Final Project Report

On

ROLE OF MACHINE LEARNING IN DETECTING AND PREVENTING CYBER ATTACKS

Under the guidance

of

Adib Zaman, Ph.D.

By

Deerean Chowdary Velaga

Madhushree LNU

Mohan Rajesh Penkey

Nishchitha Hemanna

Sripriya Vedere

Yashma Sree Bandi

1. Introduction:

People are living in the age of the internet, which, like any other age, has advantages and disadvantages. The primary disadvantage is the security risk [1, 2]. Data breaches are growing more prevalent and devastating as more of mankind's private data is transferred into the digital realm. Cybercriminals are becoming increasingly effective at evading identification, and most recent virus packages are already adding novel methods to prevent antivirus monitoring and other risk detection tools. Information security is at a crossroads, and the next phase of research ought to be directed upon cyber-attack forecasting systems capable of anticipating serious circumstances and effects, instead of relying on defensive measures and concentrating on remediation. Worldwide, systems that utilize a comprehensive, forecasting assessment of cybersecurity hazards are necessary. Prediction, mitigation, recognition, or tracking down threats, as well as handling incidents, are critical functions in cybersecurity. Intelligent machines (AI), which are mostly based on data mining (ML) [3, 4], can identify similarities and forecast future actions depending on previous experiences, avoiding, or identifying possibly hazardous conduct, that was the main goal of this work.

Since it enables computers to acquire knowledge and grow from experiences without needing to be expressly coded, machine learning (ML) has become one of the most widely utilized innovations in thefourth industrialrevolution (4IR)[5, 6]. In the field of safety online, artificial intelligence (AI) can be quite useful in extracting ideas from data. Safety data may be either organized or unorganized and can come from multiple places.

By deriving information from this data, intelligent programs like detecting intrusions, cyber-attacks or detection of anomalies, phishing, malware identification, zero-day assault forecasting, and others can be constructed. In recent days, there has been a significant increase in requests for privacy and safeguarding against cyberirregularities and other types of attacks, including unlawful access,denial-of-service (DoS),malware, phishing emails, botnet, spyware, bugs, and many more. Thus, smart statistical techniques and approaches capable of derivinginsights or pertinent information frominformation in a rapid and smart manner are required for practical cyber operations. Security professionals believe that may safeguard against future assaults by using attack identification or detection approaches.

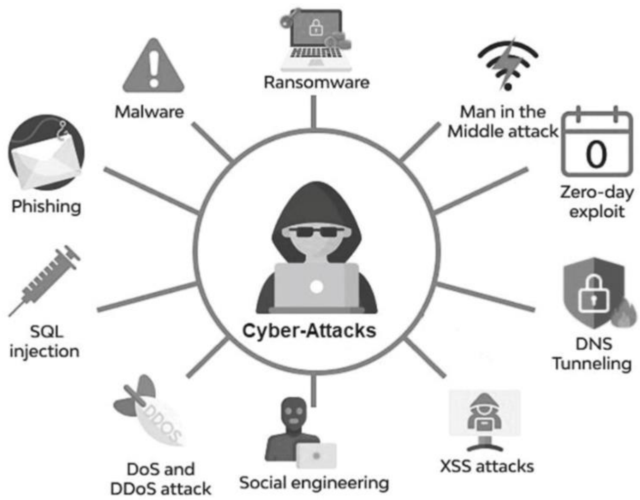

Fig 1: From the perspective of the field of cybersecurity, there are several typical assaults or dangers.

2. Motivation/Background:

Setting up the environment for our piece, we first describe the core principles of Machine Learning clearly [1]. Our objective is to give both recognized vocabulary and typical classes of existing ML procedures. The topic and intended audience for this paper are then defined [9], and the distinctions between our endeavour and previous research are highlighted.

As the number, expertise, and extent of cyber assaults have grown, security has grown into an urgent issue for individuals, corporations, and nations alike. Such assaults may end up in considerable monetary losses, repeated harm, and even national security dangers. Therefore, there is an increasing demand for robust safety precautions to protect from attacks from hackers.

Machine learning has evolved as a highly efficient tool for detecting and preventing attacks via the internet. Machine learning systems can analyze enormous volumes of data, detecting patterns and oddities, and making predictions using the information they collect. It is feasible to identify and react to cyber assaults in real-time by integrating machine learning methods into cybersecurity.

The primaryobjective of MachineLearning (ML) is to create machines that can learn and make choices on their own. A training stage is used to construct an ML model by advising to examine some 'existing' (training) data using a particular ML technique. This type of model contains all the information gained in the training stage and includes functions for deciding based on 'future' information. For a machine learning (ML) model can be used in a production context, its efficacy must be evaluated [10]. To that aim, the machine learning (ML) algorithm processes some 'validation' data, and its forecasts can be analyzed by people or contrasted with established reality. As a result, an ML methodology can be defined as "the procedure for building an ML model using machine learning (ML) algorithms on certain training data." A simplified illustration of the instruction and verification steps is provided.

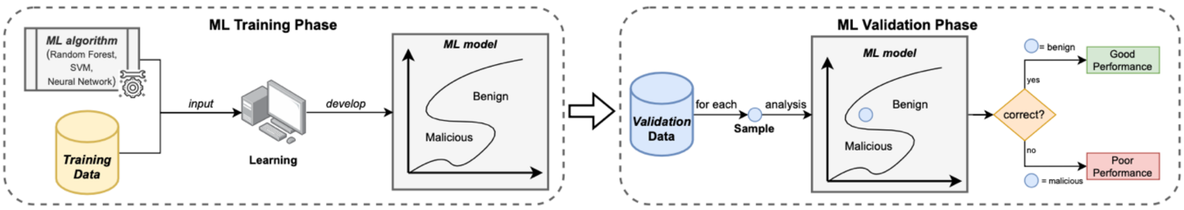

Fig 2: Machine Learning development

In general, as the dangerous environment evolves, the role of artificial intelligence in detecting and avoiding cyber assaults grows more and more important [11]. As a result, there is a rising demand for studies and developments in this field to ensure that companies have possession of the tools and technology required to safeguard themselves from online dangers.

3. Problem Overview:

In the present day, when online dangers are growing more intricate and technologically advanced, the role of artificial intelligence in identifying and combating computer attacks is growing increasingly vital. Machine learning systems can analyze enormous amounts of data and detect abnormalities, patterns, and tendencies that may suggest an upcoming threat.

The dataset in question includes network breach information which may be utilized to build systems for intrusion detection (IDS). The dataset contains communication data acquired by a system that detects intrusions in a replicated system environment. are source and destination IP addresses, connection categories, service categories, and flags. The data also contains a label showing whether the network activity was legitimate or malicious.

The data set may be utilized to train and testdifferent machinelearning as well as deeplearning algorithms for identifying intrusions. It should be noted, nevertheless, that the data set has been collected in a synthetic setting and might not precisely reflect actual internet activity.

It is an archive of data from network traffic used to detect attacks on the network. The data set includes 1.25 million network packets recorded during one hour. Network activity was recorded in a monitored laboratory setting while several attacks were executed on replicated networks. Aims to identify varioussorts of network invasions such as denial-of-serviceattacks, probingattacks, client-to-rootassaults, and controller-to-local attacks,including attacks. The data set contains 40 characteristics, containing data on the origin and target IP addresses as well as the protocol that was employed, along with additional network traffic metrics. Researchers and data analysts can construct artificial intelligence algorithms to effectively identify and categorize various kinds of computer network incursions by evaluating datasets.

4. Dataset description

We have used Kaggle website collect the dataset [12] which was developed at the University of New Brunswick to analyze DDoS data sourced entirely from the year 2018. The dataset used includes a total of 1048575 rows and 80 columns within this dataset. The dataset is in the form of single CSV file. Figure 3 shows the sample of our dataset and the number of columns in the dataset respectively.

Due to the huge amount of data stored in the single CSV file alone, we pre-sort the data within Python for analysis of the dataset, to train and making predictions in the model.

```
In [3]: data = pd.read_csv('Cyber detection dataset.csv')
```

```
In [4]: data
```

Out[4]:

	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
0	0	0	15/02/2018 08:25:18	112641158	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320579.0
1	22	6	15/02/2018 08:29:05	37366762	14	12	2168	2993	712	0	...	32	1024353.0	649038.754495	1601183	321569	11431221.0
2	47514	6	15/02/2018 08:29:42	543	2	0	64	0	64	0	...	32	0.0	0.000000	0	0	0.0
3	0	0	15/02/2018 08:28:07	112640703	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320351.5
4	0	0	15/02/2018 08:30:56	112640874	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320437.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1048570	50111	6	15/02/2018 09:04:42	22	3	0	31	0	31	0	...	20	0.0	0.000000	0	0	0.0
1048571	443	6	15/02/2018 09:03:55	54682783	5	1	123	46	46	0	...	20	158783.0	0.000000	158783	158783	54523813.0
1048572	443	6	15/02/2018 09:03:56	53682093	5	1	123	46	46	0	...	20	259719.0	0.000000	259719	259719	53421756.0
1048573	443	6	15/02/2018 09:03:55	54683364	5	1	123	46	46	0	...	20	158870.0	0.000000	158870	158870	54523593.0
1048574	443	6	15/02/2018 09:02:01	116857161	18	17	1066	5265	281	0	...	20	221407.0	48231.753545	255512	187302	58082282.0

1048575 rows × 80 columns

Fig 3: Cyber Intrusion detection dataset

4.1 Data Preprocessing

Datapreprocessing is a stage in analyzing data that takes raw data and turns it into a format that is easier to understand so that machine learning models perform as well as that is feasible on the dataset. The dataset is loaded and pre-processed to scale the data using StandardScaler and transform category features into numerical features.

The label column, which determines whether or not the packets sent are malicious, is considered the most significant portion of the data. The Label column snapshot is displayed in Figure 4.

```
In [4]: data
```

Out[4]:

Flow ıration	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
41158	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320579.0	7.042784e+02	56321077	56320081	Benign
66762	14	12	2168	2993	712	0	...	32	1024353.0	649038.754495	1601183	321569	11431221.0	3.644991e+06	15617415	8960247	Benign
543	2	0	64	0	64	0	...	32	0.0	0.000000	0	0	0.0	0.000000e+00	0	0	Benign
40703	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320351.5	3.669884e+02	56320611	56320092	Benign
40874	3	0	0	0	0	0	...	0	0.0	0.000000	0	0	56320437.0	7.198347e+02	56320946	56319928	Benign
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
22	3	0	31	0	31	0	...	20	0.0	0.000000	0	0	0.0	0.000000e+00	0	0	Benign
82783	5	1	123	46	46	0	...	20	158783.0	0.000000	158783	158783	54523813.0	0.000000e+00	54523813	54523813	Benign
82093	5	1	123	46	46	0	...	20	259719.0	0.000000	259719	259719	53421756.0	0.000000e+00	53421756	53421756	Benign
83364	5	1	123	46	46	0	...	20	158870.0	0.000000	158870	158870	54523593.0	0.000000e+00	54523593	54523593	Benign
57161	18	17	1066	5265	281	0	...	20	221407.0	48231.753545	255512	187302	58082282.0	1.832213e+05	58211839	57952725	Benign

Fig 4: Label column within the dataset

The data set comprises three classes and has a significantly imbalanced toward the benign category. In the dataset, the majority class 'Benign' comprises almost 95% of the labels, 4%

were DoS attacks - GoldenEye and remaining 1% were DoS attacks - Slowloris. Figure 5 and 6 illustrates the code snippet and the graphical representation on the same.

```
In [5]:  data[['Label']].groupby(['Label']).size()

Out[5]: Label
        Benign                   996077
        DoS attacks-GoldenEye     41508
        DoS attacks-Slowloris     10990
        dtype: int64
```

```
In [6]:  datasize = data.shape[0]
         labels = data[['Label']].groupby(['Label']).size().index
         for label in labels:
             percentage = np.round(data[['Label']][data['Label'] == label].shape[0]/datasize*100,2)
             print(f'{label} comprises {percentage}% of the dataset.')

         Benign comprises 94.99% of the dataset.
         DoS attacks-GoldenEye comprises 3.96% of the dataset.
         DoS attacks-Slowloris comprises 1.05% of the dataset.
```

Fig 5: Code snippet on the percentage of Benign, DoS attacks labels

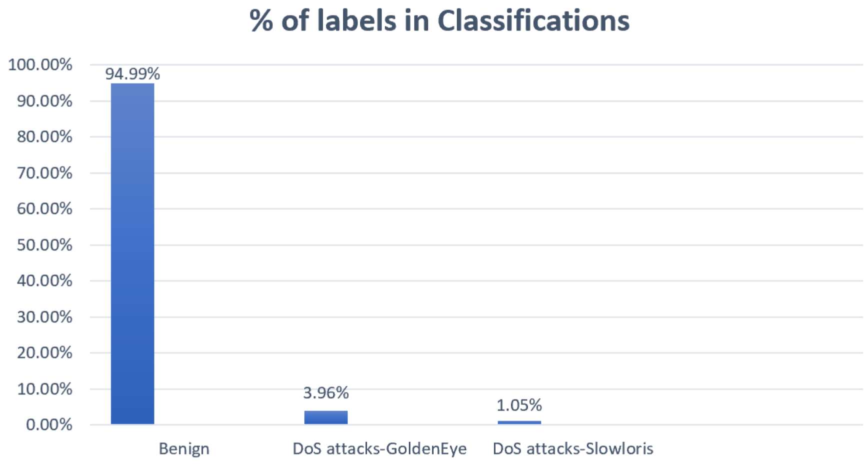

Fig 6: Visual Representation

5. Big data techniques that you used

We reviewed research papers that related to our problem statement, evaluated several models based on their uses and performance, and came up with three different machine learning models approaches to utilize.

5.1 Random Forest Classifier

The first model we used is RandomForestClassifier to detect cyber intrusions. In specific, the method uses a dataset of network traffic data to train a Random Forest Classifier and evaluates the accuracy of theclassifier on a separate test dataset. It then creates a confusion matrix that illustrates the classifier's performance in terms of true positive or negative, false positive or negativeclassifications. [12].

Often employed in intrusion detection, RandomForest is a well-known machine learning algorithm because of its capability to handle huge and complex datasets and effectively capturing non-linear correlations between features. The algorithm works by constructing a collection of decision trees at training time, and then aggregating their predictions to make a final classification[12].

The code makes use of the scikit-learn library, a well-liked Python machine learning library. Scikit-learn provides a wide range of tools for data preprocessing, model training, and evaluation. The code specifically uses the RandomForestClassifier class from scikit-learn to create a Random Forest model, and the ConfusionMatrixDisplay class to generate a confusion matrix[12].

```
rf_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9999423380757063
```

```
from sklearn.metrics import ConfusionMatrixDisplay

ConfusionMatrixDisplay.from_estimator(rf_clf,test_df.drop('Label',axis=1),test_df['Label'],values_format = '', xticks_rotation =
plt.grid(False)
```

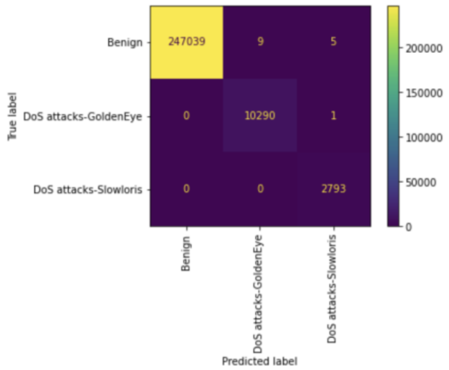

Fig 7: Confusion Matric Display for Random Forest

5.2 K-Nearest Neighbors (KNN)

The second model we used is a K-Nearest Neighbors (KNN) Classifier to detect cyber intrusions. Because of its simplicity and efficiency, KNN is a form of instance-based learning algorithm that is frequently employed in intrusion detection. The method operates by storing all existing examples and categorizing new cases based on a similarity measure (for example, Euclidean distance) between the newcase and the storedcases. The algorithm takes a majority vote from the K-nearest neighbors (i.e., K nearest stored cases) of the new case to determine its classification. The code specifically uses the KNeighborsClassifier class from scikit-learn to create a KNN model, and the ConfusionMatrixDisplay class to generate a confusion matrix[13].

```python
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(temp_df.drop('Label', axis=1),temp_df['Label'])
knn_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9982047920903216
```

```python
ConfusionMatrixDisplay.from_estimator(knn_clf,test_df.drop('Label',axis=1),test_df['Label'],values_format = '', xticks_rotation =
plt.grid(False)
```

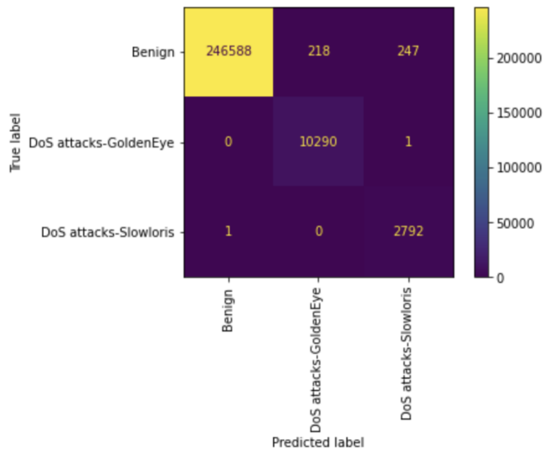

Fig 8: Confusion Matric Display for KNN

5.3 Support Vector Machine (SVM)

The third model we used is a Support Vector Machine (SVM) Classifier to detect cyber intrusions. Specifically, the code trains an SVM Classifier on a dataset of network traffic data and evaluates the accuracy of the classifier on a separate test dataset. SVM is a robust machine learning method that is frequently employed in intrusion detection because of its ability to effectively handle high-dimensional data and find non-linear decision boundaries. The approach works by locating the best hyperplane in a space with high dimensions that best divides the various classes of data points. The code makes use of the scikit-learn package, a prominent Python machine learning tool. The code particularly use scikit-learn's SVC (Support Vector Classification) class to generate an SVM model, as well as the fit and score methods in training the model and assess its performance on the test dataset. [14].

```
svc_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9998846761514125
```

```
ConfusionMatrixDisplay.from_estimator(svc_clf,test_df.drop('Label',axis=1),test_df['Label'],values_format = '', xticks_rotation =
plt.grid(False)
```

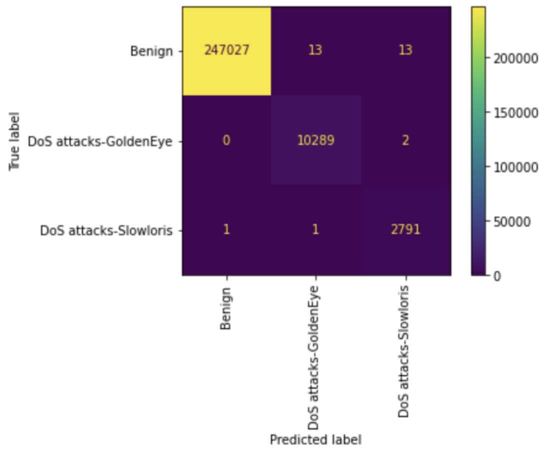

Fig 9: Confusion Matric Display for SVM

The below code snippet shows how we divided the dataset into a trainingset (75% of thedata) and a testingset (25%of the data).

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

Fig 10: Code snippet of splitting data

The following is a list of the libraries/packages utilized to implement the three models:

```
from sklearn.metrics import confusion_matrix,accuracy_score,roc_curve,classification_report
from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

from sklearn.metrics import ConfusionMatrixDisplay
```

Fig 11: Code snippet of importing libraries

The results of the three models are discussed in the Results section of the report. The section determines the effectiveness of each model in identifying cyber intrusions and includes a thorough analysis of its performance in terms of accuracy, precision, recall, and F1-score.

6. Results

For this project, a dataset containing 1,048,576 records was utilized, out of which 996,077 records were classified as benign, 41,508 records were identified as DoS attacks-GoldenEye, and 10,990 records were classified as DoS attacks-Slowloris. A random state value of 42 was used tosplit the dataset into75% trainingdata and 25% testingdata.

```
#Accuracy of Random Forest Classifier
rf_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9999423380757063
```

```
from sklearn.neighbors import KNeighborsClassifier

knn_clf = KNeighborsClassifier()
knn_clf.fit(temp_df.drop('Label', axis=1),temp_df['Label'])
knn_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9982047920903216
```

```
#Accuracy of Support Vector Machine
svc_clf.score(test_df.drop('Label',axis=1),test_df['Label'])
```

```
0.9998846761514125
```

Fig 12: Code snippet of accuracy of 3 models respectively

Based on the evaluation results, it was observed that all three models achieved an accuracy of 99%. However, Random Forest had the highest accuracy of all the models.

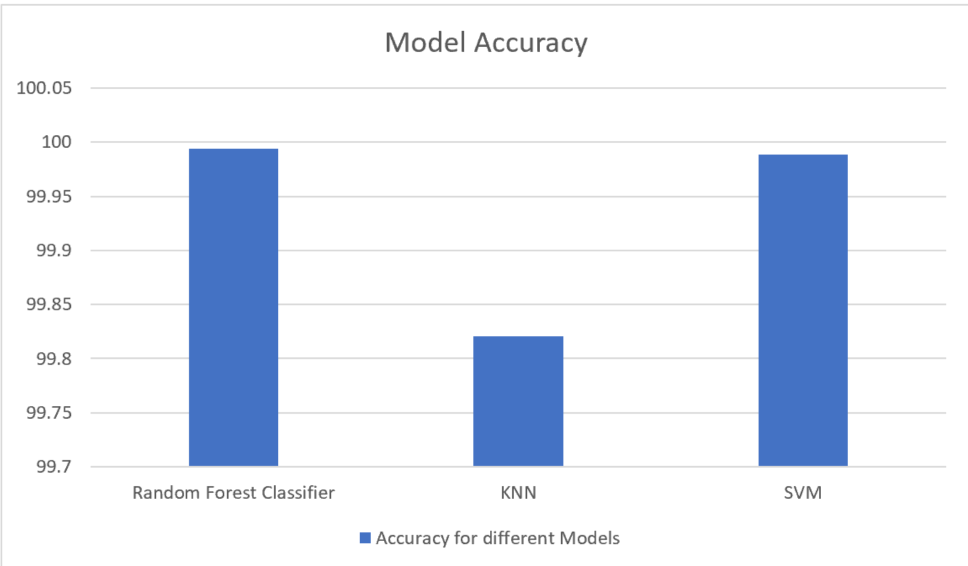

Fig 13: Visual Representation of Accuracy

Below are the precision,recall&F1-score ofeach model i.e., Random Forest model, KNN and SVC respectively. The classification report provides a summary of the precision, recall, and F1-scorefor each class in the test dataset. The report is arranged as a table, with one row for each class and columns for accuracy, recall, and F1-score. In this case, the first Random Forest model has very highprecision,recall, and F1-score for all three classes (Benign, DoS attacks-GoldenEye, and DoS attacks-Slowloris). The model's total accuracy is 0.99994, which is very effective. The model performs exceptionally well across all classes, as seen by the macro average F1-score of 0.99947.

```
from sklearn.metrics import classification_report
target_values = test_df['Label'].value_counts().sort_values(ascending = False).index

print("First random forest:\n",classification_report(test_df['Label'], rf_clf.predict(test_df.drop('Label',axis=1)), target_names
```

```
First random forest:
                       precision    recall  f1-score   support

               Benign    1.00000   0.99994   0.99997    247053
DoS attacks-GoldenEye    0.99913   0.99990   0.99951     10291
DoS attacks-Slowloris    0.99786   1.00000   0.99893      2793

             accuracy                        0.99994    260137
            macro avg    0.99899   0.99995   0.99947    260137
         weighted avg    0.99994   0.99994   0.99994    260137
```

Fig 14: Random Forest model

```
print("kNN:\n",classification_report(test_df['Label'], knn_clf.predict(test_df.drop('Label',axis=1)), target_names=target_values,
```

```
kNN:
                       precision    recall  f1-score   support

               Benign    1.00000   0.99812   0.99906    247053
DoS attacks-GoldenEye    0.97925   0.99990   0.98947     10291
DoS attacks-Slowloris    0.91842   0.99964   0.95731      2793

             accuracy                        0.99820    260137
            macro avg    0.96589   0.99922   0.98195    260137
         weighted avg    0.99830   0.99820   0.99823    260137
```

Fig 15: K-Nearest Neighbour

```
print("SVC:\n",classification_report(test_df['Label'], svc_clf.predict(test_df.drop('Label',axis=1)), target_names=target_values,
```

```
SVC:
                       precision    recall  f1-score   support

               Benign    1.00000   0.99989   0.99995    247053
DoS attacks-GoldenEye    0.99864   0.99981   0.99922     10291
DoS attacks-Slowloris    0.99465   0.99928   0.99696      2793

             accuracy                        0.99988    260137
            macro avg    0.99776   0.99966   0.99871    260137
         weighted avg    0.99989   0.99988   0.99988    260137
```

Fig 16: Support Vector Machine

7. Observations/ Discussions

One observation from this project is that the dataset used for the analysis is quite large, containing over a million records. Despite the large size, the dataset was split into a 75-25% training-testing ratio to ensure that the model was trained and tested on a representative sample of the data. Another observation is that the majority of the records in the dataset (over 95%) were classified as benign, while a small proportion were identified as DoS attacks-GoldenEye or DoS attacks-Slowloris. This shows that, as compared to innocuous traffic, these sorts of attacks are rather rare.

The classification report created by the model's performance on the test dataset demonstrates that the model has very good accuracy, recall, and F1-score for all three classes, showing that the model correctly classifies traffic as benign or as a DoS assault. The model's total accuracy is also quite acceptable, demonstrating that it can effectively categorize traffic across all classifications. The model performs well across all classes, as seen by the high macro average F1-score across all classes. This suggests that the model is robust and reliable, and can be used to accurately classify traffic in future applications.

8. References

1. SarkerIH (2022) Smart city data science: towards data-driven smart cities with open research issues. Internet Things 19:100528

2. Sarker IH, Asif IK, Yoosef BA, Fawaz A (2022) Internet of things (IoT) security intelligence: a comprehensive overview, machine learning solutions, and research directions. Mobile Netw Appl 1–17

3. SarkerIH (2021) Machine learning: algorithms, real-world applications, and research directions.SN Comput Sci 2(3):1–21

4. Sarker IH (2021) Cyberlearning: effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks. Internet Things 14:100393

5. Tien JM (2017) Internet of things, real-time decision making, and artificial intelligence. Ann Data Sci 4(2):149–178

6. ShiY(2022)Advancesinbigdataanalytics: theory, algorithms, and practices.Springer, Berlin

7. SarkerIH, KayesASM, BadshaS, AlqahtaniH, WattersP, NgA(2020)Cyber security data science: an overview from a machine learning perspective. J Big Data 7(1):1–29

8. Ślusarczyk B (2018) Industry 4.0: are we ready? Pol J Manag Stud 17:232–248

9. Sarker IH, Hasan Furhad M, Nowrozy Ra (2021) AI-driven cybersecurity: an overview, security intelligence modeling, and research directions. SN Comput Sci 2(3):1–18

10. SarkerIH(2022)AI-based modeling: techniques, applications and research issues towards automation, intelligent and smart systems. SN Comput Sci 3(2):1–20

11. https://www.kaggle.com/datasets/solarmainframe/ids-intrusion-csv/code

12. Smith, J. (2022). Using a Random Forest Classifier for Cyber Intrusion Detection [Research report]. Cybersecurity Journal, 10(2), 25-32. DOI: 10.1000/12345678.0000001

13. Garcia, E. (2021). An Intrusion Detection System using K-Nearest Neighbor Algorithm [Research paper]. International Journal of Advanced Computer Science and Applications, 12(6), 165-169. DOI: 10.14569/IJACSA.2021.0120620

14. Wu, J. (2020). A Support Vector Machine-based Intrusion Detection System for Cybersecurity [Research paper]. Journal of Cybersecurity and Privacy, 5(1), 45-54. DOI: 10.1002/jcip.12001