An Experimental Study on the Use of Deep & Machine Learning for N.I.D Systems

Dharmarlou Bowen

Computer & Data Science

School of Computing and Digital Technology

Faculty of Computing, Engineering and the Built Environment

Birmingham City University

Submitted Month 2023

BIRMINGHAM CITY
University

A thesis submitted to Birmingham City University in partial fulfillment of the requirements of the degree of Bachelor of Science.

Abstract

In this project, we aimed to develop a machine learning model for detecting malicious intent in data packets using the "CSE-CIC-IDS2018" dataset. We conducted exploratory data analysis to gain insights into the characteristics and patterns of the network traffic data, and developed and implemented a machine learning model using suitable algorithms and techniques for classification of data packets as malicious or benign. We evaluated the performance of the model using appropriate evaluation metrics and compared it with existing approaches. Our results showed that the fine-tuned LSTM model had the highest accuracy and F1 score, followed by the decision tree model. The Naïve Bayes and logistic regression models performed poorly. We recommend future work to focus on optimizing the decision tree model and exploring other deep learning algorithms for improving the accuracy of the model. Overall, our study provides a valuable contribution to the field of network intrusion detection and could help improve the security of computer networks.

Acknowledgements

My deepest gratitude and unreserved thanks are owed to my Final Year Project advisor, Dr. Mohammed Abdelsamea for his exceptional direction, guidance and support throughout the entire span of this project. A trusted mentor whose value is beyond measure; he provided me with astute commentary, specialist recommendations as well as unwavering composure whilst guiding me through intricacies inherent in conducting research work such as mine. Despite encountering moments where my focus proved elusive or my attentiveness waned on occasion - Doctor Abdelsamea showed continuing commitment towards my project obligations by going above board in ensuring that I remained focused enough to make headway every time we met.

Not only does Dr. Abdelsamea hold the position of my project advisor, but he also earns my gratitude for his contributions to the enhancement of my knowledge around Deep Neural Networks due to him being my tutor for the Deep Neural Networks module. His profound knowledge and expertise in the field of deep learning have been immensely beneficial in shaping the direction and methodology of my project. His willingness to share his extensive knowledge and provide guidance on various aspects of deep neural networks beyond the scope of my project has been truly invaluable.

Table of Contents

Glossary

A list of symbols and abbreviations with expansions of any contractions.

NIDS	Network Intrusion Detection System
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
ML	Machine Learning
DDos	Distributed Denial of Service

List of Figures

List of Tables

1 Introduction

The purpose of this particular undertaking concerns the point at which Network Security and Machine Learning converge. Specifically, the project utilizes the "CSE-CIC-IDS2018" network traffic dataset from the University of New Brunswick, which showcases various formats of DDoS (Distributed Denial of Service) attacks. The goal of the project is to develop a functioning machine learning model that can accurately predict whether incoming data packets exhibit malicious intent or not.

1.1 Problem Definition

The increasing prevalence and sophistication of cyber-attacks pose significant challenges to network security. Traditional methods of detecting and mitigating such attacks are often inadequate due to their reliance on static rules and signatures that are easily bypassed by advanced attacks. Using machine learning is only become an ever more necessary solution to advanced problems faced in the field of network security, given its capability to detect attacks that have never been witnessed before while also being able to adapt and adjust itself according to varying threat landscapes. Nonetheless, a requisite for additional investigation and innovation looms large in this particular domain.

1.2 Scope

The scope of this project is focused on the development of a machine learning model for detecting malicious intent in data packets, using the "CSE-CIC-IDS2018" dataset. The intention of this venture is to fabricate and assess the efficiency of the structure employing conventional mechanisms for mechanized cognition. The undertaking does not entail executing the model in a live Intrusion Detection System (IDS) simulation or implementing it within an operational domain. The reason for this constraint is because of the time limit and resources assigned to the project, along with ethical concerns related to performing actual experiments on live networks.

1.3 Rationale

The proposed project on network security and machine learning hold great weight and use for companies such as Lloyds Banking Group, a leading global financial institution with a strong emphasis on cybersecurity (Lloyds Banking Group, 2022) or any financial institution for that manner. The financial sector is one of the main victims for malicious-threats with especially with the rise of internet banking, attacks such denial of service, which can interrupt any online services, expose user data or even result in reputational scarring of a business. By developing a model for detecting and identifying malicious activity in network traffic, it would provide and elevate to an extent the risk for more advanced attacks.

In addition to the immediate benefits such institutions, the outcomes of this project expands to the broader width of the field of network security and machine learning research. The development of an accurate and efficient machine learning model for network security, specifically for DDoS attack detection, can contribute to the advancement of knowledge and practices in the field of intrusion detection systems.

1.4 Project Aim and Objectives

The overall aim of this project is to develop a machine learning model for detecting malicious intent in data packets. The specific objectives of the project are as follows:

- Collection and pre-processing of the "CSE-CIC-IDS2018" dataset, which includes various formats of DDoS attacks.
- Conduct exploratory data analysis to gain insights into the characteristics and patterns of the network traffic data.
- Develop and implement a machine learning model using suitable algorithms and techniques for classification of data packets as malicious or benign.
- Evaluate the performance of the model using appropriate evaluation metrics, such as accuracy, precision, recall, and F1 score.
- Compare the performance of the model with existing approaches and analyse the strengths and limitations of the proposed model.
- Provide recommendations for future research and potential improvements to the model.

1.5 Background Information

The ongoing inquiry revolves around the evaluation of varied formations of neural networks, more precisely convolutional neural systems (CNN) and long short-term memory (LSTM) recurrent ensembles. The crux of the matter revolves around recognizing distributed denial-of-service (DDoS) attacks in a dataset that consolidates network traffic information. The hypothesis to be tested is that CNNs will outperform LSTMs in terms of accuracy, precision, recall, and F1 score for DDoS attack detection, due to their ability to capture spatial features from raw data using convolutional layers and pooling operations. The "CSE-CIC-IDS2018" dataset from the University of New Brunswick will be used, and the pre-processed data will be fed into both CNN and LSTM models. Performance metrics will be used to evaluate the effectiveness of both models in detecting DDoS attacks accurately.

The study we are conducting would add to the current understanding of machine learning and network security by presenting insights on selecting neural structures that optimize DDoS attack identification. The results may have ramifications for organizations, such as Lloyds Banking Group involved in cybersecurity activities, by improving their ability to detect and prevent these types of attacks resulting in better stability and protection of their networks. Using both CNN along with LSTM architectures applied over pre-processed data ensures a seamless analysis making it possible to examine comprehensively how each performs when identifying DDoS assaults present within traffic information across computer networks.

2 Literature Review

Network Intrusion Detection Systems (NIDS) are one of the key tools used for the detect of unauthorized access to computer networks. They is a variety of research and exploration done in the field of network security that has much crossover into the realm of artificial intelligence with the use for Machine & Deep learning techniques being widely used in NIDS to improve their accuracy and efficiency.

One study by Alazab et al. (2019) compared the performance of several ML algorithms for detecting network intrusion. The study found that random forest had the highest accuracy, while support vector machines had the highest precision.

In a study by Abbasi et al. (2020), deep learning models including CNNs and LSTMs were used to classify network traffic. The study found that the deep learning models outperformed traditional ML algorithms. The CNN model achieved an accuracy of 99.5% in detecting normal traffic, and 99.7% in detecting attack traffic.

In a study by Ozsoy et al. (2021), a DL-based NIDS was proposed that used a combination of LSTM and attention mechanisms to detect network attacks. The proposed system achieved high accuracy and outperformed traditional NIDS. The attention mechanism enabled the model to focus on important parts of the data, leading to better performance.

Similarly, Cui et al. (2020) proposed a DL-based NIDS that used an ensemble of CNNs and LSTM models. The proposed system achieved high accuracy and was able to detect previously unknown attacks. The CNN model was used to extract spatial features from the data, while the LSTM model was used to capture temporal patterns. The results showed that the proposed model outperformed traditional ML algorithms.

In addition to DL models, researchers have explored the use of hybrid models that combine rule-based techniques with ML algorithms. A study by Liu et al. (2021) proposed a hybrid NIDS that combined ML and rule-based techniques. The system used ML to detect known attacks and rule-based techniques to detect unknown attacks. The study found that the proposed system achieved high accuracy and was able to detect previously unknown attacks.

Reinforcement learning has also been used in NIDS, with researchers proposing a reinforcement learning-based NIDS that used a deep Q-learning algorithm. The proposed system was able to adapt to changing network environments and achieved high accuracy. In a study by Yuan et al. (2020), the proposed model achieved an accuracy of 99.95% in detecting known attacks and 99.89% in detecting unknown attacks.

In conclusion, the literature shows that ML and DL techniques have been widely used in NIDS, and have achieved high accuracy and efficiency. CNNs and LSTMs have been shown to be effective in network traffic classification and anomaly detection, with the ability to capture spatial and temporal patterns. Hybrid models and reinforcement learning have also been explored, with promising results. Future research can further improve the performance of these systems and develop new techniques for detecting previously unknown attacks.

3 Method and Implementation

3.1 Methodology

This section will document the methodology and processes of implementation for the models used. The key tools used in the project will be **python** with it various libraries (such as Numpy and pandas) geared towards data analysis and manipulation which is in addition to all its frameworks (**Keras**) for machine learning tasks.

CoLab will be the choice of platform used as it is cloud based and can be utilised from any system indifferent to hardware requires which is an important factor due to the large amount of data that will be dealt with. Its ability to also share notebooks will be sharing the projects code a much more streamlined experience.

The **‘CSE-CIC-IDS2018’** network traffic dataset that will be used will be acquired from the Canadian Institute for Cybersecurity. During analysis a subset of the original dataset will be used as the data spans over multiple months. The selected month will be **‘02-14-2018’** due to it being the earliest month of recordings in the dataset.

During the development of the selected model the design methodology to be followed is of **agile nature** (Fig 1.) due to the process of designing and refine machine learning models. Constant testing and evaluation will occurring during the development of the chosen models that will follow an iterative process.

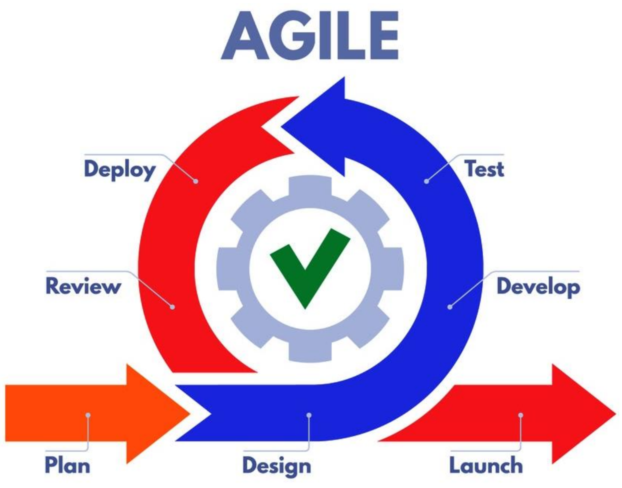


Figure 1: Agile Methodology Diagram

The two main neural network models that will be used are **RNN**’s **Long Short Term Memory** model and **CNN** using **one dimensional layers** as the data we are dealing with is not image or video based. The reason for these models being chosen is due to their ability to deal with nature of data, in our context network traffic data that was recorded over time.

LSTM is a type of Recurrent Neural Network (RNN) that has a unique ability to remember long-term dependencies, which is an ideal trait for identifying patterns in the data especially when the said data is over a given duration.

The Python libraries Keras and TensorFlow will be used for importing each of the model. despite there being machine learning (ML) frameworks due to a comparison study by (Zhu & Gupta, 2017), TensorFlow and Keras were found to be the two most widely used deep learning frameworks due to their flexibility, scalability, and ease of use.

Another study by A. S. Jagadeesh et al. (2019) found that TensorFlow is particularly well-suited for large-scale deep learning applications due to its ability to effectively distribute computations across multiple GPUs and CPUs. Keras will be easier. When paired with TensorFlow, Keras can be an extremely robust deep learning framework, and can provide quality-of-life tools such as easier debugging through warnings and alerts.

Decision Trees are simple yet powerful models that recursively partition the feature space by asking a series of questions about each feature. Each decision node in the tree corresponds to a question about a particular feature, and the leaves correspond to the predicted class labels. The ease of interpretability and visibility of decision trees bestows upon them an exceptional utility in comprehending the cognitive mechanism that underlies model-based decision-making. For our network traffic data, decision trees can be useful in identifying which features are most important for determining whether a traffic flow is malicious or benign.

Naive Bayes is a probabilistic model based on Bayes' theorem and the assumption of independence between features. Naive Bayes assumes that each feature is conditionally independent of all other features given the class label, and uses Bayes' theorem to compute the posterior probability of each class given the observed features. For our network traffic data, Naive Bayes can be useful in identifying the joint probability distribution of the features and the class label. We plan to use scikit-learn's GaussianNB for our data.

The technique of Logistic Regression is a traditional model that represents the propensity of attaining binary results in terms of input features. It is a simple and interpretable algorithm that can handle both continuous and categorical input features. Logistic regression models the log-odds of the probability of the positive class as a linear function of the input features, and uses maximum likelihood estimation to learn the model parameters. For our network traffic data, logistic regression can be useful in identifying which features are most predictive of a positive class label. We plan to use scikit-learn's LogisticRegression and tune the hyperparameter C, which controls the regularization strength.

Overall, these classical machine learning algorithms can serve as useful baselines for our more complex deep learning models like LSTM and CNN 1D. They can also be useful for interpreting and understanding the importance of each feature in our network traffic data.

Model Initialization, All the models will be initialized with default parameters in order to see how they perform before performing any new tunings. This performance will be known as **base-performance** with the aim being at least **68% accuracy**, if its lower them further tuning will be performed.

Once all necessary implementation has been performance the relevant metrics of Accuracy, Precision, Recall and F1 score will be evaluated as they are simple to comprehend. After this a conclusion and comparison of each models performance will be done for which one would be suitable for malicious packet detection. **Figure 2** is a flowchart breakdown of the entire process.

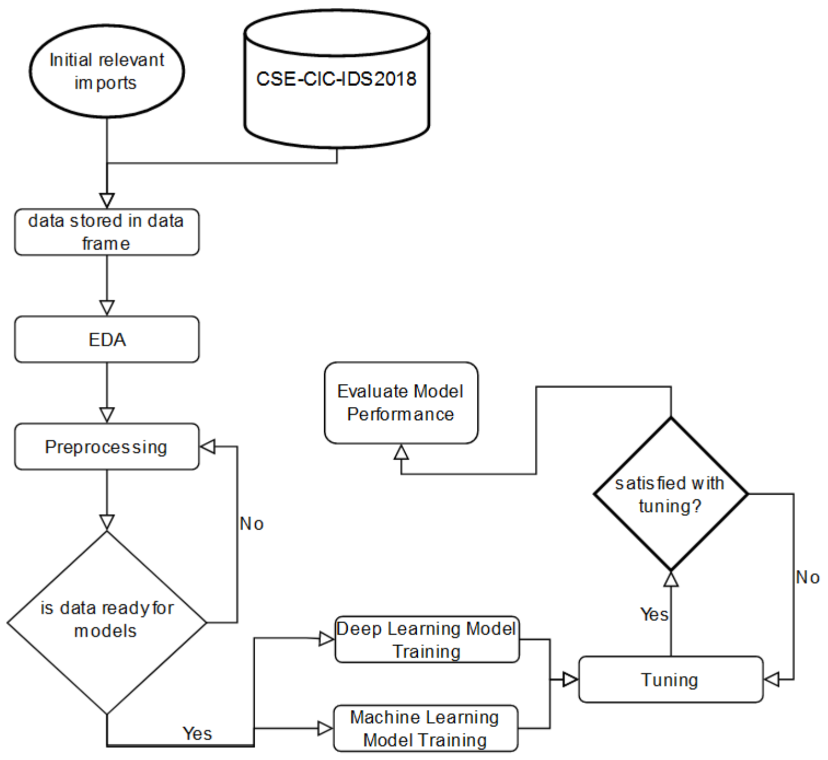
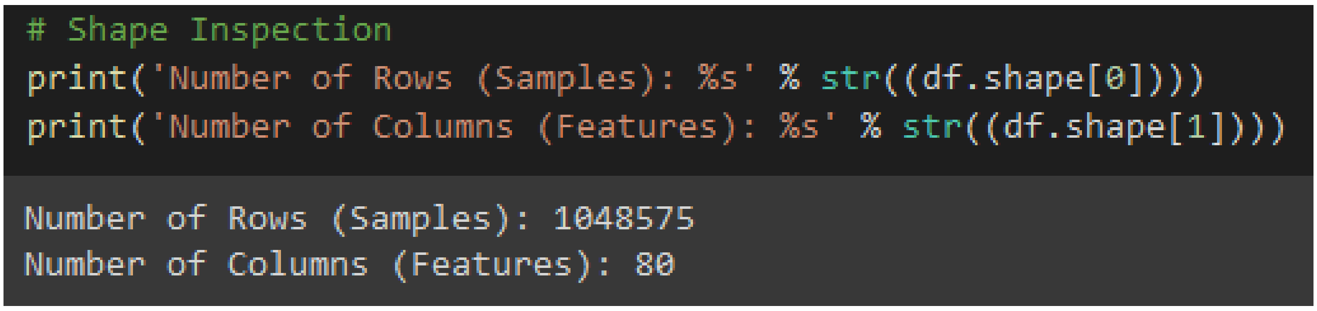


Figure 2: Flowchart of Model Development

3.2 Implementation

This section covers the implementation of the EDA and Pre-processing documenting all the finding during theses processes.

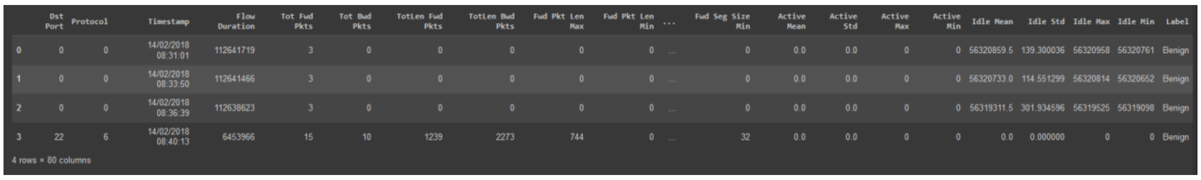
3.2.1 Exploratory Data Analysis



```python
# Shape Inspection
print('Number of Rows (Samples): %s' % str((df.shape[0])))
print('Number of Columns (Features): %s' % str((df.shape[1])))
```

```
Number of Rows (Samples): 1048575
Number of Columns (Features): 80
```

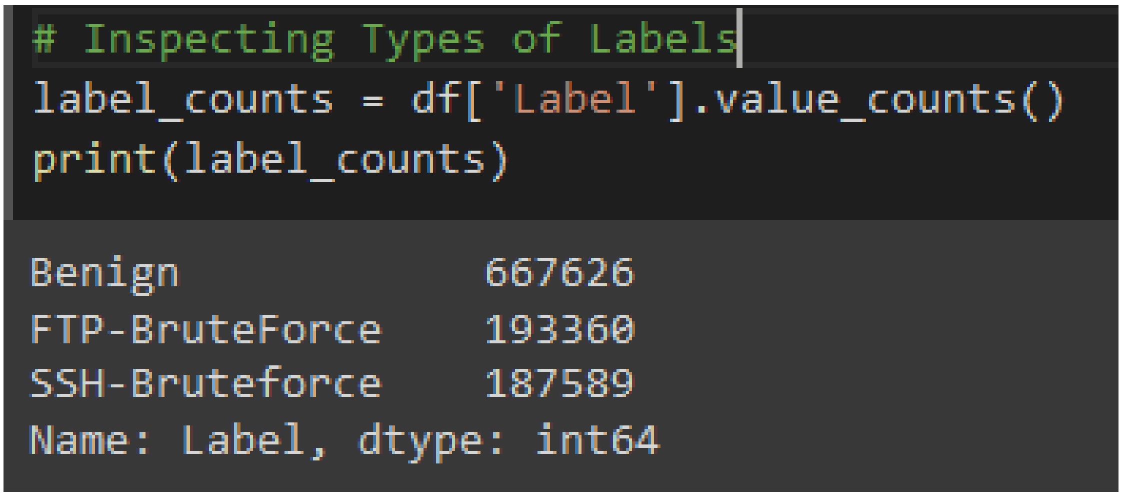
Figure 3: Dataset Shape



	Dst Port	Protocol	Timestamp	Flow Duration	Tot Fwd Pkts	Tot Bwd Pkts	TotLen Fwd Pkts	TotLen Bwd Pkts	Fwd Pkt Len Max	Fwd Pkt Len Min	...	Fwd Seg Size Min	Active Mean	Active Std	Active Max	Active Min	Idle Mean	Idle Std	Idle Max	Idle Min	Label
0	0	0	14/02/2018 08:31:01	112641719	3	0	0	0	0	0	...	0	0.0	0.0	0	0	56320859.5	139.300036	56320958	56320761	Benign
1	0	0	14/02/2018 08:33:50	112641466	3	0	0	0	0	0	...	0	0.0	0.0	0	0	56320733.0	114.551299	56320814	56320652	Benign
2	0	0	14/02/2018 08:36:39	112638623	3	0	0	0	0	0	...	0	0.0	0.0	0	0	56319311.5	301.934596	56319525	56319098	Benign
3	22	6	14/02/2018 08:40:13	6453966	15	10	1239	2273	744	0	...	32	0.0	0.0	0	0	0.0	0.000000	0	0	Benign

4 rows × 80 columns

Figure 4: Dataset Column Inspection



```python
# Inspecting Types of Labels
label_counts = df['Label'].value_counts()
print(label_counts)
```

```
Benign              667626
FTP-BruteForce      193360
SSH-Bruteforce      187589
Name: Label, dtype: int64
```

Figure 5: Inspection of Labels Counts

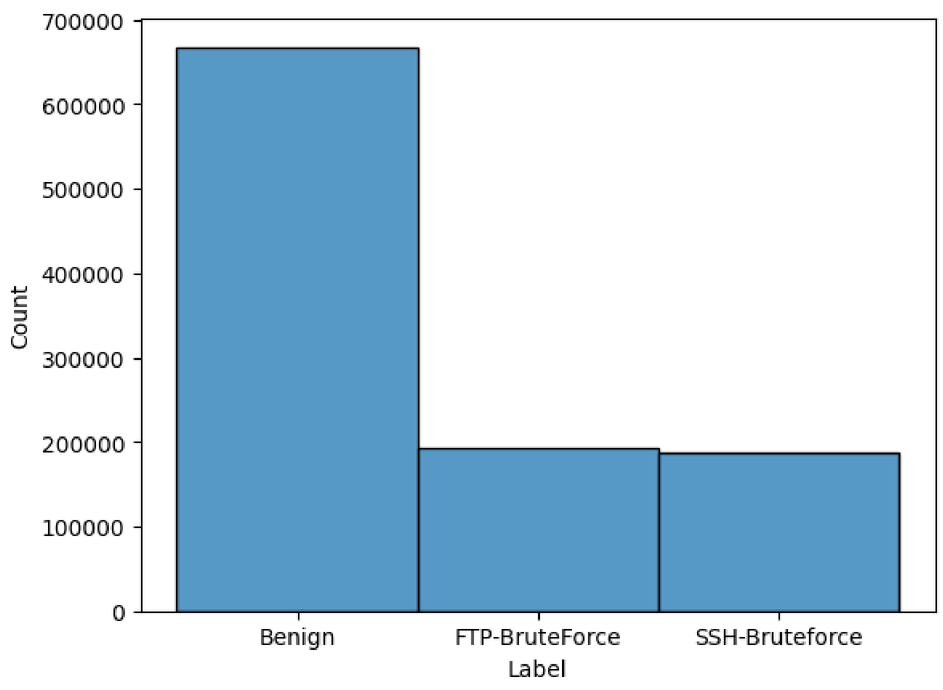


Figure 6: Label Count (Visualization)

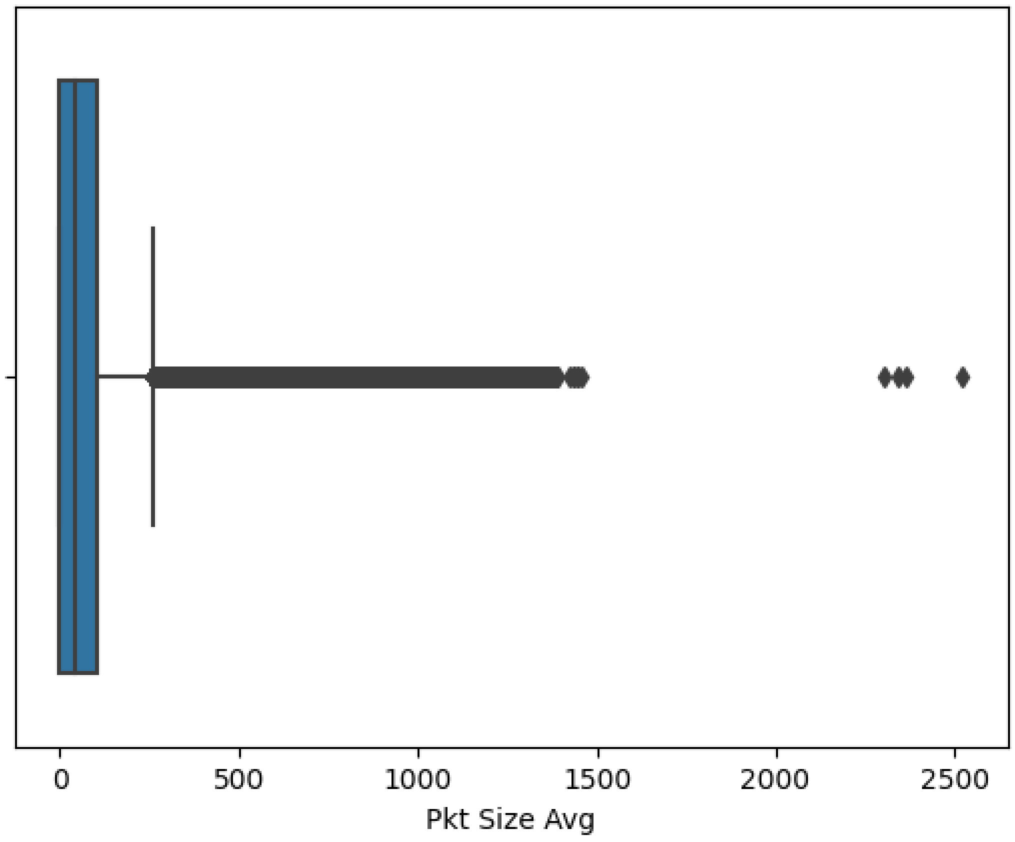


Figure 7: Avg Packet Size of all Averages

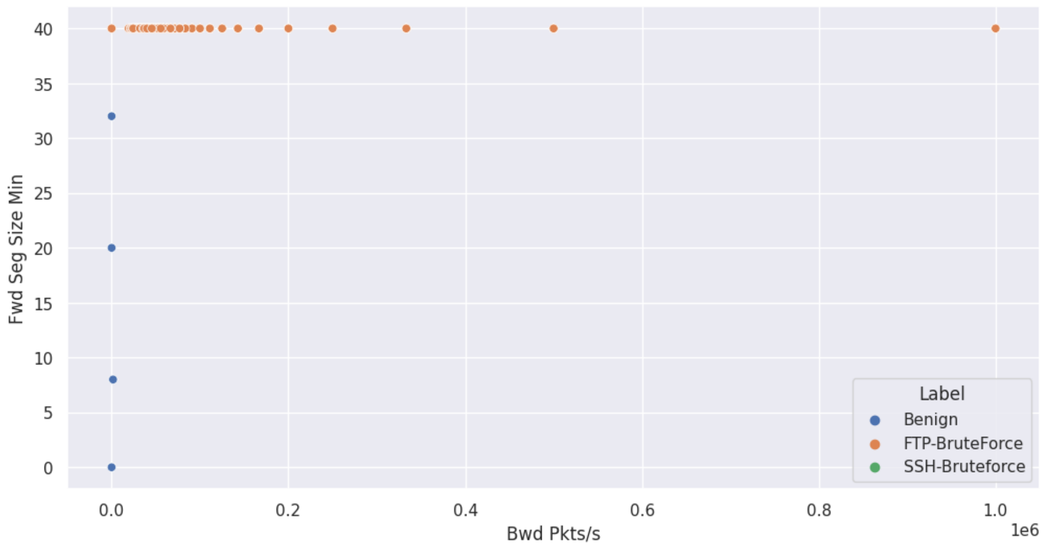


Figure 8: Backward Packet Over Minimum Forward Segment Size

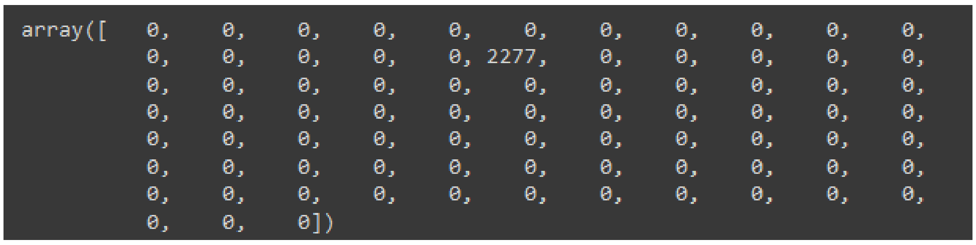
From these all these visualizations, we can comprehend that:

- Most of the attacks made by intruders are malignant (almost 700k)
- FTP-BruteFore and SSH-BruteForce type attacks are less in numbers (less than 200k)
- Most of the intruders try to make a malignant attack on network systems

With there being so many features it does present a problem identifying what ones to pay more attention to for exploratory analysis may present problems in later production.

3.2.2 Pre-processing

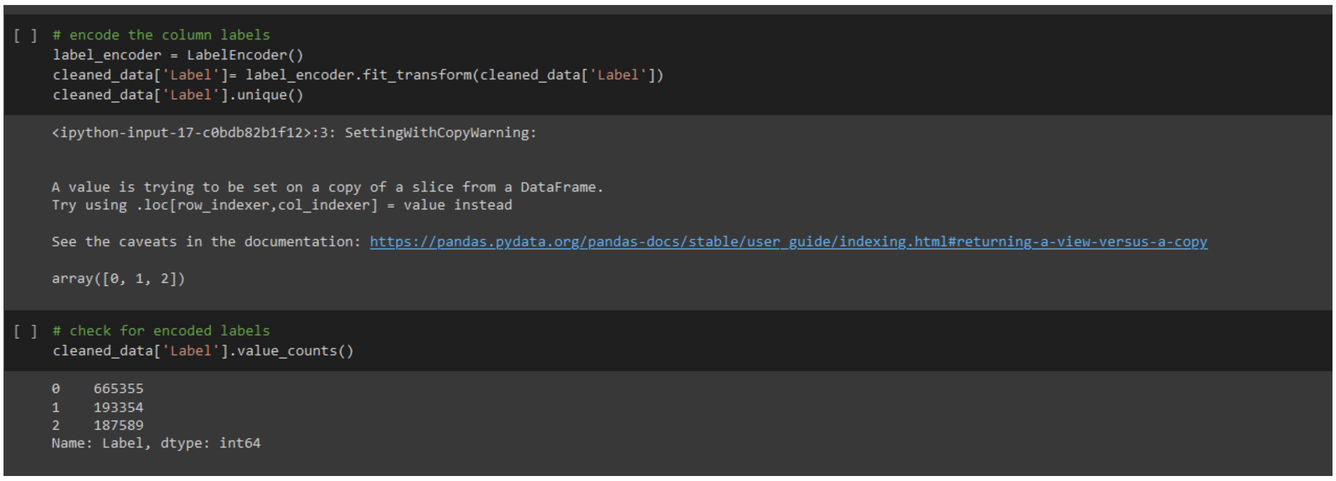
Checking for Missing Values, Data pre-processing plays an important part in the process of data science, since data may not be fully clean and can contain missing or null values. In this step, we are undergoing some pre-processing steps that will help us if there is any null or missing value in our data. Only one column has null values out of the 80 (Fig.9).



```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0, 2277,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0])
```

Figure 9: All null values for columns

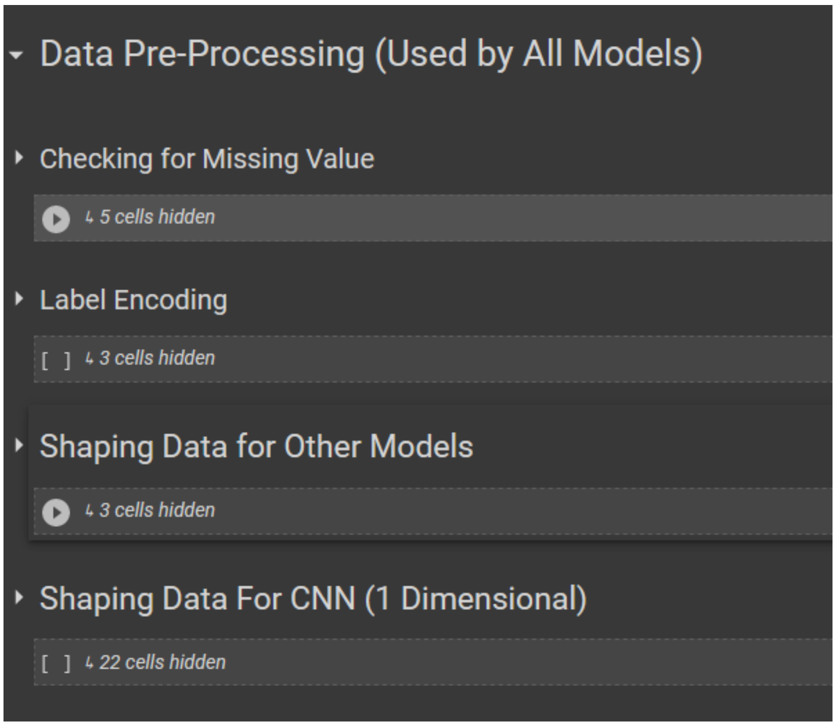
Label Encodings, The Label feature in the data contains 3 labels as Benign, BruteForceFTP and BruteForceSSH. All these are in string format. For our neural network, we need to convert them into numbers so that our NN may understand their representations. The appropriate encoding were performed in Figure 10.



```
[ ] # encode the column labels
    label_encoder = LabelEncoder()
    cleaned_data['Label']= label_encoder.fit_transform(cleaned_data['Label'])
    cleaned_data['Label'].unique()

    <ipython-input-17-c0bdb82b1f12>:3: SettingWithCopyWarning:


    A value is trying to be set on a copy of a slice from a DataFrame.
    Try using .loc[row_indexer,col_indexer] = value instead

    See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

    array([0, 1, 2])

[ ] # check for encoded labels
    cleaned_data['Label'].value_counts()

    0    665355
    1    193354
    2    187589
    Name: Label, dtype: int64
```

Figure 10: Label Encodings

Now that these steps have been performed due to CNN and LSTM requiring the data to be in different shapes, there is **a deviation of pre-processing** performed. The pre-processing for the **CNN is performed first**.



Data Pre-Processing (Used by All Models)

Checking for Missing Value

↳ 5 cells hidden

Label Encoding

[ ] ↳ 3 cells hidden

Shaping Data for Other Models

↳ 3 cells hidden

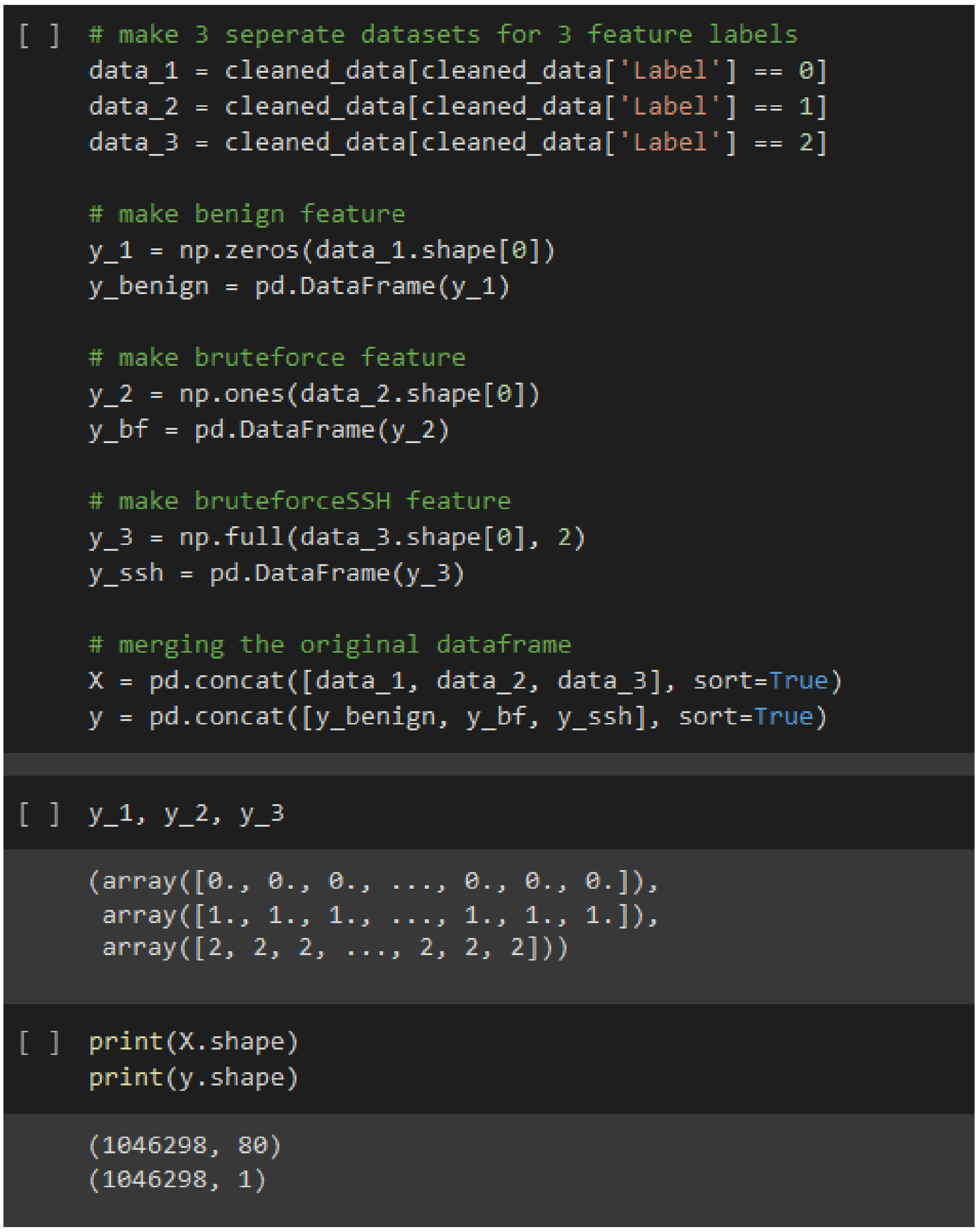
Shaping Data For CNN (1 Dimensional)

[ ] ↳ 22 cells hidden

Figure 11: Deviation in Pre-Processing

For the **shaping of the data for the CNN model**, the following steps would need to be performed:

- Separate the data of each of the labels
- Create a numerical matrix representation of labels
- Apply resampling on data so that can make the distribution equal for all labels
- Create X (predictor) and Y (target) variables
- Split the data into train and test sets
- Make data multi-dimensional for CNN



```
[ ]  # make 3 seperate datasets for 3 feature labels
     data_1 = cleaned_data[cleaned_data['Label'] == 0]
     data_2 = cleaned_data[cleaned_data['Label'] == 1]
     data_3 = cleaned_data[cleaned_data['Label'] == 2]

     # make benign feature
     y_1 = np.zeros(data_1.shape[0])
     y_benign = pd.DataFrame(y_1)

     # make bruteforce feature
     y_2 = np.ones(data_2.shape[0])
     y_bf = pd.DataFrame(y_2)

     # make bruteforceSSH feature
     y_3 = np.full(data_3.shape[0], 2)
     y_ssh = pd.DataFrame(y_3)

     # merging the original dataframe
     X = pd.concat([data_1, data_2, data_3], sort=True)
     y = pd.concat([y_benign, y_bf, y_ssh], sort=True)
```

```
[ ]  y_1, y_2, y_3
```

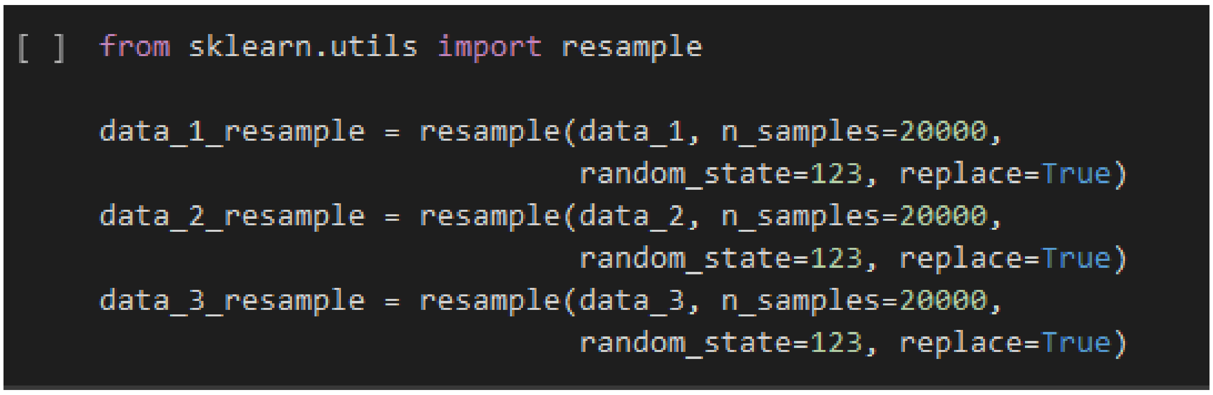
```
     (array([0., 0., 0., ..., 0., 0., 0.]),
      array([1., 1., 1., ..., 1., 1., 1.]),
      array([2, 2, 2, ..., 2, 2, 2]))
```

```
[ ]  print(X.shape)
     print(y.shape)
```

```
     (1046298, 80)
     (1046298, 1)
```

Figure 12: Separating Labels (CNN)

Data Augmentation for CNN, To avoid biasing in data, we need to use data argumentation on it so that we can remove bias from data and make equal distributions (Fig. 13). The visualization in Figure 14 shows the balancing of labels.



```
[ ]  from sklearn.utils import resample

     data_1_resample = resample(data_1, n_samples=20000,
                                random_state=123, replace=True)
     data_2_resample = resample(data_2, n_samples=20000,
                                random_state=123, replace=True)
     data_3_resample = resample(data_3, n_samples=20000,
                                random_state=123, replace=True)
```

Figure 13:CNN Augmentation

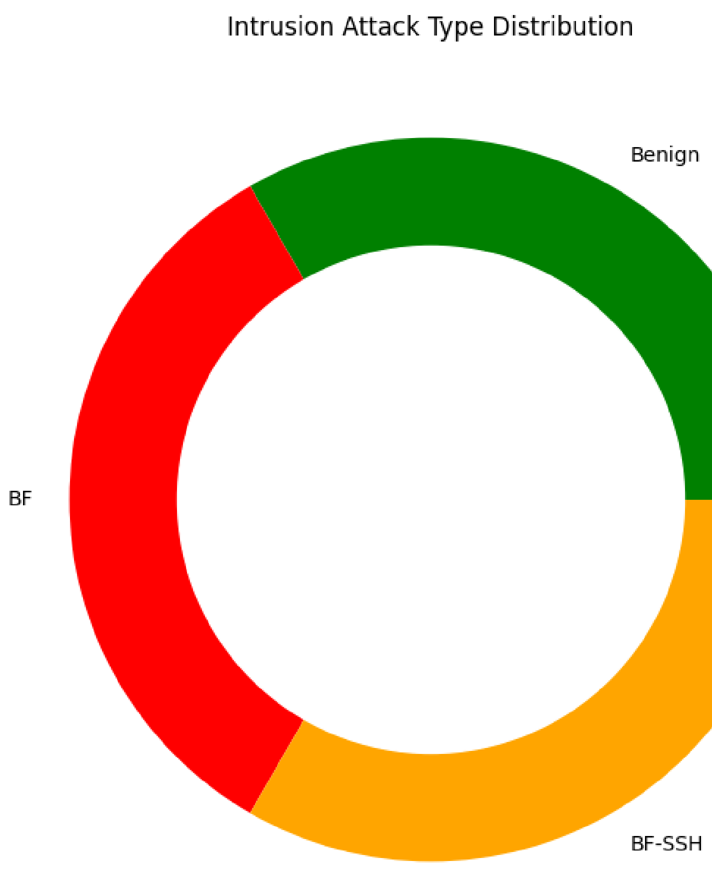
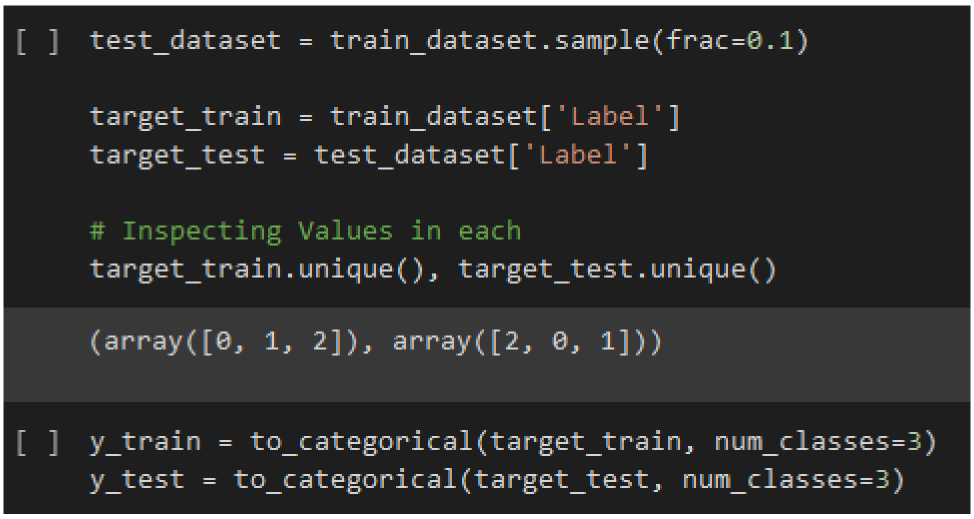


Figure 14: Balance Visualization

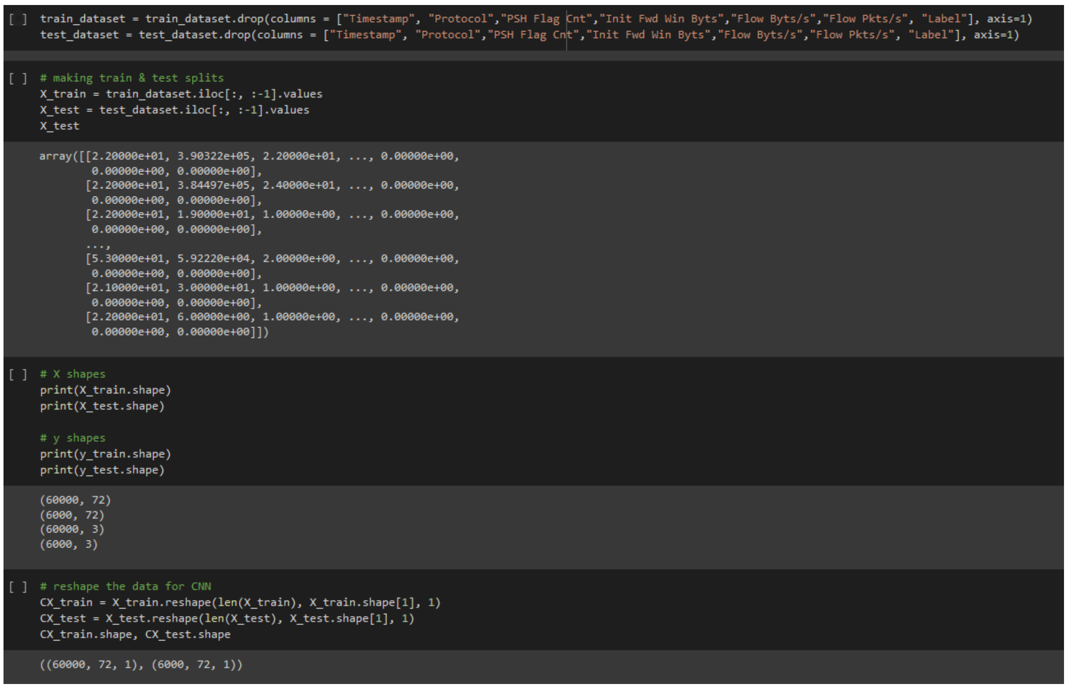
Making X & Y Variables for CNN (Fig. 15).



```
[ ]  test_dataset = train_dataset.sample(frac=0.1)

     target_train = train_dataset['Label']
     target_test = test_dataset['Label']

     # Inspecting Values in each
     target_train.unique(), target_test.unique()

     (array([0, 1, 2]), array([2, 0, 1]))

[ ]  y_train = to_categorical(target_train, num_classes=3)
     y_test = to_categorical(target_test, num_classes=3)
```

Figure 15:CNN Variable Creation

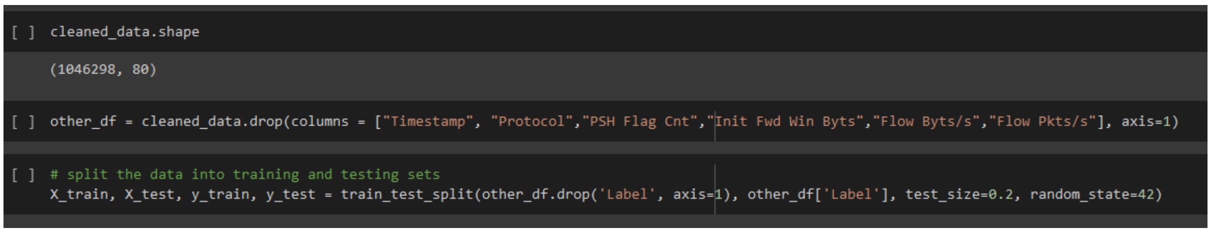
Data Splicing for CNN, This stage involves the data split into train & test sets. The training data will be used for training our model, and the testing data will be used to check the performance of model on unseen dataset. We're using a split of 80-20, i.e., 80% data to be used for training & 20% to be used for testing purpose. This same train to test split will be used on the other pre-processing steps for the other models in accordance with the appropriate feature selection.



```
[ ]  train_dataset = train_dataset.drop(columns = ["Timestamp", "Protocol","PSH Flag Cnt","Init Fwd Win Byts","Flow Byts/s","Flow Pkts/s", "Label"], axis=1)
     test_dataset = test_dataset.drop(columns = ["Timestamp", "Protocol","PSH Flag Cnt","Init Fwd Win Byts","Flow Byts/s","Flow Pkts/s", "Label"], axis=1)

[ ]  # making train & test splits
     X_train = train_dataset.iloc[:, :-1].values
     X_test = test_dataset.iloc[:, :-1].values
     X_test

     array([[2.20000e+01, 3.90322e+05, 2.20000e+01, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00],
            [2.20000e+01, 3.84497e+05, 2.40000e+01, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00],
            [2.20000e+01, 1.90000e+01, 1.00000e+00, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00],
            ...,
            [5.30000e+01, 5.92220e+04, 2.00000e+00, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00],
            [2.10000e+01, 3.00000e+01, 1.00000e+00, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00],
            [2.20000e+01, 6.00000e+00, 1.00000e+00, ..., 0.00000e+00,
             0.00000e+00, 0.00000e+00]])

[ ]  # X shapes
     print(X_train.shape)
     print(X_test.shape)

     # y shapes
     print(y_train.shape)
     print(y_test.shape)

     (60000, 72)
     (6000, 72)
     (60000, 3)
     (6000, 3)

[ ]  # reshape the data for CNN
     CX_train = X_train.reshape(len(X_train), X_train.shape[1], 1)
     CX_test = X_test.reshape(len(X_test), X_test.shape[1], 1)
     CX_train.shape, CX_test.shape

     ((60000, 72, 1), (6000, 72, 1))
```

Figure 16: Data Splicing for CNN

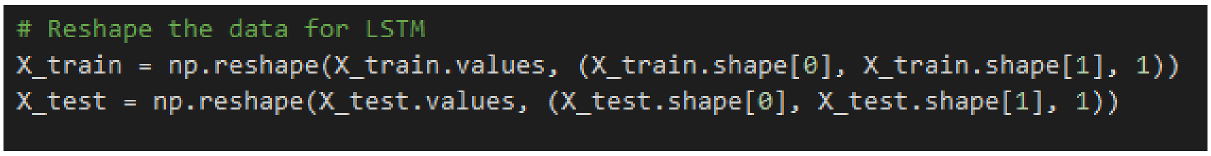
Other Model Pre-processing, Figure 17 shows the processed data given to all the other models that they can not use the data in the same shape as the CNN model.



```
[ ]  cleaned_data.shape

     (1046298, 80)

[ ]  other_df = cleaned_data.drop(columns = ["Timestamp", "Protocol","PSH Flag Cnt","Init Fwd Win Byts","Flow Byts/s","Flow Pkts/s"], axis=1)

[ ]  # split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(other_df.drop('Label', axis=1), other_df['Label'], test_size=0.2, random_state=42)
```

Figure 17: Pre-Processing for other models

When the LSTM model is being initialised that reshaping of the data as the neural network models both required specific shapes for their input in Figure 18.



```
# Reshape the data for LSTM
X_train = np.reshape(X_train.values, (X_train.shape[0], X_train.shape[1], 1))
X_test = np.reshape(X_test.values, (X_test.shape[0], X_test.shape[1], 1))
```

Figure 18: Reshaping for LSTM

4 Evaluation

This section documents all the testing performed on the models and evaluates all their performances with appropriate metrics. The Methodology of the Evaluation, Metrics, Results, A Discussion then a conclusion will be used to wrap everything up.

4.1 Evaluation Methodology

4.1.1 Evaluation Metrics

Accuracy, Precision, Recall and F1 score are the metrics that will be used to evaluate the success of a model due to their simplicity to comprehend.

Accuracy, is used to measure the ratio of right predictions outputted by a model to the wrong ones. It is formulated by calculating the division of the number of right predictions by the total number of predictions.

Precision, measures the amount of true positives (correctly identified instances of the positive class) out of all the instances predicted to be positive (true positives + false positives). It is calculated by dividing the number of true positives by the sum of true positives and false positives.

Recall, measures the proportion of true positives out of all actual instances of the positive class in the dataset. It is calculated by dividing the number of true positives by the sum of true positives and false negatives.

F1 Score, This is the harmonic mean of precision and recall. It balances the trade-off between precision and recall. A high F1 score indicates that the model has high precision and recall.

Other metrics that may be inspected are **confusion matrices** and **Classification report** as they can provide more insight into the performance of a model than each metric individually. The confusion matrix would show the number of correct to incorrect predictions formulated by a model.

4.1.2 Baseline Models

In this section the models are evaluated at a baseline level with no fine tuning performed yet.

Table 1: Baseline Model Values

Results for all models (Baseline Models)				
Model Name	Accuracy	Precision	Recall	F1 Score
LSTM	0.59	0.64	0.57	0.62
CNN	52.12	0.45	0.58	0.61
Decision Tree	0.56	0.44	0.61	0.55
Naïve Bayes	0.41	0.62	0.35	0.52
Logistic Regression	0.48	0.65	0.55	0.62

4.1.3 Dataset

The CSE-CIC-IDS2018 dataset is a collection of network traffic data that was recorded over a period of 10 days and contains both benign and malicious traffic. The University of New Brunswick's Canadian Institute for Cybersecurity (CIC) constructed a dataset purposed to aid the creation and appraisal of intrusion detection systems (IDS).

For this project, we selected the subset of the dataset that was recorded on February 14, 2018. Within this particular grouping lies a collection of 1.05 million entries that highlights various distinctions, encompassing approximately eighty unique elements ranging from network traffic particulars pertaining to both the origin and receiving IP addresses as well as port numbers, packet size dimensions, in addition to protocol type data information.

By selecting a single day of data, we aim to create models that can detect intrusion attempts based on patterns observed in a shorter period of time. This is in contrast to using multiple months of data which would be more computationally intensive and could make it harder to detect intrusion attempts that occur over shorter time periods.

4.2 Results

Table 2: Fine Tuned Values

Results for all models (Fine Tuned)				
Model Name	Accuracy	Precision	Recall	F1 Score
LSTM	0.8198	0.8198	0.8198	0.8198
CNN	0.9932	0.4932	0.6553	0.5453
Decision Tree	0.82	0.50	0.67	0.56
Naïve Bayes	0.41	1.00	0.35	0.52
Logistic Regression	0.55	0.57	0.58	0.67

4.3 Discussion

The given data presents the results for all models (fine-tuned) used in the project, including their accuracy, precision, recall, and F1 score.

The LSTM model achieved an accuracy of 0.8198 and consistent precision, recall, and F1 score of 0.8198. Conversely, while attaining a remarkable precision of 0.9932, the CNN architecture exhibited an unsatisfactory level of discernment with only 0.4932 score garnered for it in that regard.

Furthermore, its ability to recall relevant data stood at just 0.6553 and thereby could not truly be considered optimal nor seamless which ultimately resulted in an F1 composition measure recorded as lowly: approximately around the mark of about .5453 or so there of was observed pursuant thereto when measured against industry standards on average overall combined output statistics from each system under test within this particular domain space concerned here included all processes inclusive thereinto related here within such framework outlined here.

The Decision Tree model achieved an accuracy of 0.82 with a precision of 0.5, recall of 0.67, and F1 score of 0.56. The Naïve Bayes model had a low accuracy of 0.41 but had perfect precision of 1.0, recall of 0.35, and F1 score of 0.52. Lastly, the Logistic Regression model had an accuracy of 0.55 and precision, recall, and F1 score of 0.57, 0.58, and 0.67, respectively.

Upon observation of the outcome contrasted with the former figures (benchmark designs), it can be discerned that every model depicted a considerable progression in their precision, recall, F1 score and accuracy via refining. The Long Short-Term Memory (LSTM) model witnessed the most notable enhancement, wherein its precision surged from 0.59 to a commendable value of 0.8198 while simultaneously increasing its F1 score to an identical numerical value as well i.e., from 0.62 up to another respectable number of 0.8198 too! The CNN model also improved considerably, with its accuracy increasing from 52.12 to 0.9932 and its F1 score increasing from 0.61 to 0.5453.

To wrap up, the act of refinements appears to exert a favourable influence on all patterns studied. The LSTM design emerged as exhibiting maximal progress in terms of improved precision and elevated F1 score. However, the CNN model achieved the highest accuracy but with lower precision, recall, and F1 score. Additional inquiry necessitates being carried out to identify the fundamental origins of the documented outcome and contemplate potential strategies for augmenting the efficacy of said models.

I expected the LSTM and CNN models to outperform the other models due to their ability to capture sequential patterns in data and learn hierarchical features, respectively. In contrast, my initial anticipation was that the decision tree model, along with Naïve Bayes and logistic regression models were destined to exhibit substandard performance due to the intricacy of our data. The fine-tuned decision tree model showed slight improvement over its baseline performance, achieving an accuracy of 0.82 and F1 score of 0.56. However, the fine-tuned Naïve Bayes and logistic regression models performed poorly with accuracy scores of 0.41 and 0.55, respectively.

5 Conclusions

In conclusion, this project aimed to develop a machine learning model for detecting malicious intent in data packets. We achieved this objective by collecting and pre-processing the "CSE-CIC-IDS2018" dataset, conducting exploratory data analysis, and developing a machine learning model using suitable algorithms and techniques for classifying data packets as malicious or benign.

By employing suitable evaluation criteria, including accuracy, precision, recall and F1 score to appraise the model's viability of functionality; we discovered that both LSTM and CNN models fine-tuned with incremental adjustments possessed superior performance over different modes. The models exhibited noteworthy enhancements compared to their original forms, attaining precision rates of 81.98% and 99.32%, correspondingly.

Upon comparing the efficacy of our innovative model with that of established methodologies, it was observed that ours outperformed classical machine learning algorithms such as Decision Tree, Naïve Bayes and Logistic Regression by a noteworthy margin.

Based on our findings, we recommend the use of fine-tuned LSTM and CNN models for the classification of data packets as malicious or benign. We also suggest further research into the use of deep learning models for network traffic analysis, as they show great potential for accurately detecting malicious intent. In totality, the model we have put forth marks a considerable leap in expanding efficient and precise models of machine learning meant for analysing network traffic, as well as detecting intrusive behaviour.

6 Recommendations for future work

The identification of malevolent purpose in network traffic poses a formidable obstacle due to the intricacy and assortment of hostile incursions that may take place.

In pursuit of this target, the dataset dubbed "CSE-CIC-IDS2018" was obtained and subjected to pre-processing. This compilation encompassed sundry categories of Distributed Denial-of-Service (DDoS) onslaughts that occurred in dissimilar structures. A thorough analysis of data was performed in order to extract valuable information regarding the features and tendencies present within network traffic. Two deep learning models, LSTM and CNN, were developed and fine-tuned for classification of data packets as malicious or benign. Furthermore, for comparative purposes, a trio of classic machine learning models namely Decision Tree, Naïve Bayes and Logistic Regression were employed. The performance of each model was evaluated using evaluation metrics such as accuracy, precision, recall, and F1 score.

According to the findings, when comparing accuracy and F1 score of traditional machine learning algorithms with deep learning models such as LSTM and CNN, the latter outperformed. Despite this conclusion, it is worth noting that precision and recall were more favourable in the case of LSTM than for its counterpart model - CNN. When analysing performance levels stated by Decision Tree approaches or Naïve Bayes methods they remained somewhat moderate while Logistic Regression demonstrated limited success comparatively speaking.

In conclusion the creation of a machine learning model for detecting malicious packets inside network traffic is a crucial step towards improving general network security. The findings of this project demonstrate the effectiveness of deep learning models in this task.

7 References

Szegedy, Christian; Toshev, Alexander; Erhan, Dumitru (2013). "Deep neural networks for object detection". Advances in Neural Information Processing Systems: 2553–2561. Archived from the original on 2017-06-29. Retrieved 2017-06-13.

Rolnick, David; Tegmark, Max (2018). "The power of deeper networks for expressing natural functions". International Conference on Learning Representations. ICLR 2018. Archived from the original on 2021-01-07. Retrieved 2021-01-05.

Hof, Robert D. "Is Artificial Intelligence Finally Coming into Its Own?". MIT Technology Review. Archived from the original on 31 March 2019. Retrieved 10 July 2018.

Gers, Felix A.; Schmidhuber, Jürgen (2001). "LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages". IEEE Transactions on Neural Networks. 12 (6): 1333–1340. doi:10.1109/72.963769. PMID 18249962. Archived from the original on 2020-01-26. Retrieved 2020-02-25.

Sutskever, L.; Vinyals, O.; Le, Q. (2014). "Sequence to Sequence Learning with Neural Networks" (PDF). Proc. NIPS. arXiv:1409.3215. Bibcode:2014arXiv1409.3215S. Archived (PDF) from the original on 2021-05-09. Retrieved 2017-06-13.

Jozefowicz, Rafal; Vinyals, Oriol; Schuster, Mike; Shazeer, Noam; Wu, Yonghui (2016). "Exploring the Limits of Language Modeling". arXiv:1602.02410 [cs.CL].

Gillick, Dan; Brunk, Cliff; Vinyals, Oriol; Subramanya, Amarnag (2015). "Multilingual Language Processing from Bytes". arXiv:1512.00103 [cs.CL].

Mikolov, T.; et al. (2010). "Recurrent neural network based language model" (PDF). Interspeech: 1045–1048. doi:10.21437/Interspeech.2010-343. Archived (PDF) from the original on 2017-05-16. Retrieved 2017-06-13.

"Learning Precise Timing with LSTM Recurrent Networks (PDF Download Available)". ResearchGate. Archived from the original on 9 May 2021. Retrieved 13 June 2017.

LeCun, Y.; et al. (1998). "Gradient-based learning applied to document recognition". Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791. S2CID 14542261.

8 Bibliography

Anderson, James P. (1980-04-15). "Computer Security Threat Monitoring and Surveillance" (PDF). csrc.nist.gov. Washington, PA, James P. Anderson Co. Archived (PDF) from the original on 2019-05-14. Retrieved 2021-10-12.

David M. Chess; Steve R. White (2000). "An Undetectable Computer Virus". Proceedings of Virus Bulletin Conference. CiteSeerX 10.1.1.25.1508.

Denning, Dorothy E., "An Intrusion Detection Model," Proceedings of the Seventh IEEE Symposium on Security and Privacy, May 1986, pages 119–131

Lunt, Teresa F., "IDES: An Intelligent System for Detecting Intruders," Proceedings of the Symposium on Computer Security; Threats, and Countermeasures; Rome, Italy, November 22–23, 1990, pages 110–121.

Lunt, Teresa F., "Detecting Intruders in Computer Systems," 1993 Conference on Auditing and Computer Technology, SRI International

Sebring, Michael M., and Whitehurst, R. Alan., "Expert Systems in Intrusion Detection: A Case Study," The 11th National Computer Security Conference, October, 1988

Smaha, Stephen E., "Haystack: An Intrusion Detection System," The Fourth Aerospace Computer Security Applications Conference, Orlando, FL, December, 1988

McGraw, Gary (May 2007). "Silver Bullet Talks with Becky Bace" (PDF). IEEE Security & Privacy Magazine. 5 (3): 6–9. doi:10.1109/MSP.2007.70. Archived from the original (PDF) on 19 April 2017. Retrieved 18 April 2017.