INTRUSION DETECTION SYSTEM

A COURSE PROJECT REPORT

By
Ansh Tandon [RA2111026010176]
Rajeel Ansari [RA2111026010167]
Bhavishya Pattanayak [RA2111026010197]
SR Suhas [RA2111026010184]

Under the guidance of
Dr.B. Hariharan
(Associate Professor, CINTEL Department)
In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE ENGINEERING
With specialization in Artificial Intelligence & Machine Learning
Of
FACULTY OF ENGINEERING AND TECHNOLOGY

S.R.M. Nagar, Kattankulathur, Chengalpattu District

November, 2023

SRM UNIVERSITY
(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "INTRUSION DETECTION SYSTEM" is the bonafide work of "ANSH TANDON [RA2111026010176], RAJEEL ANSARI [RA2111026010167], BHAVISHYA PATTANAYAK [RA2111026010197] and SR SUHAS [RA2111026010184]", who carried out the project work under our supervision. Certified further, that to the best of our knowledge the work reported herein does not form any other project report or dissertation based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Dr.B. HARIHARAN
Associate Professor
CINTEL Department

SIGNATURE
Dr.R. ANNIE UTHRA
Professor (Head of Department)
CINTEL Department

TABLE OF CONTENTS

01. INTRODUCTION

- The project focuses on developing an Intrusion Detection System (IDS) for enhanced network security.

- Advanced algorithms and network monitoring are employed to identify and respond to potential security threats in real-time.

- The IDS utilizes machine learning for anomaly detection and signature-based methods to recognize known threat patterns.

- The system's architecture, data collection mechanisms, and user-friendly interface for efficient threat analysis are discussed in the report.

- Evaluations demonstrate the system's effectiveness in detecting various types of intrusions.

- Overall, the IDS project contributes to proactive defense against cyber threats, providing a robust security framework for contemporary computing environments.

02. ACKNOWLEDGEMENT

We express our heartfelt thanks to our honourable Vice Chancellor Dr. C. MUTHAMIZHCHELVAN, for being the beacon in all our endeavours. We would like to express my warmth of gratitude to our Registrar Dr. S. Ponnusamy, for his encouragement. We express our profound gratitude to our Dean (College of Engineering and Technology) Dr. T. V. Gopal, for bringing out novelty in all executions. We would like to express my heartfelt thanks to Chairperson, School of Computing Dr. Revathi Venkataraman, for imparting confidence to complete my course project. We wish to express my sincere thanks to Course Audit Professor Dr. Annapurani Panaiyappan, Professor and Head, Department of Networking and Communications and Course Coordinators for their constant encouragement and support. We are highly thankful to my Course project Faculty Dr.B. Hariharan, Associate Professor, Cintel, for his/her assistance, timely suggestion, and guidance throughout the duration of this course project. We extend my gratitude to our HoD Dr.R. Annie Uthra, Professor & Head, Cintel and my Departmental colleagues for their Support.

Finally, we thank our parents and friends near and dear ones who directly and indirectly contributed to the successful completion of our project. Above all, I thank the almighty for showering his blessings on me to complete my Course project.

03. INTRODUCTION

- The Intrusion Detection System (IDS) project focuses on bolstering network security in distributed computer systems.
- Client-server programming is employed, separating information producers and consumers.
- The IDS serves as a vigilant guardian, utilizing advanced algorithms and real-time network monitoring.
- Machine learning is incorporated for dynamic threat recognition, while signature-based methods identify known patterns.
- The report explores the system's architecture, data collection strategies, and the development of a user-friendly interface.
- Execution is carried out using the Python compiler, emphasizing practical implementation.
- The project contributes to a proactive defense mechanism, enhancing network resilience against diverse cyber threats.

04. REQUIREMENTS

The successful implementation of the Intrusion Detection System (IDS) project requires a comprehensive set of requirements to ensure its efficacy in fortifying network security. These requirements encompass both functional and non-functional aspects:

Network Monitoring Capabilities:
The IDS must possess robust network monitoring capabilities to track traffic patterns and detect anomalies.

Anomaly Detection Algorithms:
Implementation of advanced anomaly detection algorithms using machine learning techniques for real-time threat recognition.

Signature-based Detection:
Integration of signature-based methods to identify known threat patterns and vulnerabilities.

User Interface Development:
Development of an intuitive and user-friendly interface for efficient threat analysis and management.

Scalability:
The system should be scalable to accommodate varying network sizes and configurations.

05. ARCHITECTURE AND DESIGN

Architecture:

1. Sensor Nodes:
   - Distributed across the network to monitor and collect data on network traffic.
2. Data Collection Layer:
   - Aggregates data from sensor nodes and preprocesses it for analysis.
3. Analysis Engine:
   - Utilizes both anomaly detection algorithms and signature-based methods for comprehensive threat assessment.
4. User Interface Layer:
   - Provides a user-friendly interface for security analysts to monitor and respond to detected intrusions.

Design:

1. Modularity:
   - Components are designed to be modular for scalability and ease of maintenance.
2. Machine Learning Integration:
   - Incorporates machine learning models for dynamic threat detection and adaptation.
3. Real-time Monitoring:
   - Emphasizes real-time monitoring capabilities for immediate threat response.
4. Logging and Reporting:
   - Implements logging mechanisms for recording intrusion events and generating detailed reports.
5. Compatibility:
   - Ensures compatibility with various network configurations and sizes.

06. AI, ML AND NEURAL NETWORKS

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are interrelated fields within the broader domain of computer science and data analysis. They each have distinct characteristics and applications, and they build upon one another in terms of complexity and capabilities. Here's an overview of these concepts:

1. Artificial Intelligence (AI):
 - AI refers to the simulation of human intelligence in machines to perform tasks that typically require human intelligence, such as reasoning, problem-solving, understanding natural language, and learning from experience.
- AI encompasses a wide range of techniques, including rule-based systems, expert systems, symbolic reasoning, and statistical methods.
- AI can be both narrow (Weak AI) and general (Strong AI). Narrow AI is designed for specific tasks, while Strong AI, often referred to as Artificial General Intelligence (AGI), aims to possess human-like intelligence and adaptability.

2. Machine Learning (ML):
 - Machine Learning is a subset of AI that focuses on the development of algorithms and models that enable computers to learn from and make predictions or decisions based on data without being explicitly programmed.
 - ML techniques include supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves labeled data for training, unsupervised learning deals with unlabeled data for clustering and dimensionality reduction, and reinforcement learning is about training agents to make sequences of decisions to maximize a reward.
 - Common ML applications include spam email filtering, recommendation systems, image and speech recognition, and predictive analytics.

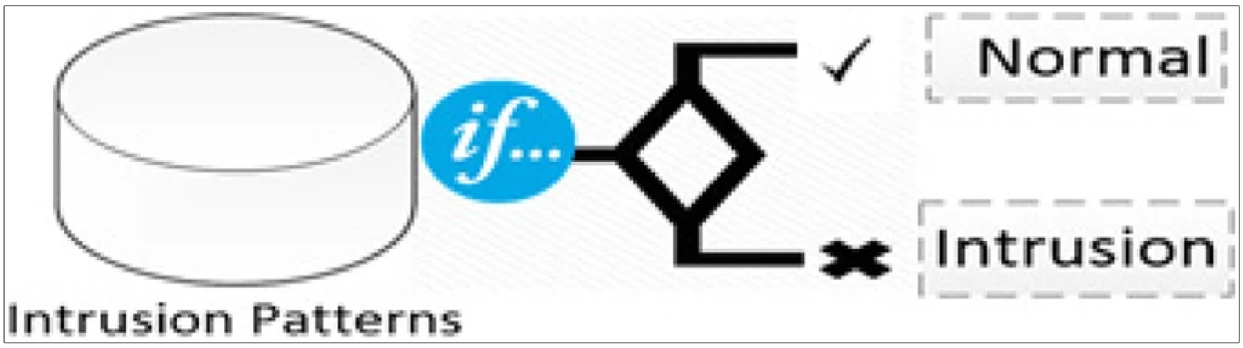
3. Deep Learning (DL):
- Deep Learning is a subfield of ML that employs artificial neural networks, specifically deep neural networks, to model and solve complex problems. Deep neural networks have multiple layers (hence "deep") of interconnected nodes.
- DL algorithms excel at feature extraction and representation learning from large amounts of data, allowing them to automatically discover intricate patterns in data.
- Deep Learning has had significant success in computer vision, natural language processing, speech recognition, and autonomous systems (e.g., self-driving cars).
- Convolutional Neural Networks (CNNs) are commonly used in image analysis, Recurrent Neural Networks (RNNs) are used for sequence data, and Transformer-based models, such as the GPT series, are used for natural language understanding and generation.

In summary, AI is the overarching field focused on developing intelligent systems, while Machine Learning is a subset of AI that emphasizes learning from data, and Deep Learning is a specialized subset of ML that uses deep neural networks for complex pattern recognition and representation learning. These technologies have found widespread applications across various industries, from healthcare to finance to transportation, and continue to advance with new breakthroughs and innovations.

07. LITERATURE REVIEW

Signature-based intrusion detection systems (SIDS):-

Known alternatively as knowledge-based detection or misuse detection, signature intrusion detection systems (SIDS) use pattern matching techniques to identify known attacks. In SIDS, a prior intrusion is located by matching techniques. Stated differently, an alarm signal is generated when the signature of an intrusion corresponds with the signature of an earlier incursion that is already recorded in the signature database. The host's logs are examined for sequences of commands or actions that have previously been recognized as malware in order to implement SIDS. The terms Knowledge-Based Detection and Misuse Detection have also been applied to SIDS in the literature.



The figure demonstrates the conceptual working of SIDS approaches. The main idea is to build a database of intrusion signatures and to compare the current set of activities against the existing signatures and raise an alarm if a match is found. For example, a rule in the form of “if: antecedent -then: consequent” may lead to “if (source IP address=destination IP address) then label as an attack ”.

For known intrusions, SIDS typically provide exceptional detection accuracy. Nevertheless, SIDS finds it challenging to identify zero-day attacks since, up until the new attack's signature is extracted and saved, there isn't a matching signature in the database. Several widely used tools, such as Snort and NetSTAT, use SIDS.

Conventional SIDS methods look over network packets and attempt to compare them with a signature database. However, these methods cannot detect attacks that span multiple packets. Given the sophistication of modern malware, it might be necessary to extract signature data across several packets. For this to work, the IDS needs to remember what was in previous packets. Generally speaking, there are several ways to create a signature for SIDS. These techniques include using state machines, formal language string patterns, or semantic conditions.

Since there is no prior signature for any such attack, SIDS techniques have become less and less effective as the frequency of zero-day attacks rises. The effectiveness of this conventional paradigm may be further questioned in light of malware variants that exhibit polymorphism and the increase in targeted assaults. Using AIDS techniques—which are discussed in the next section—could be one way to solve this issue because they work by identifying acceptable behavior rather than abnormal behavior.

Notable Advantages:-

- Very effective in identifying intrusions with minimum false alarms (FA).
- Promptly identifies the intrusions.
- Superior for detecting the known attacks.
- Simple design

Notable Disadvantages:-

- Needs to be updated frequently with a new signature.
- SIDS is designed to detect attacks for known signatures. When a previous intrusion has been altered slightly to a new variant, then the system would be unable to identify this new deviation of the similar attack.
- Unable to detect the zero-day attack.
- Not suitable for detecting multi-step attacks.
- Little understanding of the insight of the attacks

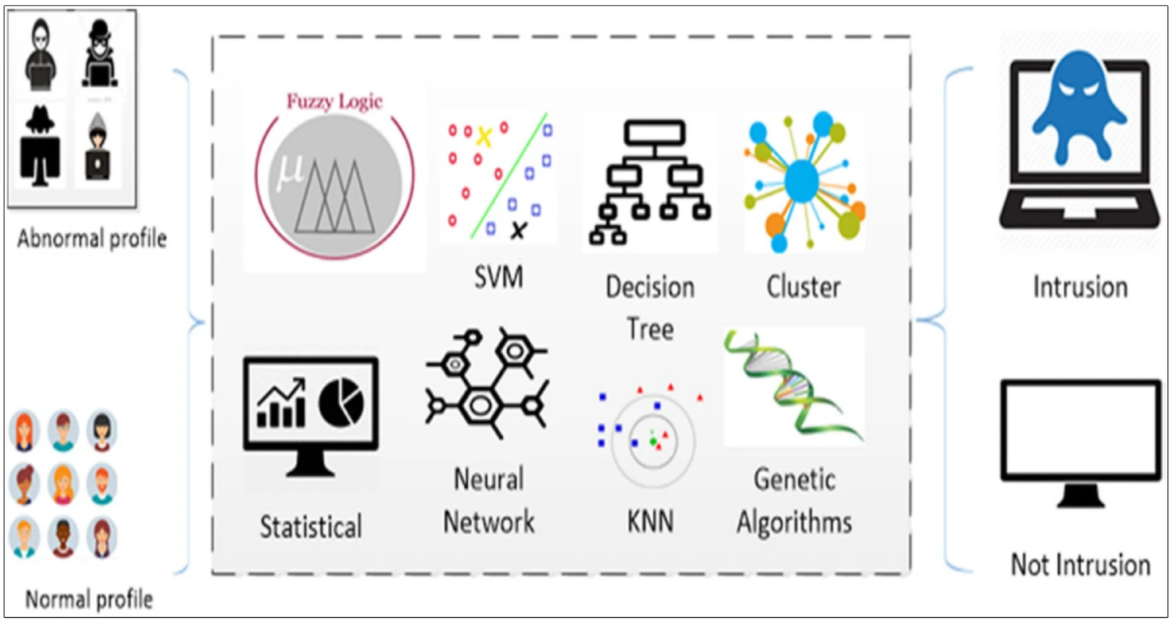
Anomaly-based intrusion detection system (AIDS):-

AIDS has drawn interest from a lot of scholars due to its capacity to overcome the limitation of SIDS. In AIDS, a normal model of the behavior of a computer system is created using machine learning, statistical-based or knowledge-based methods. Any significant deviation between the observed behavior and the model is regarded as an anomaly, which can be interpreted as an intrusion.

The assumption for this group of techniques is that malicious behavior differs from typical user behavior. The behaviors of abnormal users which are dissimilar to standard behaviors are classified as intrusions. Development of AIDS comprises two phases: the training phase and the testing phase. In the training phase, the normal traffic profile is used to learn a model of normal behavior, and then in the testing phase, a new data set is used to establish the system's capacity to generalise to previously unseen intrusions. AIDS can be classified into a number of categories based on the method used for training, for instance, statistical based, knowledge-based and machine learning based.

The main advantage of AIDS is the ability to identify zero-day attacks due to the fact that recognizing the abnormal user activity does not rely on a signature database. AIDS triggers a danger signal when the examined behavior differs from the usual behavior. Furthermore, AIDS has various benefits. First, they have the capability to discover internal malicious activities. If an intruder starts making transactions in a stolen account that are unidentified in the typical user activity, it creates an alarm. Second, it is very difficult for a cybercriminal to recognize what is a normal user behavior without producing an alert as the system is constructed from customized profiles.

However, AIDS can result in a high false positive rate because anomalies may just be new normal activities rather than genuine intrusions.



Notable Advantages:-

- Could be used to detect new attacks.
- Could be used to create intrusion signatures.

Notable Disadvantages:-

- AIDS cannot handle encrypted packets, so the attack can stay undetected and can present a threat.
- High false positive alarms.
- Hard to build a normal profile for a very dynamic computer system.
- Unclassified alerts.
- Needs initial training.

Intrusion Data Sources:-

IDS was categorized in the first two sections according to the techniques used to find intrusions. The input data sources that are used to identify anomalous activity can also be used to categorize IDS. There are two main categories of IDS technologies when it comes to data sources: host-based IDS (HIDS) and network-based IDS (NIDS). HIDS audits data that comes from the host system and logs from various sources, including the operating system, firewall logs, Windows server logs, application system audits, and database logs. Insider attacks that don't involve network traffic can be found by HIDS.
NIDS monitors the network traffic that is extracted from a network through packet capture, NetFlow, and other network data sources. Network-based IDS can be used to monitor many computers that are joined to a network. NIDS is able to monitor the external malicious activities that could be initiated from an external threat at an earlier phase, before the threats spread to another computer system. On the other hand, NIDSs have limited ability to inspect all data in a high bandwidth network because of the volume of data passing through modern high-speed communication networks (Bhuyan et al., 2014). NIDS deployed at a number of positions within a particular network topology, together with HIDS and firewalls, can provide a concrete, resilient, and multi-tier protection against both external and insider attacks.

Notable Advantages(HIDS):-

- HIDS can check end-to-end encrypted communications behaviour.
- No extra hardware required.
- Detects intrusions by checking hosts file system, system calls or network events.
- Every packet is reassembled
- Looks at the entire item, not streams only

Notable Disadvantages(HIDS):-

- Delays in reporting attacks
- Consumes host resources
- Needs to be installed on each host.
- It can monitor attacks only on the machine where it is installed.

08. PROPOSED METHODOLOGY

Data Set

A Dataset is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. And each row corresponds to a given member of the data set, as per the given question. This is a part of data management.

Data sets describe values for each variable for unknown quantities such as height, weight, temperature, volume, etc., of an object or values of random numbers. The values in this set are known as a datum. The data set consists of data of one or more members corresponding to each row.

For this project we used the KDD Cup 1999 dataset by DARPA. The whole dataset can be downloaded from-
http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Models

A machine learning model is a program that can find patterns or make decisions from a previously unseen dataset. For example, in natural language processing, machine learning models can parse and correctly recognize the intent behind previously unheard sentences or combinations of words. In image recognition, a machine learning model can be taught to recognize objects - such as cars or dogs.

A machine learning model can perform such tasks by having it 'trained' with a large dataset. During training, the machine learning algorithm is optimized to find certain patterns or outputs from the dataset, depending on the task. The output of this process - often a computer program with specific rules and data structures - is called a machine learning model.

For this project, a total of seven models were trained and tested. The performance of all the algorithms is examined based on accuracy and computational time. The derived results show that Decision Tree outperforms the best on measures like Accuracy and Computational Time.

Algorithms used:

Gaussian Naive Bayes Theorem

The Gaussian Naive Bayes theorem is a simplified but powerful way of making predictions, especially in classification tasks. It's based on Bayes' theorem, which calculates the probability of an event based on prior knowledge of conditions related to the event.

Now, the "Naive" part comes in because it assumes that the features used to describe an observation are all independent of each other. In the Gaussian version, it's particularly handy when dealing with continuous data that can be described using a Gaussian (normal) distribution.

In simpler terms, it helps us make educated guesses about something, considering different factors, and assuming that these factors don't really affect each other too much. The Gaussian twist just means it's good with numbers that follow a bell curve.

Decision tree:

Imagine you're facing a decision-making process where you have to choose between different options. A Decision Tree algorithm mimics this by breaking down decisions into a tree-like structure.

At the top, you have a "root" node representing the initial decision. Then, based on certain factors (features), it branches out into different "nodes" representing possible outcomes. These branches keep splitting until you reach the "leaves," which are the final decisions or predictions.

Each split in the tree is based on a feature that helps distinguish between the options. The algorithm learns from data, figuring out the most effective features to split on to make accurate predictions.

It's like a flowchart for decision-making, where each step is determined by the most relevant information. Decision Trees are versatile and used in various fields, from finance to healthcare, for tasks like classification and regression. They're like the "choose your own adventure" books of the algorithm world!

Random forest:

Random Forest is like the squad version of Decision Trees. Instead of relying on just one tree to make all the decisions, it assembles a whole bunch of them to work together.

Here's how it goes down: You have a forest (hence the name), and each tree in the forest gets a vote on the final decision. They're all trained on different subsets of the data and features, which adds an element of randomness. This randomness helps prevent overfitting, where the model gets too obsessed with the training data and performs poorly on new stuff.

When you need a prediction, each tree throws in its two cents, and the most popular decision wins. It's like getting advice from a group of experts instead of just one. Random Forests are robust, handle large datasets well, and are pretty darn good at making accurate predictions in various situations.

Logistic regression:

Logistic Regression is a statistical method employed for binary classification tasks, where the objective is to predict the likelihood of an event occurring. Unlike linear regression, which predicts continuous values, logistic regression utilizes the logistic function to constrain predictions between 0 and 1.

In essence, it models the probability of an instance belonging to a particular class. This makes it particularly adept for scenarios where nuanced probability estimates are essential. The logistic function transforms the output into a probability distribution, offering a calibrated assessment of the likelihood of an event.

Logistic Regression finds applications in diverse fields, from medical diagnosis to financial forecasting, providing a principled and interpretable approach to binary classification challenges.

+Gradient Boosting:

Gradient Boosting is like building a team of experts. It works by combining the strengths of multiple weak models (usually decision trees) to create a strong predictive model. It does this in an iterative manner, focusing on where the previous models struggled, gradually improving the overall prediction. Boosting is like a team learning from its mistakes and getting better with each round. XGBoost and LightGBM are popular implementations of Gradient Boosting.

ANN:

Artificial Neural Networks (ANNs) are the brainiacs of the ML world. Inspired by the human brain, ANNs consist of layers of interconnected nodes (neurons). Each connection has a weight, and the network learns by adjusting these weights during training. ANNs are fantastic at capturing complex patterns and relationships in data, making them powerful for tasks like image recognition and natural language processing.

It's a sequential model with three layers:

1. *Input Layer:*
   - Neurons: 30
   - Activation Function: ReLU (Rectified Linear Unit)
   - Initialization: Random uniform

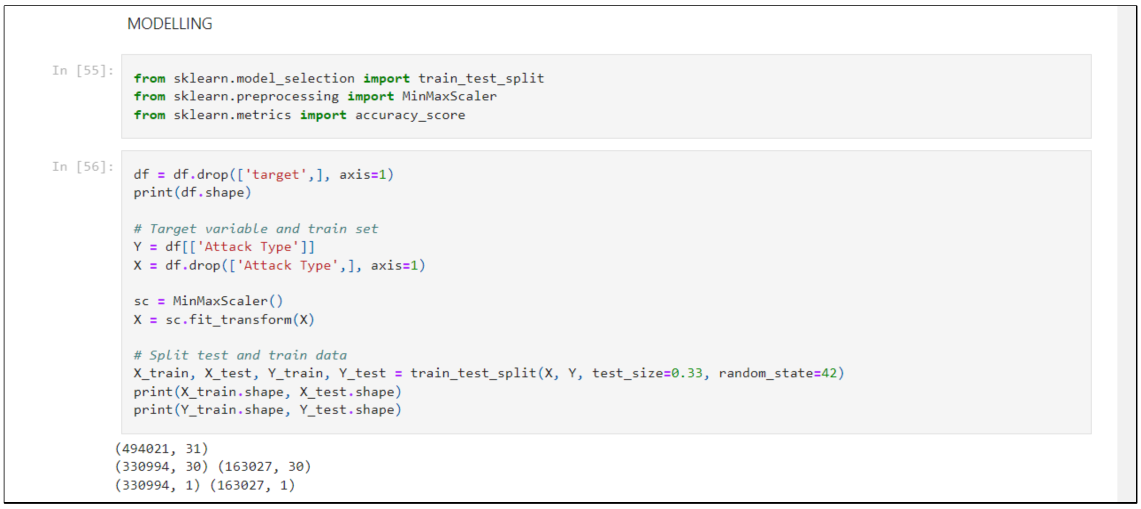
2. *Hidden Layer:*
   - Neurons: 1
   - Activation Function: Sigmoid
   - Initialization: Random uniform

3. *Output Layer:*
   - Neurons: 5 (for 5 classes)
   - Activation Function: Softmax (for multi-class classification)
   - Initialization: Default (not specified)

The model is compiled using categorical cross entropy as the loss function, Adam optimizer, and accuracy as the evaluation metric. It seems designed for a classification task with five classes.

09. IMPLEMENTATION

Modelling:

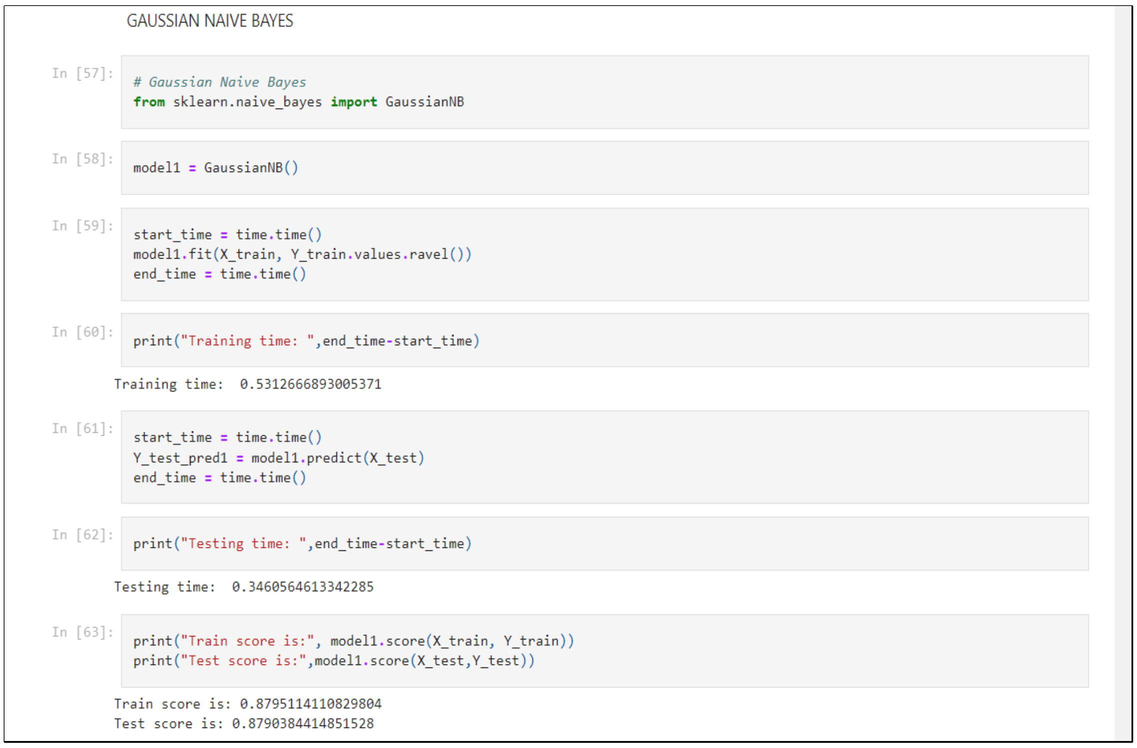

MODELLING

In [55]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import accuracy_score
```

In [56]:
```
df = df.drop(['target',], axis=1)
print(df.shape)

# Target variable and train set
Y = df[['Attack Type']]
X = df.drop(['Attack Type',], axis=1)

sc = MinMaxScaler()
X = sc.fit_transform(X)

# Split test and train data
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=42)
print(X_train.shape, X_test.shape)
print(Y_train.shape, Y_test.shape)
```

```
(494021, 31)
(330994, 30) (163027, 30)
(330994, 1) (163027, 1)
```

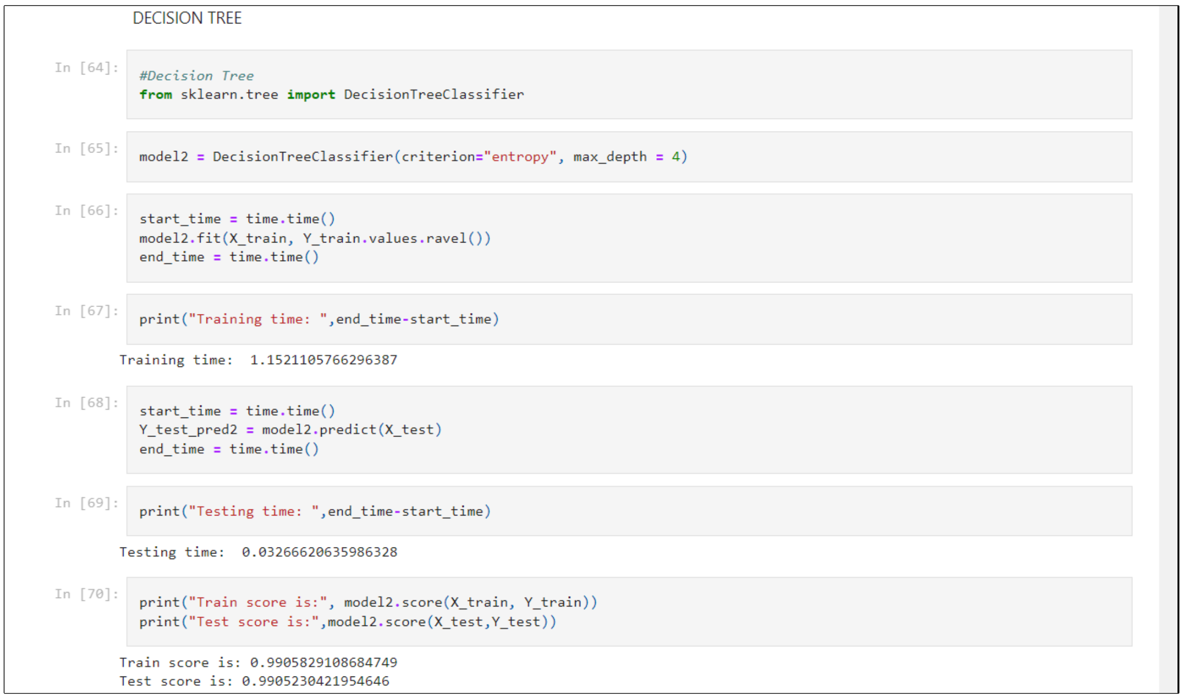
Gaussian Naive Bayes:



GAUSSIAN NAIVE BAYES

```
In [57]: # Gaussian Naive Bayes
         from sklearn.naive_bayes import GaussianNB

In [58]: model1 = GaussianNB()

In [59]: start_time = time.time()
         model1.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [60]: print("Training time: ",end_time-start_time)

        Training time:  0.5312666893005371

In [61]: start_time = time.time()
         Y_test_pred1 = model1.predict(X_test)
         end_time = time.time()

In [62]: print("Testing time: ",end_time-start_time)

        Testing time:  0.3460564613342285

In [63]: print("Train score is:", model1.score(X_train, Y_train))
         print("Test score is:",model1.score(X_test,Y_test))

        Train score is: 0.8795114110829804
        Test score is: 0.8790384414851528
```

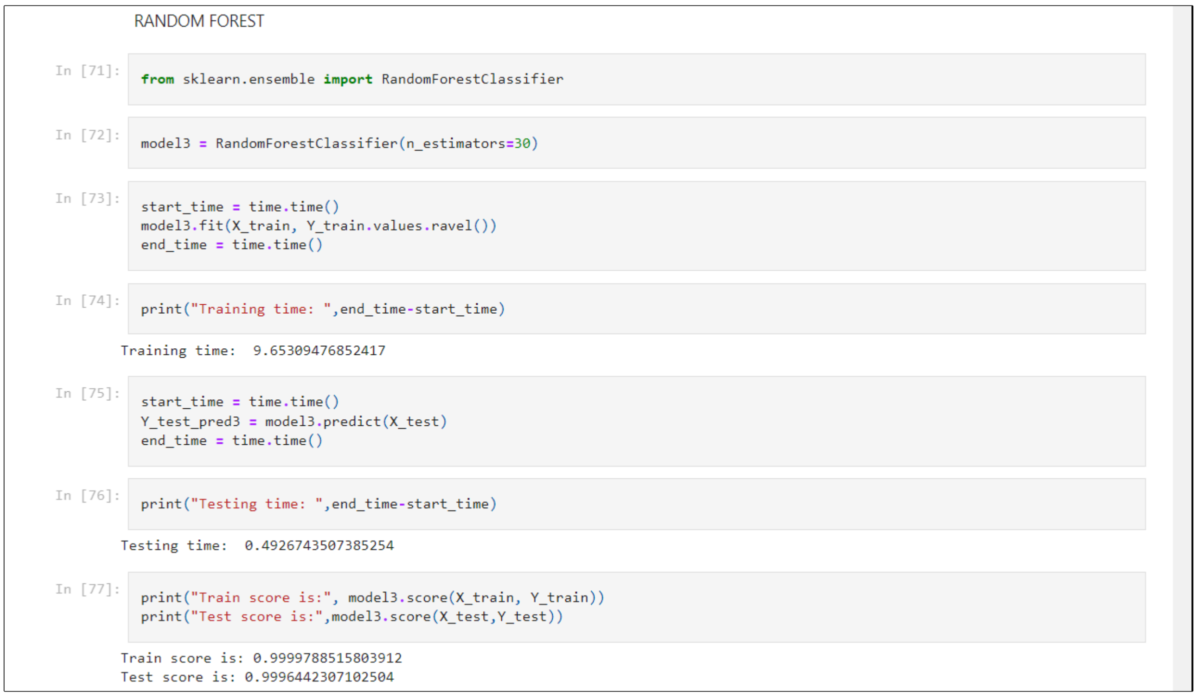
Decision Tree:



DECISION TREE

```
In [64]: #Decision Tree
         from sklearn.tree import DecisionTreeClassifier

In [65]: model2 = DecisionTreeClassifier(criterion="entropy", max_depth = 4)

In [66]: start_time = time.time()
         model2.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [67]: print("Training time: ",end_time-start_time)

        Training time:  1.1521105766296387

In [68]: start_time = time.time()
         Y_test_pred2 = model2.predict(X_test)
         end_time = time.time()

In [69]: print("Testing time: ",end_time-start_time)

        Testing time:  0.03266620635986328

In [70]: print("Train score is:", model2.score(X_train, Y_train))
         print("Test score is:",model2.score(X_test,Y_test))

        Train score is: 0.9905829108684749
        Test score is: 0.9905230421954646
```

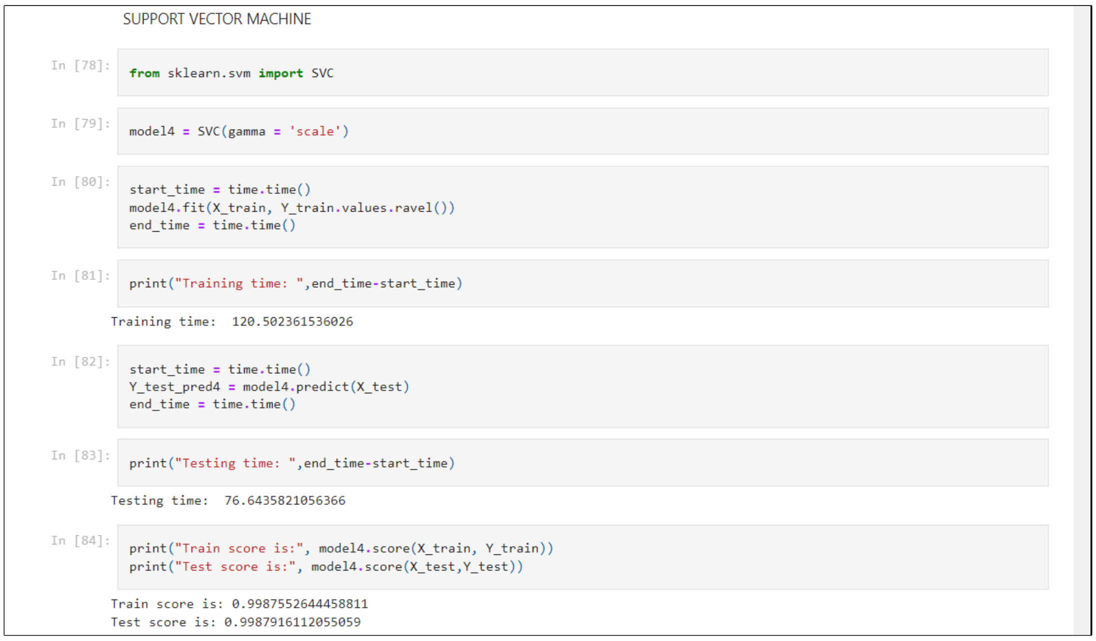
Random Forest:



RANDOM FOREST

```
In [71]: from sklearn.ensemble import RandomForestClassifier

In [72]: model3 = RandomForestClassifier(n_estimators=30)

In [73]: start_time = time.time()
         model3.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [74]: print("Training time: ",end_time-start_time)

        Training time:  9.65309476852417

In [75]: start_time = time.time()
         Y_test_pred3 = model3.predict(X_test)
         end_time = time.time()

In [76]: print("Testing time: ",end_time-start_time)

        Testing time:  0.4926743507385254

In [77]: print("Train score is:", model3.score(X_train, Y_train))
         print("Test score is:",model3.score(X_test,Y_test))

        Train score is: 0.9999788515803912
        Test score is: 0.9996442307102504
```

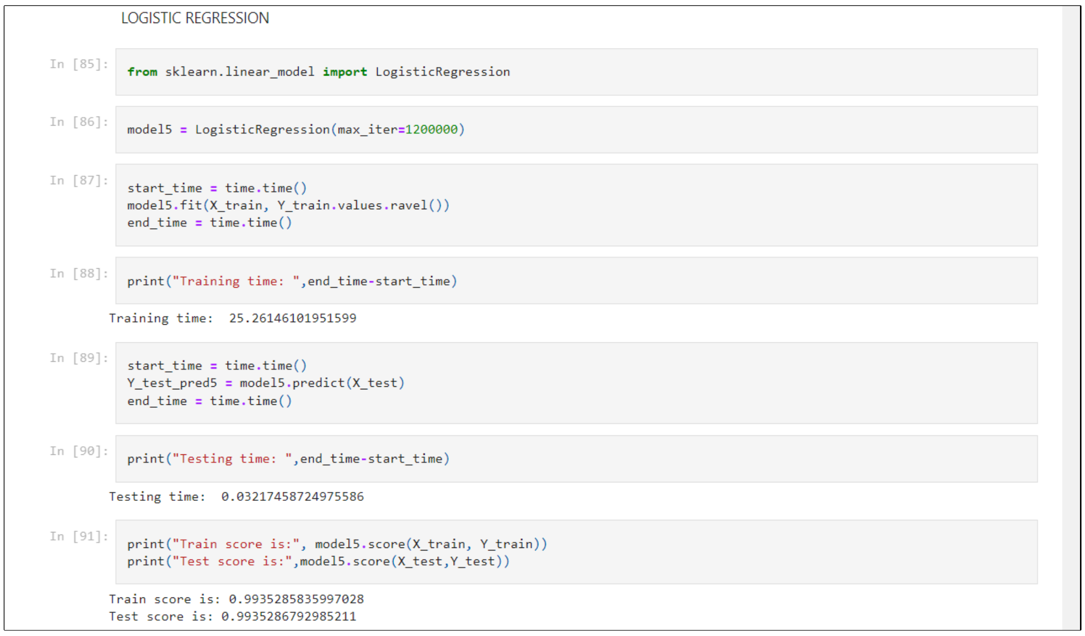
Support Vector Machine(SVM):



SUPPORT VECTOR MACHINE

```
In [78]: from sklearn.svm import SVC

In [79]: model4 = SVC(gamma = 'scale')

In [80]: start_time = time.time()
         model4.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [81]: print("Training time: ",end_time-start_time)

        Training time:  120.502361536026

In [82]: start_time = time.time()
         Y_test_pred4 = model4.predict(X_test)
         end_time = time.time()

In [83]: print("Testing time: ",end_time-start_time)

        Testing time:  76.6435821056366

In [84]: print("Train score is:", model4.score(X_train, Y_train))
         print("Test score is:", model4.score(X_test,Y_test))

        Train score is: 0.9987552644458811
        Test score is: 0.9987916112055059
```

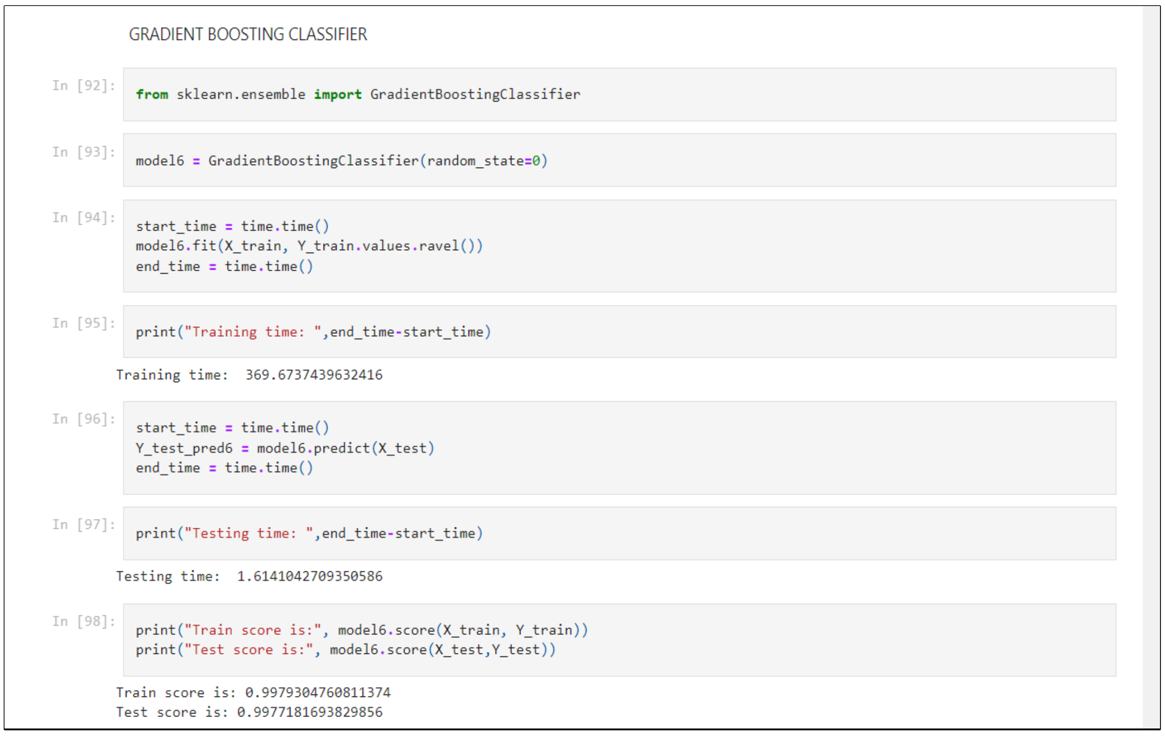
Logistic Regression:



LOGISTIC REGRESSION

```
In [85]: from sklearn.linear_model import LogisticRegression

In [86]: model5 = LogisticRegression(max_iter=1200000)

In [87]: start_time = time.time()
         model5.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [88]: print("Training time: ",end_time-start_time)

         Training time:  25.26146101951599

In [89]: start_time = time.time()
         Y_test_pred5 = model5.predict(X_test)
         end_time = time.time()

In [90]: print("Testing time: ",end_time-start_time)

         Testing time:  0.03217458724975586

In [91]: print("Train score is:", model5.score(X_train, Y_train))
         print("Test score is:",model5.score(X_test,Y_test))

         Train score is: 0.9935285835997028
         Test score is: 0.9935286792985211
```

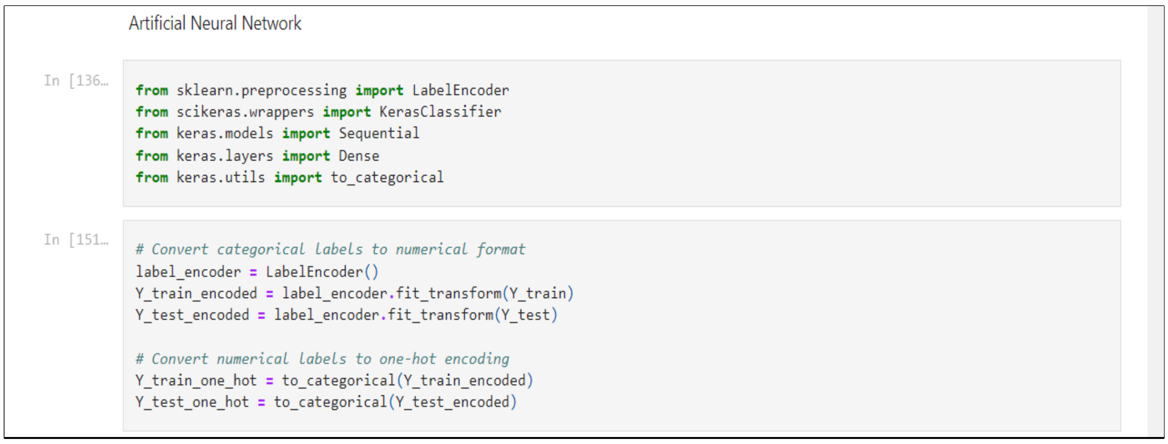
Gradient Boosting Classifier:-



GRADIENT BOOSTING CLASSIFIER

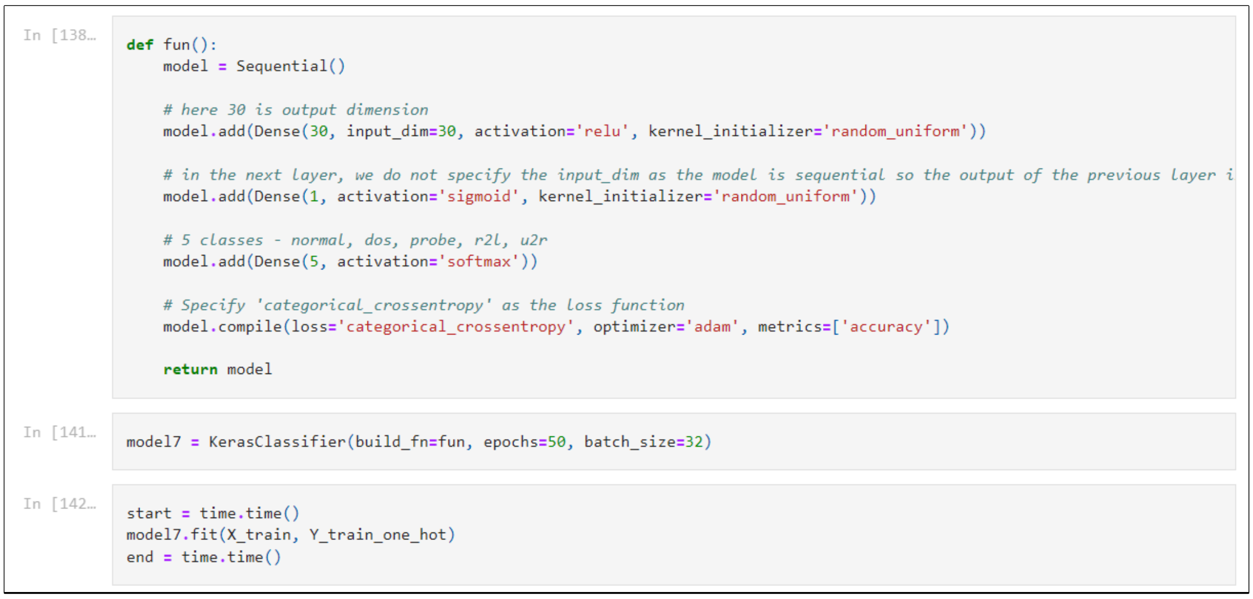
```
In [92]: from sklearn.ensemble import GradientBoostingClassifier

In [93]: model6 = GradientBoostingClassifier(random_state=0)

In [94]: start_time = time.time()
         model6.fit(X_train, Y_train.values.ravel())
         end_time = time.time()

In [95]: print("Training time: ",end_time-start_time)

         Training time:  369.6737439632416

In [96]: start_time = time.time()
         Y_test_pred6 = model6.predict(X_test)
         end_time = time.time()

In [97]: print("Testing time: ",end_time-start_time)

         Testing time:  1.6141042709350586

In [98]: print("Train score is:", model6.score(X_train, Y_train))
         print("Test score is:", model6.score(X_test,Y_test))

         Train score is: 0.9979304760811374
         Test score is: 0.9977181693829856
```

Artificial Neural Network(ANN):-


Artificial Neural Network

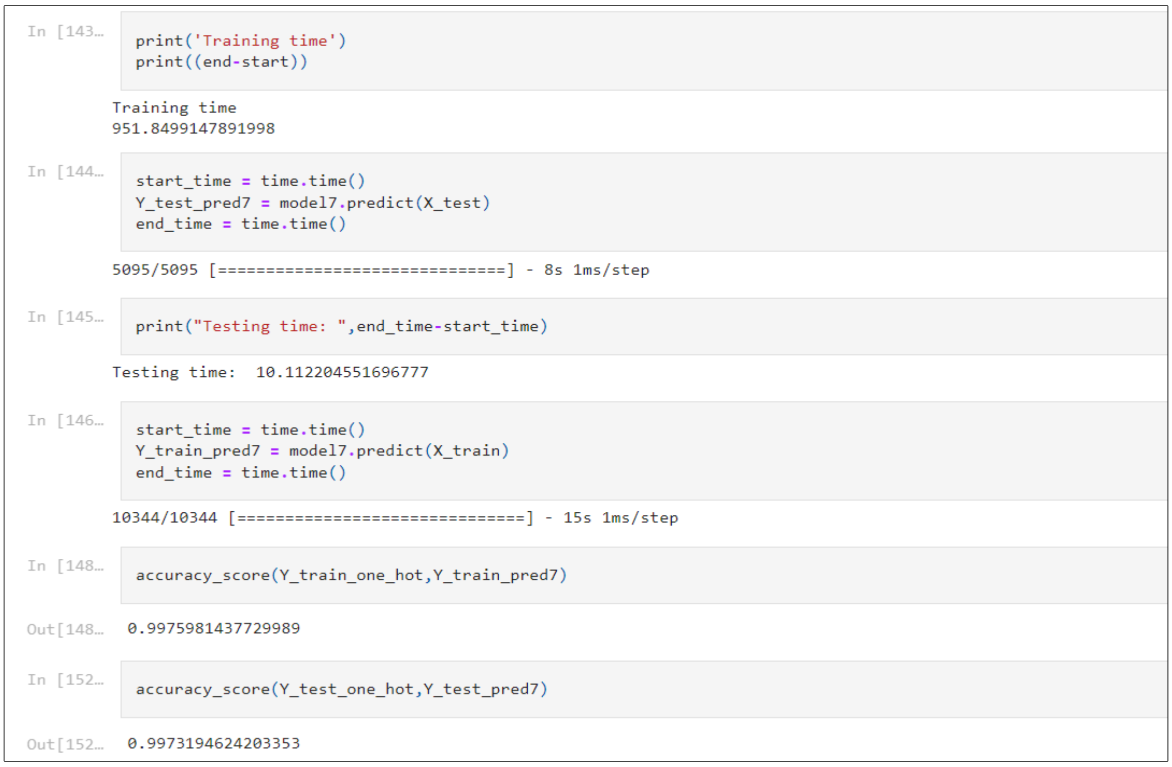
In [136…
```
from sklearn.preprocessing import LabelEncoder
from scikeras.wrappers import KerasClassifier
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
```

In [151…
```
# Convert categorical labels to numerical format
label_encoder = LabelEncoder()
Y_train_encoded = label_encoder.fit_transform(Y_train)
Y_test_encoded = label_encoder.fit_transform(Y_test)

# Convert numerical labels to one-hot encoding
Y_train_one_hot = to_categorical(Y_train_encoded)
Y_test_one_hot = to_categorical(Y_test_encoded)
```


In [138…
```
def fun():
    model = Sequential()

    # here 30 is output dimension
    model.add(Dense(30, input_dim=30, activation='relu', kernel_initializer='random_uniform'))

    # in the next layer, we do not specify the input_dim as the model is sequential so the output of the previous layer i
    model.add(Dense(1, activation='sigmoid', kernel_initializer='random_uniform'))

    # 5 classes - normal, dos, probe, r2l, u2r
    model.add(Dense(5, activation='softmax'))

    # Specify 'categorical_crossentropy' as the loss function
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

    return model
```

In [141…
```
model7 = KerasClassifier(build_fn=fun, epochs=50, batch_size=32)
```

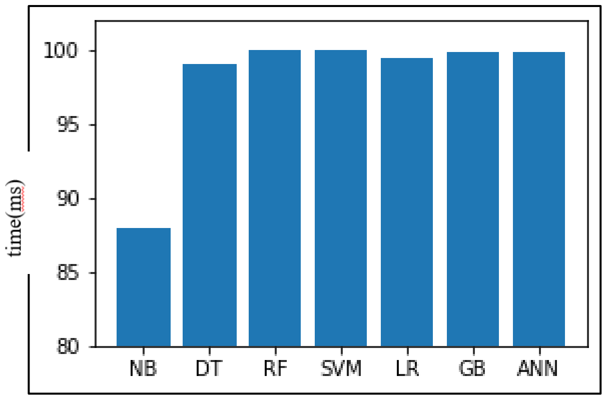
In [142…
```
start = time.time()
model7.fit(X_train, Y_train_one_hot)
end = time.time()
```



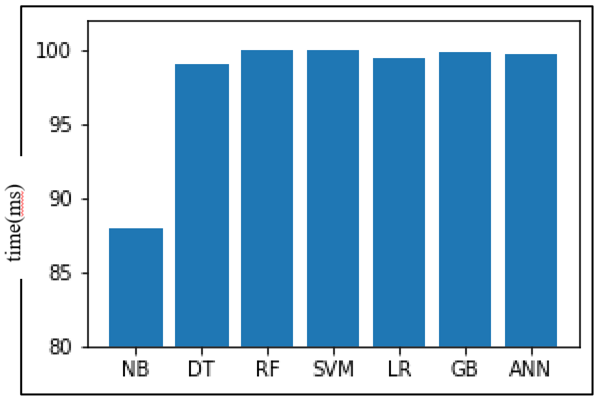
```
In [143…   print('Training time')
           print((end-start))

          Training time
          951.8499147891998

In [144…   start_time = time.time()
           Y_test_pred7 = model7.predict(X_test)
           end_time = time.time()

          5095/5095 [==============================] - 8s 1ms/step

In [145…   print("Testing time: ",end_time-start_time)

          Testing time:  10.112204551696777

In [146…   start_time = time.time()
           Y_train_pred7 = model7.predict(X_train)
           end_time = time.time()

          10344/10344 [==============================] - 15s 1ms/step

In [148…   accuracy_score(Y_train_one_hot,Y_train_pred7)

Out[148…   0.9975981437729989

In [152…   accuracy_score(Y_test_one_hot,Y_test_pred7)

Out[152…   0.9973194624203353
```

10. Result & Output

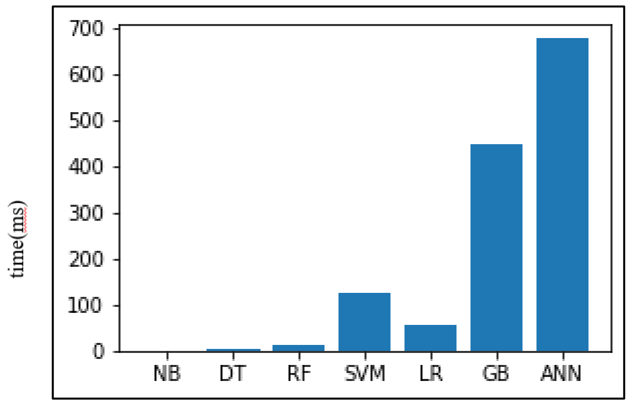
Training Accuracy:-

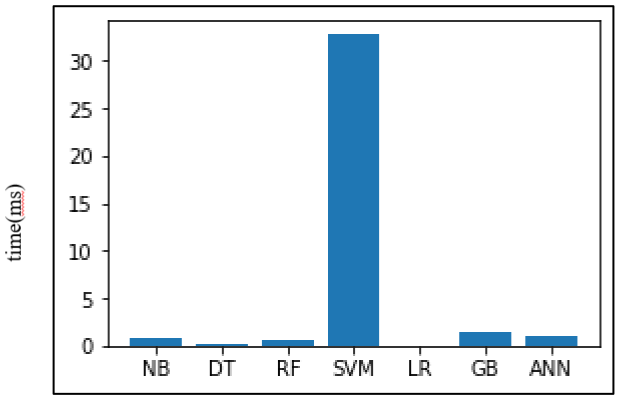


Testing Accuracy:-



Training Time:- Testing Time:-





11. CONCLUSION

Upon analysis of the various algorithms, we made the following observations:

- All algorithms show similar accuracy with training and testing data aside from Naive Bayes which provides a below average accuracy
- The largest training time was noticed in ANN followed by Gradient Boosting.
- Testing Time is negligible in all except SVM

In conclusion, the implementation of an Intrusion Detection System (IDS) utilizing machine learning algorithms represents a significant advancement in enhancing the security posture of computer networks. Through the integration of sophisticated algorithms, our project has demonstrated the capability to effectively detect and mitigate various forms of intrusions, ranging from known patterns to emerging threats.

The successful deployment of machine learning models not only improves the accuracy of intrusion detection but also allows for adaptive learning, enabling the system to evolve and adapt to the dynamic landscape of cyber threats. As we continue to refine and optimize these algorithms, the potential for proactive threat detection and response becomes even more promising, contributing to a resilient and robust defense against cyber threats in contemporary network environments.

This project serves as a testament to the efficacy of machine learning in bolstering the security infrastructure of computer networks, paving the way for future innovations in the field of cybersecurity.

12. REFERENCE

1. Axelsson, Stefan. "Intrusion Detection Systems: A Survey and Taxonomy." Published in 2000.

2. Roesch, Martin. "Snort - Lightweight Intrusion Detection for Networks." Published in 1999.

3. Vaarandi, Risto. "Anomaly-Based Intrusion Detection." Published in 2003.

4. Denning, Dorothy E. "An Intrusion Detection Model." Published in 1986.

5. Alazab, Mamoun et al. "Machine Learning Techniques in Intrusion Detection Systems: A Survey." Published in 2012.

6. Spafford, Eugene H. "The Design and Implementation of Tripwire: A File System Integrity Checker." Published in 1994.

7. Kaza, Siddharth et al. "Intrusion Detection in Wireless Ad Hoc Networks." Published in 2004.

8. Chakrabarti, Amlan et al. "A Survey of Intrusion Detection Systems in Cloud." Published in 2017.

9. Patcha, Animesh, and Jung-Min Park. "An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends." Published in 2007.

10. Lee, Wenke et al. "Data Mining Techniques Applied to Intrusion Detection." Published in 2000.

11. Tan, Kay Chen et al. "A Neural Network Based Distributed Intrusion Detection System for the Detection of SYN Flood Attacks." Published in 2002.
12. Scarfone, Karen, and Peter Mell. "Guide to Intrusion Detection and Prevention Systems (IDPS)." Published in 2007.
13. Lippmann, Richard P. et al. "The 1999 DARPA Off-Line Intrusion Detection Evaluation." Published in 2000.
14. Porras, Phillip A. et al. "A Mission-Based Approach to Host Intrusion Detection." Published in 1998.
15. Tavallaee, Mahbod et al. "A Detailed Analysis of the KDD CUP 99 Data Set for Intrusion Detection Systems." Published in 2009.
16. Lunt, Teresa F. "Automated Auditing for UNIX: A Brief Overview." Published in 1991.
17. Salem, Abdelaziz et al. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection." Published in 2018.
18. Paxson, Vern, and Sally Floyd. "Wide-Area Traffic: The Failure of Poisson Modeling." Published in 1995.
19. Huang, Ying et al. "A Hybrid Intrusion Detection System Based on Ensemble Learning Algorithms." Published in 2018.
20. Javitz, Harold S., and Angelos D. Keromytis. "The Nemesis Intrusion Detection System." Published in 1994.