrence of 'labels' (14.75), the high standard deviation (15.63), and the presence of both extremely frequent and rare labels indicate a significant class imbalance, leading to a biased model that performs well for frequent labels but poorly on rare ones. This bias model is ineffective in detecting certain types of intrusions, namely less frequent occurrences and therefore overlooks potential threats.

It is crucial to understand the distribution of this data in order to build a robust intrusion detection system.

In relation to imbalance datasets, it is vital to focus on precision, recall and F1-score, especially on the minority variables (Kanstren, 2020). For example, the 'guess_passwd.' category would benefit from precision and recall in ensuring that the model correctly identified the actual occurrences of this attack type without many false positives or negatives (Brownlee, 2020).

## ***Numeric Variables***

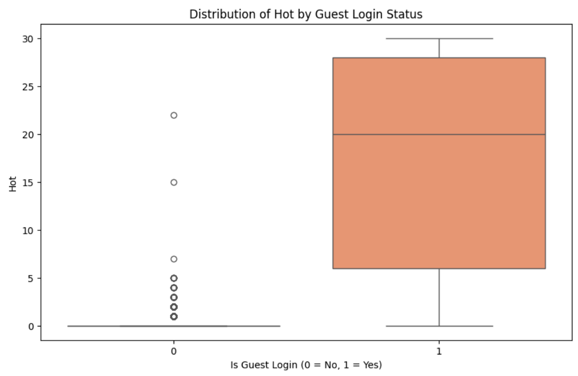
Now the analysis focus shifts from the categorical variables to numeric ones.
In order to better understand the latters, a correlation matrix is plotted.



When analysing the above correlation matrix some difficulties arise. These were given mainly by two factors, firstly the numerous variables present in the graph that made impossible the analysis of all of the interactions. Indeed we opt to plot only the one with high correlation, at this point comes the second problem. Since the majority of the variables present have has main value either 0 or in some cases 1 interpreting the result was very challenging.

*Hot and is_guest _login*

- *Hot:* represents the number of 'hot' indicators within a network connection. These indicators represent suspicious or unusual activities that might signal an intrusion or attack.
- *Is_guest _login: represents* whether a network connection was made using a guest login. 1 indicates that the login was guest while 0 that was not a guest log in.

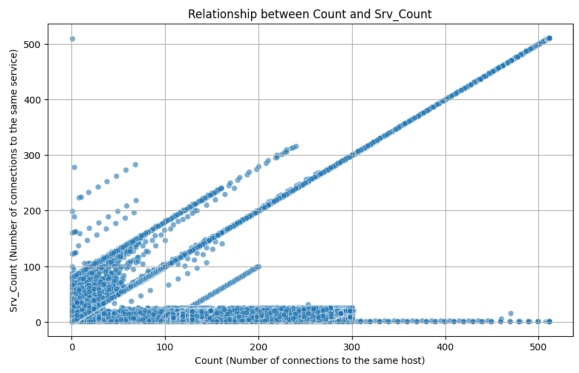


Now that the variables are more clear, it can be understood that the majority of the logins was not a guest login.

*Srv_count and Count*

- *Count*: Refers to the number of connections to the same host as the current connection in the past two seconds. It includes connections that were initiated or received by the host, and can be an indicator of the frequency or volume of traffic directed at a host within a very short timeframe.

- *Srv_count*: refers to the number of connections to the same service as the current connection in the past two seconds. Unlike count, which is focused on connections per host, srv_count is specific to the service type, providing insight into the concentrated use or potential targeting of specific services.



The scatter plot illustrates a positive correlation between the two variables. A prominent diagonal cluster indicates that higher connections to a host correspond to higher connections to a service. Clusters at lower values suggest many connections involve fewer than 100 interactions. The plot helps in distinguishing normal network traffic from anomalies, aiding in intrusion detection.

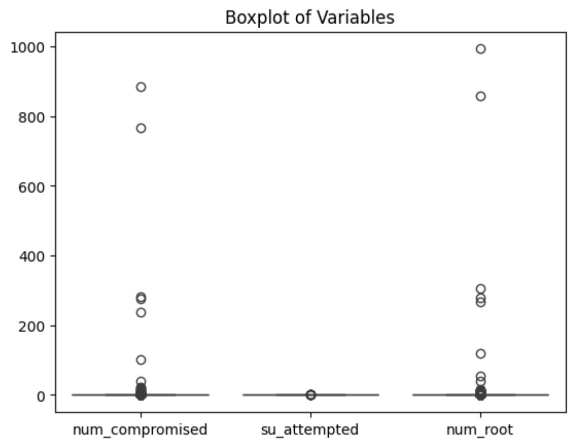
*Num_compromised, su_attempted, and num_root*

These three variables highly correlated all three to each other, so they were plotted together.

- *Num_compromised*: This variable indicates the number of compromised conditions on the network.

- *Su_attempted*: This variable records the number of 'su' command attempts, which is used to switch user identities.

- *Num_root*: This variable measures the number of 'root' accesses or actions performed.



The boxplot shows that the variables "num_compromised," "su_attempted," and "num_root" have their main distribution close to zero, indicating these events are generally rare. However, the presence of numerous outliers suggests occasional significant spikes in compromised conditions, 'su' attempts, and root access, highlighting instances of severe security breaches.

## IV. Classification Models

### Preprocessing data for the classification model

Before processing the data and classifying them, datasets have been cleaned and preprocessed in order to maximise the accuracy of the classifier. This ensures the data format is suitable and analysable by the machine learning algorithm. Firstly, we compare the data labels present in the training data and the corrected data by printing out all the unique data labels from both datasets. Secondly, we count the frequency of data under each label to give a holistic distribution of the cyberattacks, with the most attacks being *smurf* and *neptune*. Thirdly, we filter out the data with labels that are not present in the training data. Fourthly, the dictionary *Label Mapping* is used to rename the labels into the categories we specified for cyberattacks. Fifthly, the dictionary then replaces the labels in the training and corrected dataset into the new categories. The data labels are further encoded into numerical values. After that, the standard scaler is used to standardise values of the numeric columns. Finally, the data is split into training and testing X and Y data.

### *Defining terms*

Before explaining the rationales and definitions of classification models, interpreting the insights of the models requires understanding of the terms in the classification report, confusion matrix, and accuracy score.

The classification report shows the performance on the test set. These terms are defined to assist the performance of the model:

- *Precision score* identifies the true positive predictions to the total number of positive predictions.

- *Recall score* shows the ratio of true positive predictions to the total number of actual positive instances.

- *F1-score* combines the precision and recall score, which shows the general performance of the model.

- *Support* represents the number of instances in each class in the test set.

- *Confusion matrix* is a detailed breakdown with the rows represent true classes and columns represent predicted classes. Diagonals represent true positives and off diagonal are false positives and negatives

- Accuracy score represents the percentage of instances that are correctly identified. It is the most intuitive metrics to assess the effectiveness of the model

### *Classification models and insights*

1. *KNN*

   a. Rationale

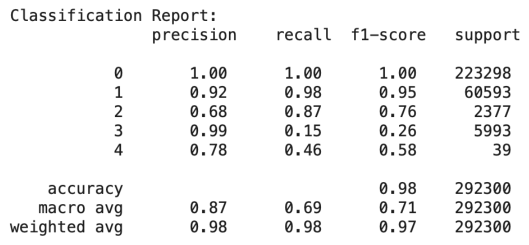
KNN stands for K-nearest neighbour, which is a supervised learning classifier in machine learning that groups similar individual data points together. In our case, we aim to fit the class labels in the training data in X_train and labels in Y_train and predict the labels in the X_ test. KNN is especially suitable for intrusion detection due to its simpler nature of setting fewer parameters, only the value of neighbours (K) and the distance metric. It is a lazy learning algorithm. KNN is a lazy learner and does not pick up discriminatory measures but memorises the dataset instead. Hence there is no learning time needed for KNN.

### b. Methodology

The first step is to initialise the KNN classifier by setting the random state at 42 to ensure uniform generation of random numbers each round, and by calling the KNeighboursClassifier() function. Random state 42 gives the same output when making the data split as it ensures the same numbers are generated when users run the examples. Secondly, the parameter grid is classified by setting the targeted number of neighbours, weights and metrics. Weights are uniform and the Euclidean distance is chosen as the hyperparameters. Euclidean distance effectively groups similar data points together from calculations of various dimensions. Thirdly, a grid search would be performed to find the best matched parameter for the kNN model. Fourthly, the model is evaluated by using classification report, confusion matrix and accuracy score. Fifthly, the best parameters would be retrieved. Sixthly, the target variable is encoded. Precision, recall, F-score, and support for each class is calculated. And lastly, a data frame is created to hold the results.

### c. Results

The best parameters are, namely *euclidean* for distance, *7* for n_neighbours, and *uniform* weights. It means all 7 neighbours contribute equally to the classification using euclidean distance.



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

           0       1.00      1.00      1.00    223298
           1       0.92      0.98      0.95     60593
           2       0.68      0.87      0.76      2377
           3       0.99      0.15      0.26      5993
           4       0.78      0.46      0.58        39

    accuracy                           0.98    292300
   macro avg       0.87      0.69      0.71    292300
weighted avg       0.98      0.98      0.97    292300
```

From the classification report, it shows that:

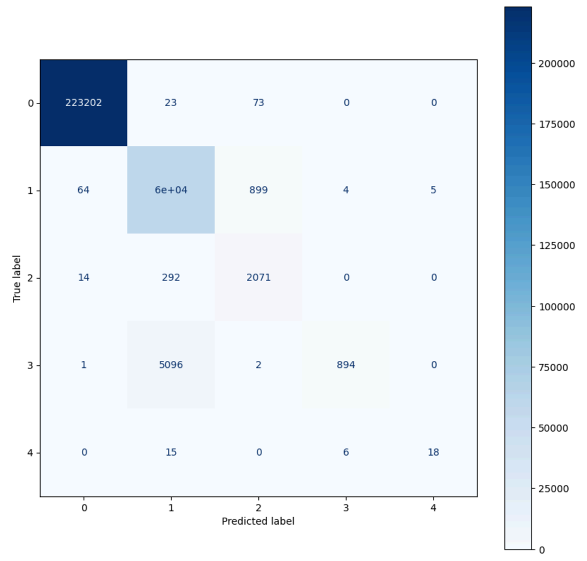
- *Class 0:* The model has an outstanding performance with an exceptionally high precision (1.00) and recall (1.00).

- *Class 1*: The model has a generally good performance with a high precision (0.92) and recall (0.98)

- *Class 2*: The model has a mediocre performance with precision (0.68) and recall (0.87)

- *Class 3:* The model has high precision (0.99) but very low recall (0.15), It can identify correctly but many numbers are missing

- *Class 4:* The model has moderate precision (0.78) and recall (0.46), which it has difficulty identifying precisely.



With an overall accuracy of the model being 0.98, it correctly classifies 98% of the instances in the test set. In summary, the model performs very well in class 0 and 1, but is less accurate with class 3 and 4.

## 2. *Random Forest*

### a. Rationale

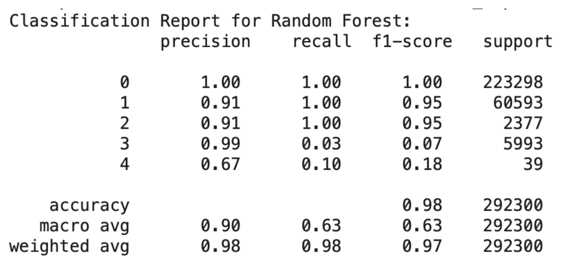
Random forest is an ensemble decision model that combines the decisions of multiple decision trees. It randomly chooses observations and takes the majority results of the decision tree, which makes it highly accurate and relatively unbiased. When it acts as a classifier, the mode of the classes of the decision trees will be taken. The questions in the decision tree act as a means of splitting the data and eventually reaches a final conclusion. Random forest prevents the risks of overfitting data, and has a short training time as well. It can also handle large amounts of data in a database precisely and is good at estimating missing data.

### b. Methodology

Firstly, grid search parameters are set up, which decides on the number of trees, maximum depth of tree, minimum numbers of samples to split the node, are determined at the first stage. Then grid search cross validation would be carried out. Predict() is used to make predictions on test data, and the remaining steps are similar to the KNN.

### c. Results

The ideal maximum depth of the trees is 20. Minimum number of samples required to be at a leaf node is 1. Minimum number of samples required to split a node is 5 and the total number of trees in the forest is 100.



```
Classification Report for Random Forest:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00    223298
           1       0.91      1.00      0.95     60593
           2       0.91      1.00      0.95      2377
           3       0.99      0.03      0.07      5993
           4       0.67      0.10      0.18        39

    accuracy                           0.98    292300
   macro avg       0.90      0.63      0.63    292300
weighted avg       0.98      0.98      0.97    292300
```

The classification report shows that:

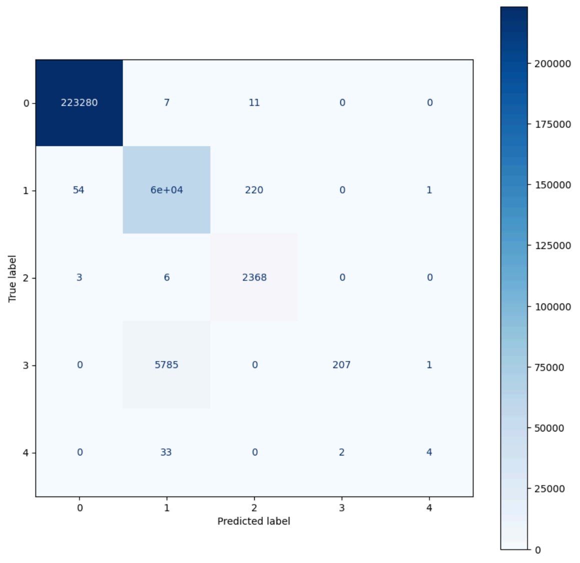
- Class 0: It has an outstanding performance in precision (1.00) and recall (1.00).

- Class 1: The model has high precision (0.91) and recall (1.00).

- Class 2: The model has high precision (0.91) and recall (1.00)-

- Class 3: The model has high precision (0.99) but very low recall (0.03), and similarly it identifies accurately but misses lots of numbers.

- Class 4: The model has medium precision (0.67) and low recall (0.10), hence the model is inaccurate in class 4.



With an excellent accuracy of 0.98, 98% of instances are correctly identified. Recall scores and precision scores indicate a terrific performance in class 0 and struggles in class 4 the most. Compared to the Knn model, it operates slightly less accurately, but performs better in class 1 and 2, and both struggled with class 3 and 4.

### 3. *Neural Network*

#### a. Rationale

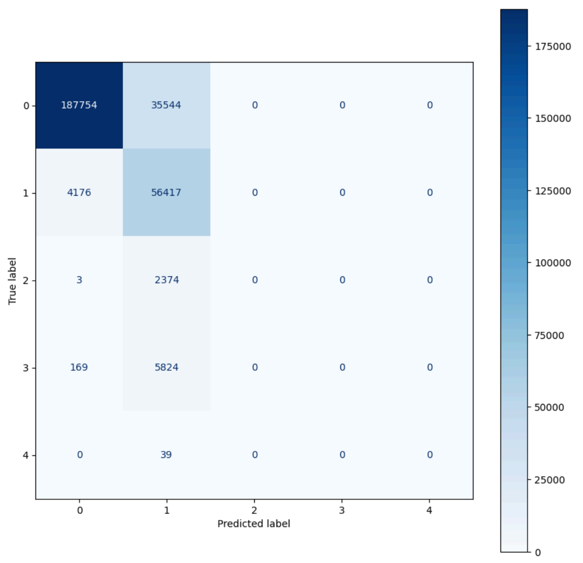
Neural network performs similarly to the human brain. When the output of a node exceeds the threshold and weight allocated, then it moves onto the next layer of the network.

#### b. Methodology

Firstly, we set the random seed and we choose to use a sequential model, with the first layer having 12 nodes and a rectified linear unit, second with 8 nodes and third layer with 1 node. The model is then compiled, trained and evaluated.

#### c. Results

50 epochs means the full training set has been run thoroughly 50 times. The decrease in lose values over the tries shows that the model is learning and improving every run.



```
Test Accuracy: 0.8353438377380371
```

The accuracy values are very consistent, staying at around 97-98%. The test accuracy of 0.8423 shows a  classification accuracy of around 84.23% on the test data. The high training accuracy and low test accuracy may hint on overfitting issues.

## V. Conclusions

In our analysis, the **K-Nearest Neighbour (KNN)** model showed an accuracy of *0.98*, with excellent performance for Class 0 and Class 1 but struggled with Classes 3 and 4.

The **Random Forest** classifier demonstrated high overall accuracy, *0.98,* particularly excelling in Classes 1 and 2, though it faced similar challenges with Classes 3 and 4 as KNN.

The **Neural Network** displayed strong learning capabilities with high training accuracy but indicated potential overfitting due to a lower test accuracy, 0.83. Overall, while each model has strengths, improvements are needed for better handling of specific classes and enhancing generalisation.

## *Reference*

- Abdulganiyu, O. H., Tchakoucht, T. A., & Saheed, Y. K. (2023). A systematic literature review for network intrusion detection system (IDS). *International Journal of Information Security*, *22*. https://doi.org/10.1007/s10207-023-00682-2
- Addai, P., Freas, R., Elnatan Mesfin Tesfa, Sellers, M., & Tauheed Khan Mohd. (2023). Prevention and Detection of Network Attacks: A Comprehensive Study. *Prevention and Detection of Network Attacks: A Comprehensive Study*, *474*, 56–66. https://doi.org/10.1007/978-3-031-32534-2_5
- Ali Hamza, A., & Urayh Al-Janabi, J. s. (2024). Detecting Brute Force Attacks on SSH and FTP Protocol Using Machine Learning: A Survey. *Journal of Al-Qadisiyah for Computer Science and Mathematics*, *16*(1). https://doi.org/10.29304/jqcsm.2024.16.11432
- Anđelić, N., & Šegota, S. (2024). Enhancing Network Intrusion Detection: A Genetic Programming Symbolic Classifier Approach. *Information*, *15*(3), 154–154. https://doi.org/10.3390/info15030154
- Ayub, M., Lajam, O., Alnajim, A., & Niazi, M. (2022). Use of Machine Learning for Web Denial-of-Service Attacks: A Multivocal Literature Review. *Arabian Journal for Science and Engineering*, *48*. https://doi.org/10.1007/s13369-022-07517-7
- Brownlee, J. (2020, January 2). *How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification*. Machine Learning Mastery. https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/
- Buczak, A. L., & Guven, E. (2015). A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials, 18*(2), 1153-1176.
- Dastres, R. & Soori, M. A Review in Recent Development of Network Threats and Security Measures. *International Journal of Information Sciences and Computer Engineering*, 2021. Ffhal03128076f
- Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.
- Hettich, S., & Bay, S. D. (1999). The UCI KDD Archive [http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html]. Irvine, CA: University of California, Department of Information and Computer Science.
- Kanstrén, T. (2020, October 31). *A Look at Precision, Recall, and F1-Score*. Medium. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec
- Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, *2*(1), 1–22. https://doi.org/10.1186/s42400-019-0038-7
- Ko, H., Rim, K., & Praça, I. (2021). Influence of Features on Accuracy of Anomaly Detection for an Energy Trading System. *Sensors*, *21*(12), 4237–4237. https://doi.org/10.3390/s21124237
- Lim, C. (2020). Improving Congestion Control of TCP for Constrained IoT Networks. *Sensors*, *20*(17), 4774. https://doi.org/10.3390/s20174774
- N/A. (2019, March 5). *Classification: Precision and Recall | Machine Learning Crash Course | Google Developers*. Google Developers. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall
- Naresh Chillur, Patel, A., Patel, S., & Swain, D. (2022). Deep Analysis of Attacks and Vulnerabilities of Web Security. *Lecture Notes in Electrical Engineering*, *936*, 1087–1097. https://doi.org/10.1007/978-981-19-5037-7_78
- Oliynykov, R., Kuznetsov, O., Lemeshko, O., & Radivilova, T. (2022). *Information security technologies in the decentralized distributed networks* (pp. 115–131). Springer.
- Ramamoorthy, J., Damilola Oladimeji, Garland, L., & Liu, Q. (2023). Detection and Classification of Web Application Attacks. *Lecture Notes in Computer Science*, *13926*, 301–312. https://doi.org/10.1007/978-3-031-36822-6_26
- Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review. *Sensors*, *23*(8). MDPI. https://doi.org/10.3390/s23084117
- Vishnu, V., & Praveen, K. (2022). Identifying Key Strategies for Reconnaissance in Cybersecurity. *Studies in Computational Intelligence*, *1007*, 35–47. https://doi.org/10.1007/978-981-16-8012-0_3
- Werener de Vargas, V., Arthur Schneider Aranda, J., dos Santos Costa, R., Ricardo da Silva Pereira, P., & Luis Victoria Barbosa, J. (2022). Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. *Imbalanced Data Preprocessing Techniques for Machine Learning: A Systematic Mapping Study*, *65*(1), 31–57. https://doi.org/10.1007/s10115-022-01772-8