ADIKAVI NANNAYA UNIVERSITY

RAJAMAHENDRAVARAM

UNIVERSITY COLLEGE OF ENGINEERING

INTRUSION DETECTION SYSTEM

An Internship report is submitted in partial fulfillment of the requirement

for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

CHOWHAN BHAVANI NEERAJ SINGH

Reg.no:178297601003

Under the esteemed guidance of

Dr.D.LATHA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ADIKAVI NANNAYA UNIVERSITY

RAJAMAHENDRAVARAM

2017-2021

EDUYAANTRIX
TOWARDS RIGHT EDUCATION

Sunrise Incubation Centre, Auto Nagar,
Behind CYIENT, Kakinada
info@eduyaantrix.com
www.eduyaantrix.com

INTERNSHIP CERTIFICATE

TO WHOMEVER IT MAY CONCERN

Eduyaantrix Technologies Private Limited (SRN R47842752) certifies that **Mr. CHOWHAN BHAVANI NEERAJ SINGH** successfully completed the internship program from 25-05-2020 to 05-07-2020 on Cyber Security.

During this time, He trained on Network Security, Metasploit, SQL Injection, Social Engineering, Malware Analysis and Penetration Testing Technologies & Techniques. Later he did a project on **Intrusion Detection on Networks using Machine Learning**.

He displayed professional traits during his internship period and managed to complete all assigned tasks as requested. He was hardworking, dedicated, and committed. If you have any questions about his project training experience, you can contact us.

Regards,

P S S Siva Krishna
Co-Founder & Managing Director
Eduyaantrix Technologies Private Limited (SRN R47842752)
info@eduyaantrix.com
www.eduyaantrix.com

ADIKAVI NANNAYA UNIVERSITY, RAJAMAHENDRAVARAM

UNIVERSITY COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that this internship report entitled, "**INTRUSION DETECTION SYSTEM**" is a bonafied work of **CHOWHAN BHAVANI NEERAJ SINGH**, Reg.No:178297601003 submitted in a partial fulfillment of the requirements for the award of Degree of B.Tech(CSE) during the period 2017-2021. This work is carried out by him under my supervision and guidance and submitted to Department of Computer Science and Engineering, Adikavi Nannaya University.

.

INTERNSHIP/COURSE GUIDE **HEAD OF THE DEPARTMENT**

DECLARATION

I, **CH.B.Neeraj Singh**, Reg.No:**178297601003**, hereby declare that the project report entitled **Intrusion Detection System** done by me under the guidance of **Dr D.Latha** , Assistant Professor, Department of Computer Science and Engineering, University College of Engineering, Adikavi Nannaya University, is submitted for the partial fulfillment of requirement for the award of the degree, Bachelors of Technology in Computer Science and Engineering in the academic year **2017-2021**.

CH.B.Neeraj Singh

178297601003

ACKNOWLEDGEMENT

I wish to express my deep sense of gratitude to the management of **Eduyaantrix Technologies Private Ltd**, Kakinada, for giving me an opportunity to complete my **Intrusion Detection System** for the partial fulfillment of my degree in Bachelors of Technology in Computer Science and Engineering.

I would like express my gratitude to Mr.D.Sai Madhav, Eduyaantrix, who has shown keen interest and patience and extended his valuable help during the project period. Also for his valuable suggestions and constant motivation that greatly helped the project work to get successfully completed.

ABSTRACT

An Intrusion Detection System is a software application that monitors a network or systems for malicious activity or policy violations. Any intrusion activity or violation is typically reported to an administrator. An Intrusion Detection System monitors the packets from the device and will alert the user or administrator if suspicious activity is detected. It takes a snapshot of existing system files and matches it to the previous snapshot. It is used for detection of attacks by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware.

Therefore, this project is carried out to build a prototype to detect the attack in network . Evolutionary methodology was implemented.

.

INDEX

S.No.	Contents	Page no
1.	INTRODUCTION	1
2.	SYSTEM ANALYSIS 2.1 Problem Statement 2.2 Software requirements specification 2.2.1 Functional Requirements 2.2.2 Non- Functional Requirements 2.2.3 Software Requirements 2.2.4 Hardware Requirements 2.2.5 Feasibility Study 2.3 Objective 2.4 Scope and Applications	2-4
3.	SYSTEM DESIGN 3.1 System design 3.2 Phases of Intrusion detection system 3.1.1 Environment setup 3.2.2 Load datasets 3.2.3 Data preprocessing 3.3.4 Model training 3.4.5 Evaluating model 3.4.6 Testing the model	5-14
4	SYSTEM IMPLEMENTATION 4.1 Implementation tools 4.2 System Implementation	15-24
5.	Conclusion	25
6.	Bibliography	26

1. INTRODUCTION TO

INTRUSTION DETECTION SYSTEM

The growing number of security threats on the Internet and computer networks demands highly reliable security solutions. So the people are now showing interest in the security on the information and the software's that are used to secure their information. An Intrusion Detection System (IDS) is a system that monitors network traffic for suspicious activity and issues alert when such activity is discovered. It is a software application that scans a network or a system for the harmful activity or policy breaching. Any malicious venture or violation is normally reported either to an administrator or collected centrally using a security information and event management (SIEM) system. A SIEM system integrates outputs from multiple sources and uses alarm filtering techniques to differentiate malicious activity from false alarms.

Although intrusion detection systems monitor networks for potentially malicious activity, they are also disposed to false alarms. Hence, organizations need to fine-tune their IDS products when they first install them. It means properly setting up the intrusion detection systems to recognize what normal traffic on the network looks like as compared to malicious activity. Intrusion prevention systems also monitor network packets inbound the system to check the malicious activities involved in it and at once sends the warning notifications. The various types of IDS are Network Intrusion Detection System (NIDS), Host Intrusion Detection System (HIDS), Protocol-based Intrusion Detection System (PIDS), Application Protocol-based Intrusion Detection System(APIDS), Hybrid Intrusion Detection System.

This system recognizes the various types of intrusions that occurred at the network level and using ML algorithms it tells how accurate our model is.

2. SYSTEM ANALYSIS

2.1. PROBLEM STATEMENT:

Intrusion Detection System is key software in most networking-related applications and active research topic in the Cybersecurity domain. Different methods, techniques, and algorithms have been used to develop for Intrusion Detection System. Intrusion Detection System helps the user to automatically detect intrusion in the network without user monitoring. Previously, An user is needed to observe and check the packets whether there is any attack happened or happening manually. So this project is developing to replace users to monitor the network and automatically finds the intrusions.

2.2. SOFTWARE REQUIREMENTS SPECIFICATION:

2.2.1. Functional Requirements:

Functional requirements are the functions of a system or its component, where a function is described as a specification of behavior between outputs and inputs. Some of the Functional components are Information source, Analysis, and Response.

2.2.2. Non-Functional Requirements:

Nonfunctional requirements are not directly concerned with the specified function delivered by the system. Some of the non-functional requirements are usability, reliability.

2.2.3. Software Requirements:

The software requirements of the project are:

a) Python programming language

b) Jupyter Notebook (Python editor)

c) Windows OS

2.2.4. Hardware Requirements:

The necessary hardware requirements of the project:

a) Fluently working Laptops (64 bit preferable)

b) RAM minimum 6GB

2.2.5. Feasibility Study:

Before starting the project, feasibility study is carried out to measure the viable of the system. Feasibility study is determined to creating a new or improved system is friendly with the cost, benefits, operation, technology and time. Following feasibility study is given as below:

a) **Technical Feasibility-**

Technical feasibility is one of the first studies that must be conducted after the project has been identified. Technical feasibility study includes the hardware and software de vices. The required technologies are Python language and Python IDLE IDE.

b) **Operational Feasibility-**

Operational Feasibility is a measure of how well a proposed system solves the problem and takes the advantage of opportunities which are identified during scope definition. The project's technical feasibility of the system will takes the input of the packets and processed to classify the type of attack in the network.

c) **Economic Feasibility-**

The purpose of economic feasibility is to determine the positive economic benefits that include quantification and identification. The system is economically feasible due to availability of all requirements such as collection of data from internet.

d) **Schedule Feasibility-**

Schedule feasibility is a measure for the reasonability of the project timetable. The system is found schedule feasible due to the system is designed in such a way that it finishes with in prescribed time.

2.3. Objective:

1) To develop a model to detect intrusion.
2) To experiment machine learning algorithm in the domain of Cyber security.
3) To classify the type of Attacks in network.

2.4. Scope and Applications:

The scope of this system is to tackle with the problems that can arise in day to day life. Some of the scopes are:

1) The system can be used to monitoring traffic flow and facilitating information systems to deal with the attacks.
2) Intrusion Detection System is used in Intrusion Prevention Systems, that identify malicious packets and prevent that the network traffic is altered or in the worst case interrupted.
3) The system is used in the firewalls to avoid the misuse of free ports in it.
4) The system can also identifies the intrusions by monitoring and interpreting the communication on application specific protocols in a servers.
5) The system can be used for protecting an enterprise design by identifying, logging, reporting, and sending an alarm whenever there is a probe

3. SYSTEM DESIGN

3.1. System Design:

In this System design there are different phases of Intrusion detection system which are Environment setup, Data loading, Data preprocessing, Model training and Model evaluation. Below diagram shows the phases in the system:

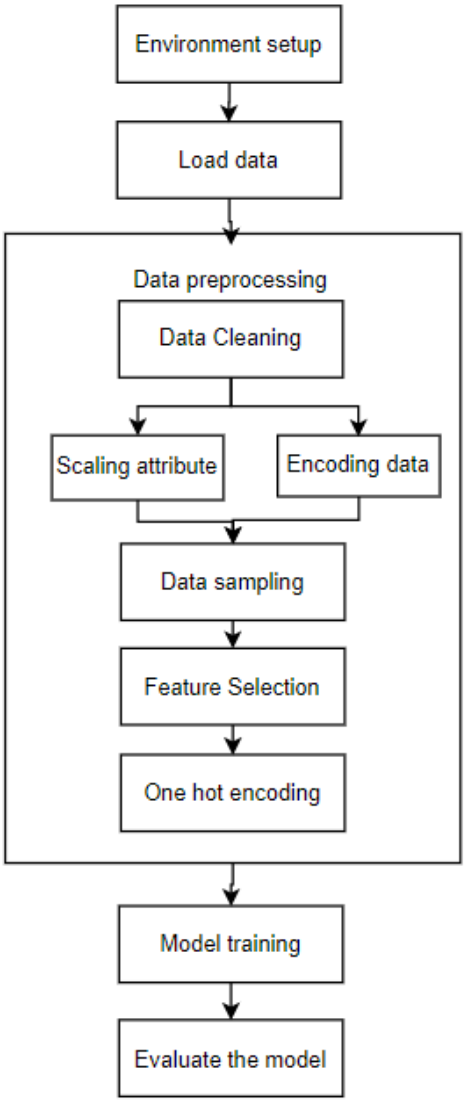

Fig 3.1 Phases of Intrusion Detection System

3.2. Phases of Intrusion Detection System:

Intrusion detection System is trained using supervised learning approach, which takes dataset of packets. The system includes the training and testing phases followed by Importing dataset, Preprocessing data, Prediction. The model is trained using K-Nearest Neighbor classifier and Logistic regression.

3.2.1. Environment Setup:

All necessary libraries in python are imported. The libraries required to build Intrusion Detection System are below stated:

1) NumPy: It is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

2) Seaborn: It is a data visualization library built on top of matplotlib and closely integrated with pandas data structures in Python. Visualization is the central part of Seaborn which helps in exploration and understanding of data.

3) Pandas : It is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the Data Frame. Data Frames allow you to store and manipulate tabular data in rows of observations and columns of variables.

4) Matplotlib : It is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

5) Sklearn : It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

6) Imblearn: It is a python package offering a number of re-sampling techniques commonly used in datasets showing strong between-class imbalance.

3.2.2. Load Datasets: The Train and Test datasets are downloaded from Kaggle dataset website which consists of train datasets. It contains over 125973 rows and 42 columns of data which are used to train the required model. And the test dataset consists of 22544 rows and 42 columns .Some of the attributes in the dataset are:

1) **Duration:** Length of time duration of the connection
2) **Protocol_type:** Protocol used in the connection
3) **Service:** Destination network service used
4) **Flag:** Status of the connection – Normal or Error
5) **Src_bytes:** Number of data bytes transferred from source to destination in a single connection
6) **Dst_bytes:** Number of data bytes transferred from destination to source in single connection
7) **Land:** If source and destination IP addresses and port numbers are equal then, this variable takes value 1 else 0
8) **Wrong_fragment:** Total number of wrong fragments in this connection
9) **Urgent:** Number of urgent packets in this connection. Urgent packets are packets with the urgent bit activated.
10) **Hot:** Number of „hot" indicators in the content such as: entering a system directory, creating programs and executing programs.
11) **Num_failed _logins:** Count of failed login attempts.
12) **Logged_in Login Status**: 1 if successfully logged in; 0 otherwise.
13) **Num_compromised**: Number of "compromised' ' conditions.
14) **Root_shell:** 1 if root shell is obtained; 0 otherwise.
15) **Su_attempted:** 1 if "su root" command attempted or used; 0 otherwise.
16) **Num_root:** Number of "root" accesses or number of operations performed as a root in the connection.
17) **Num_file_creations:** Number of file creation operations in the connection.
18) **Num_shells:** Number of shell prompts.
19) **Num_access_files**: Number of operations on access control files .
20) **Num_outbound_cmds:** Number of outbound commands in an ftp session.
21) **s_hot_login:** 1 if the login belongs to the "hot" list i.e., root or admin,else 0.

22) **Is guest login:** 1 if the login is a “guest” login; 0 otherwise .

23) **Count:** Number of connections to the same destination host as the current connection in the past two seconds

24) **Srv_count:** Number of connections to the same service (port number) as the current connection in the past two seconds.

25) **Serror_rate:** The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in count (23)

26) **Srv_serror_rate:** The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in srv_count (24)

27) **Rerror_rate:** The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in count (23)

28) **Srv_rerror_rate:** The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in srv_count (24)

29) **Same_srv_rate:** The percentage of connections that were to the same service, among the connections aggregated in count (23)

30) **Diff_srv_rate:** The percentage of connections that were to different services, among the connections aggregated in count (23)

31) **Srv_diff_host_ rate:** The percentage of connections that were to different destination machines among the connections aggregated in srv_count (24)

32) **Dst_host_count:** Number of connections having the same destination host IP address.

33) **Dst_host_srv_ count:** Number of connections having the same port number.

34) **Dst_host_same _srv_rate:** The percentage of connections that were to the same service, among the connections aggregated in dst_host_count (32) .

35) **Dst_host_diff_ srv_rate:** The percentage of connections that were to different services, among the connections aggregated in dst_host_count (32)

36) **Dst_host_same _src_port_rate:** The percentage of connections that were to the same source port, among the connections aggregated in dst_host_srv_c ount (33) .

37) **Dst_host_srv_ diff_host_rate:** The percentage of connections that were to different destination machines, among the connections aggregated in dst_host_srv_count (33).

38) **Dst_host_serro r_rate:** The percentage of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_count (32).

39) **Dst_host_srv_s error_rate:** The percent of connections that have activated the flag (4) s0, s1, s2 or s3, among the connections aggregated in dst_host_srv_c ount (33).

40) **Dst_host_rerro r_rate:** The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in dst_host_count (32) .

41) **Dst_host_srv_r error_rate:** The percentage of connections that have activated the flag (4) REJ, among the connections aggregated in a dst_host_srv_c ount (33).

42) **Label :** It defines the type of attack class that are DOS,R2L,Probe,U2R

3.2.3. Data Preprocessing: The data gets transformed, or encoded, to bring it to such a state that, the machine can easily parse it. In other words, the features of the data can now easily interpreted by the algorithm. The features can be categorical or ordinal data. .

1) **Data cleaning**: Data cleaning is a process where the identification of all the missing values and the unnecessary attributes are dropped .

2) **Scaling Numerical Attributes:** All the integer or float data values present in the dataset is done around the mean and with unit variance instead of their original values. This can be done using Standard Scalar present in sklearn library.

3) **Encoding the Categorical data**: The categorical values into numerical labels using LabelEncoder which is present in sklearn library. For example:

Labels	Dos	Normal	Probe
Label after encoding	0	1	2

Fig 3.2. LableEncoding categorical data

4) **Data Sampling**: Data sampling is a statistical analysis technique used to select, manipulate and analyze the representative subset of data points to identify patterns and trends in the larger data set which are examined. The outputs of the train dataset from both scaling and encoding steps are combined using RandomOverSampler. The naive strategy is to generate new samples which are randomly sampled with the replacement of the currently available samples.This RandomOverSampler can be used for two-class (binary) classification problems and multi-class classification problems with one or more majority or minority classes.

5) **Feature Selection:** Feature selection is the process of reducing the number of input variables by developing a predictive model. It is desirable to reduce the number of input variables to reduce the computational cost of Modeling. In some cases, it will improve the performance of the model. By using the Random Forest Classifier on the training set. Random forests consist of 4 –12 hundred decision trees. Each tree is also a sequence of yes-no questions based on a single or combination of features. At each node the tree divides the dataset into 2 buckets, each of them hosting observations that are more similar among themselves and different from the ones in the other bucket. The level of importance of the features are known and among those features main attributes are obtained by using the Recursive feature elimination.

6) **One Hot encoding:** This is done using One Hot Encoder. It refers to splitting of column which contains numerical categorical data to many columns depending on the number of categories present in that column. Each column contains "0" or "1" corresponding to which column it has been placed. In One Hot encoder the integer encoded variable in the label encoding is removed and one new binary variable is added for each unique integer value in the variable. For Example.

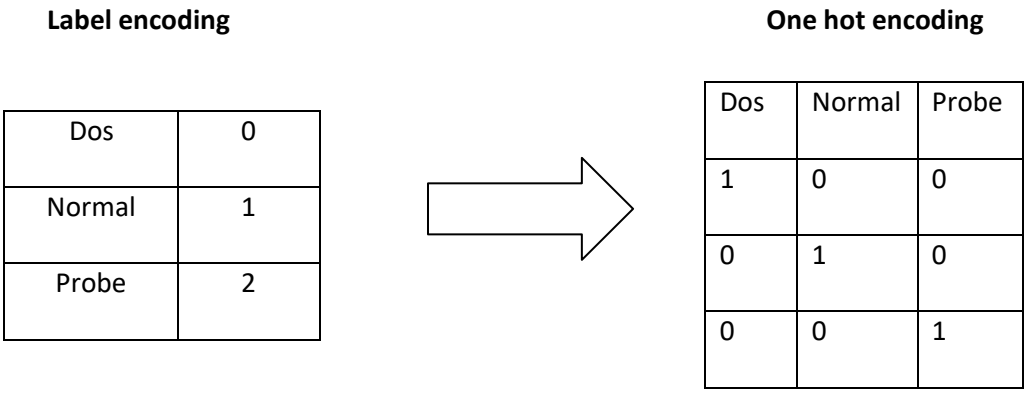

Fig 3.3. OneHotEncoding from LabelEncoding.

7) **Final Data preparation**: The final data is prepared after undergoing all the changes in every step of the preprocessing process. Now the Machine learning algorithms are applied on this data.

3.2.4. Model training:

In this step, we apply Machine Learning algorithms on the final train dataset of the preprocessing step.

1) **K-Nearest Neighbor (K-NN):** It is a Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. It is also called a lazy learner algorithm because it does not learn from the training set. Instead it stores the dataset and at the time of classification, it performs an action on the dataset.

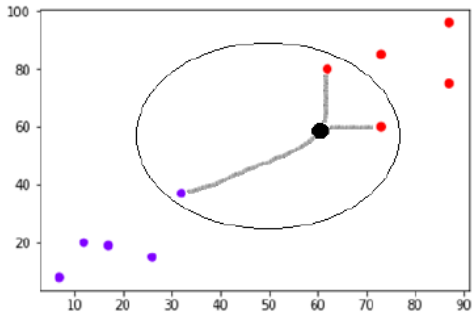

Fig 3.4 K-Nearest Neighbor graph of measuring distance

Pseudo code:

Step 1 − For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

Step 2 − Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

Step 3 − For each point in the test data do the following

3.1 − Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

3.2 − Now, based on the distance value, sort them in ascending order.

3.3 − Next, it will choose the top K rows from the sorted array.

3.4 − Now, it will assign a class to the test point based on most frequent class of these rows.

Step 4 – End

2) **Logistic regression:** Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. In this we use multinomial logistic regression .The logistic regression uses a more complex cost function which is sigmoid function or logistic function. The hypothesis of logistic regression tends to limit the cost function between 0 and 1.

Sigmoid function maps any real value into another value between 0 and 1, we use sigmoid to map predictions to probabilities. The sigmoid formula is

$$f(x) = \frac{1}{1+e^{-(x)}}$$

The representation of the hypothesis of logistic regression is shown below

$$y = e^{(b0 + b1*x)} / (1 + e^{(b0 + b1*x)})$$

The logistic regression graph is shown as.

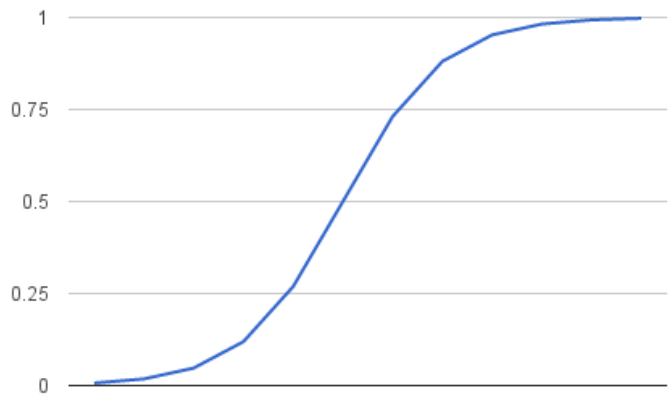

Fig 3.5 Graph of Logistic regression

3.2.5. Evaluating the model:

In this step the model is evaluated by using metrics in sklearn. The different metrics used are accuracy score, confusion matrix, and classification report.

1) **Accuracy score:** The ratio of correct predictions to the total number of input samples.

$$\text{Accuracy score} = \frac{\text{Number of corrected predictions}}{\text{Total number of predictions made}}$$

2) **`Confusion matrix:** A confusion matrix is also known as an error matrix. This matrix gives us the output as a matrix which describes the complete performance of the model. The accuracy for the matrix is calculated by taking average of the values laying across the main diagonal.

$$\text{Accuracy} = \frac{\text{(True positive + True negative) x 100}}{\text{i. Total Sample}}$$

3) **Classification Report**: The classification report visualizer displays the precision, recall, f1-score, and support for the model.

   a) **Precision:** It defines what percent of the model predictions are correct. It is the ratio of True positive (TP) to the sum of a True positive and False positive (FP).

$$\text{Precision} = \frac{\text{TP}}{\text{(TP+FP)}}$$

   b) **Recall:** It is the fraction of positives that are correctly identified by the classifie**r.** It is the ratio of True positive (TP) to the sum of a True positive and False negative (FN).

$$\text{Recall} = \frac{\text{TP}}{\text{(TP+FN)}}$$

c) **F1-Score:** It is a weighted harmonic mean of precision and recall such that the best score is 1 (one) and the worst score is 0 (zero).

$$\text{F1-Score} = \frac{(2 \times \text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}}$$

d) **Support**: Support is the number of actual occurrences of the class in a specified dataset.

3.2.6. Testing the Models:

The testing is done on the test dataset which is the output of the final data after preprocessing step. The model is tested based on the metrics, Accuracy, Classification report, Confusion Matrix.

4. SYSTEM IMPLEMENTATION

4.1. Implementation Tools:

1) **Python:**

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

2) Jupyter Notebook:

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text.

Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Installing Jupyter Notebook-

This information explains how to install the Jupyter Notebook and the IPython kernel.

Prerequisite: Python

Installing Jupyter using Anaconda and conda :

Anaconda conveniently installs Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

a) Download Anaconda’s latest Python 3 version (currently Python 3.5).

b) Install the version of Anaconda which you downloaded, following the instructions on the download page.

c) Finally, Launch the jupyter notebook.

Alternative for experienced Python users: Installing Jupyter with pip using cmd

Jupyter installation requires Python 3.3 or greater, or Python 2.7. IPython 1.x, which included the parts that later became Jupyter, was the last version to support Python 3.2 and 2.6.

As an existing Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

First, ensure that you have the latest pip; older versions may have trouble with some dependencies:

pip3 install --upgrade pip

Then install the Jupyter Notebook using:

pip3 install jupyter

The Jupyter Notebook will be installed by these processes.

4.2. Implementation:

a) **Environment Setup:** The libraries like matplotlib, pandas, numpy, imblearn ,sklearn , seaborn are imported using the 'import' statement in the initial step of development.

Environment setup

```
In [1]: # import relevant modules
import matplotlib
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
import sklearn
import imblearn
import sys

print("pandas : {0}".format(pd.__version__))
print("numpy : {0}".format(np.__version__))
print("matplotlib : {0}".format(matplotlib.__version__))
print("seaborn : {0}".format(sns.__version__))
print("sklearn : {0}".format(sklearn.__version__))
print("imblearn : {0}".format(imblearn.__version__))
```

```
pandas : 1.0.1
numpy : 1.16.6
matplotlib : 3.1.3
seaborn : 0.10.0
sklearn : 0.23.2
imblearn : 0.7.0
```

Fig 4.1 importing libraries

b) **Load Datasets:** In this phase, both the test and train datasets are imported using 'read_csv()' function in pandas libraries. The number of rows and columns of data frame is given by the ".shape" attribute.

```
In [2]: # Load train dataset
df_train = pd.read_csv("Train.csv")
df_train.shape

Out[2]: (125973, 44)

In [3]: # Load test dataset
df_test = pd.read_csv("Test.csv")
df_test.shape

Out[3]: (22544, 43)
```

Fig 4.2 Importing dataset

The diagram states that the 'Train' dataset has 125973 records and 44 attributes, similarly the 'Test' dataset have 22544 records and 43 attributes.

c) **Data Preprocessing:**

The final data is prepared by cleaning, integration and transforming the raw data that is present in our dataset it gives the data to model for training efficiently.

1) **Data cleaning:** As a part of data cleaning process all the missing values of attributes and unnecessary attributes are removed in this phase. As there is no missing values in dataset, unnecessary attributes are checked using value_count() function. The 'value_count()' function gives the count of unique values in the attribute and drop the unnecessary attributes using 'drop()' in pandas. The drop () function takes label of the column or row, axis to drop (column or row).

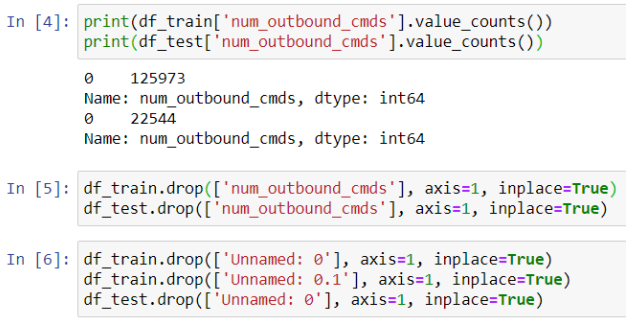

Fig 4.3 Treating with unnecessary attributes

The 'num_outbound_cmds' field has all 0 values. Hence, it will be removed from both train and test dataset as it is a redundant field.

2) **Scaling Numerical Attributes**: This is done using StandardScalar present in sklearn library through "fit_transfer()" function. The integer and float (numerical) values are scaled in this step.

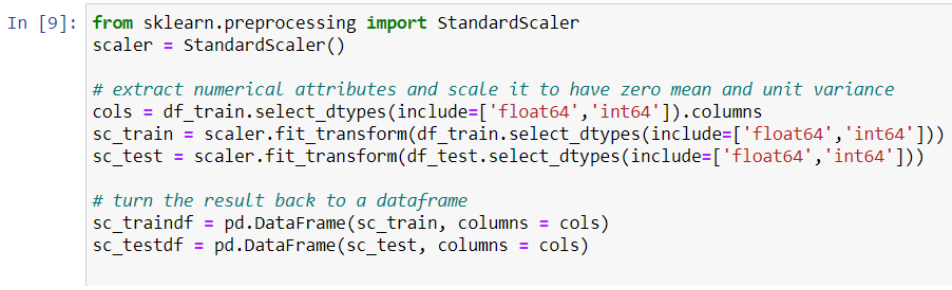

Fig 4.4 Scaling numerical attribute

3) **Encoding the Categorical data**: The categorical values present in both train and test dataset are transformed to numerical labels using 'fit_transform' attribute by 'apply()' function in 'LabelEncoder' which is imported from 'sklearn' library.

```
In [10]: from sklearn.preprocessing import LabelEncoder
         encoder = LabelEncoder()

         # extract categorical attributes from both training and test sets
         cattrain = df_train.select_dtypes(include=['object']).copy()
         cattest = df_test.select_dtypes(include=['object']).copy()

         # encode the categorical attributes
         traincat = cattrain.apply(encoder.fit_transform)
         testcat = cattest.apply(encoder.fit_transform)

         # separate target column from encoded data
         enctrain = traincat.drop(['attack_class'], axis=1)
         enctest = testcat.drop(['attack_class'], axis=1)

         cat_Ytrain = traincat[['attack_class']].copy()
         cat_Ytest = testcat[['attack_class']].copy()
```

Fig 4.5 LabelEncoding the attributes in datasets

4) **Data Sampling**: The outputs of the train dataset from both scaling and encoding steps are combined using 'Concat()' function and sampled by using the 'fit_sample()' of 'RandomOverSampler' imported from 'imblearn' library.

```
In [11]: from imblearn.over_sampling import RandomOverSampler
         from collections import Counter

         # define columns and extract encoded train set for sampling
         sc_traindf = df_train.select_dtypes(include=['float64','int64'])
         refclasscol = pd.concat([sc_traindf, enctrain], axis=1).columns
         refclass = np.concatenate((sc_train, enctrain.values), axis=1)
         X = refclass

         # reshape target column to 1D array shape
         c, r = cat_Ytest.values.shape
         y_test = cat_Ytest.values.reshape(c,)

         c, r = cat_Ytrain.values.shape
         y = cat_Ytrain.values.reshape(c,)

         # apply the random over-sampling
         ros = RandomOverSampler(random_state=42)
         X_res, y_res = ros.fit_sample(X, y)
         print('Original dataset shape {}'.format(Counter(y)))
         print('Resampled dataset shape {}'.format(Counter(y_res)))

Original dataset shape Counter({1: 67343, 0: 45927, 2: 11656, 3: 995, 4: 52})
Resampled dataset shape Counter({1: 67343, 0: 67343, 3: 67343, 2: 67343, 4: 67343})
```

.

Fig 4.6 Sampling the data in dataset

5) **Feature Selection:** The 'RandomForestClassifier()' is used to fit the training dataset and the feature importance score is given by the '.feature_importances' attribute and plotted the graph in sorted order of importance using '.sort_values()' function.

```
In [14]: from sklearn.ensemble import RandomForestClassifier
         rfc = RandomForestClassifier();

         # fit random forest classifier on the training set
         rfc.fit(X_res, y_res);
         # extract important features
         score = np.round(rfc.feature_importances_,3)
         importances = pd.DataFrame({'feature':refclasscol,'importance':score})
         importances = importances.sort_values('importance',ascending=False).set_index('feature')
         # plot importances
         plt.rcParams['figure.figsize'] = (11, 4)
         importances.plot.bar();
```

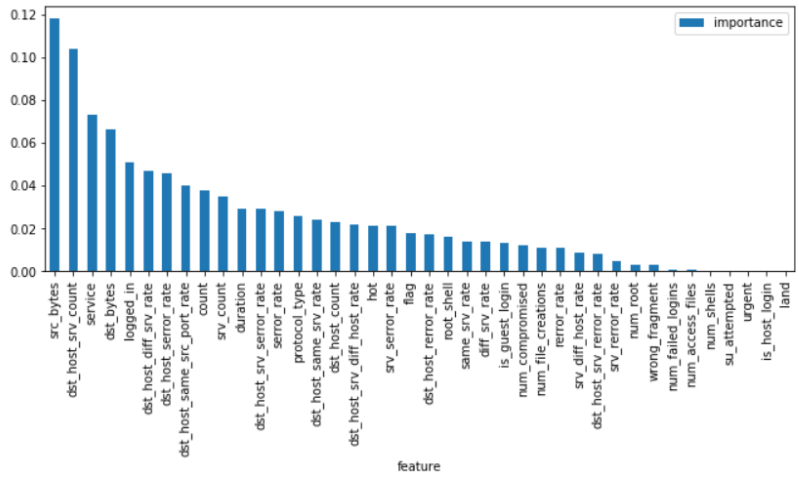

Fig 4.7 Feature importance of attributes in dataset

By using the 'RFE'(recursive feature elimination) from the 'feature_selection' module of sklearn library, the top 10 attributes are selected that are important and reaming are discarded .the selected attributes are shown below:

```
In [21]: from sklearn.feature_selection import RFE
         import itertools
         rfc = RandomForestClassifier()

         # create the RFE(Recursive feature elimination) model and select 10 attributes
         rfe = RFE(rfc, n_features_to_select=10)
         rfe = rfe.fit(X_res, y_res)

         # summarize the selection of the attributes
         feature_map = [(i, v) for i, v in itertools.zip_longest(rfe.get_support(), refclasscol)]
         selected_features = [v for i, v in feature_map if i==True]
         selected_features

Out[21]: ['src_bytes',
          'dst_bytes',
          'logged_in',
          'count',
          'srv_count',
          'dst_host_srv_count',
          'dst_host_diff_srv_rate',
          'dst_host_same_src_port_rate',
          'dst_host_serror_rate',
          'service']
```

Fig 4.8 Feature selection of attributes in dataset

6) **Final train data preparation:** This is done using 'OneHotEncoder()' function from preprocessing module of sklearn. The final independent and dependent variables of the train dataset are X_train, Y_train. The final independent and dependent variables of the test dataset are X_test, Y_test. This data is encoded by the '.fit()' function and that transforms the data of test and train dataset. Using the 'concatenate()' function the encoded is added to dataset of train and test .

```
In [26]: from sklearn.preprocessing import OneHotEncoder
enc = OneHotEncoder()

Xresdf = pretrain
newtest = pretest

Xresdfnew = Xresdf[selected_features]
Xresdfnum = Xresdfnew.drop(['service'], axis=1)
Xresdfcat = Xresdfnew[['service']].copy()

Xtest_features = newtest[selected_features]
Xtestdfnum = Xtest_features.drop(['service'], axis=1)
Xtestcat = Xtest_features[['service']].copy()

# Fit train data
enc.fit(Xresdfcat)

# Transform train data
X_train_1hotenc = Xresdfcat

# Transform test data
X_test_1hotenc = Xtestcat

X_train = np.concatenate((Xresdfnum.values, X_train_1hotenc), axis=1)
X_test = np.concatenate((Xtestdfnum.values, X_test_1hotenc), axis=1)

y_train = Xresdf[['attack_class']].copy()
c, r = y_train.values.shape
Y_train = y_train.values.reshape(c,)

y_test = newtest[['attack_class']].copy()
c, r = y_test.values.shape
Y_test = y_test.values.reshape(c,)
```

Fig 4.9 OneHotEncoder of data

d) **Model training:** The KNN classifier and Logistic regression algorithm is applied using 'KNeighborsClassifier' model from 'neighbor' module and 'LogisticRegression' model from 'linear_model' module of sklearn library and trained the model using '.fit()' function , giving independent and dependent variables of pre-processed dataset of train dataset as a parameters to it.

```
In [27]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression


# Train KNeighborsClassifier Model
KNN_Classifier = KNeighborsClassifier(n_jobs=-1)
KNN_Classifier.fit(X_train, Y_train);

# Train LogisticRegression Model
LGR_Classifier = LogisticRegression(n_jobs=-1, random_state=0)
LGR_Classifier.fit(X_train, Y_train);
```

Fig 4.10 Model training

e) Evaluating the model: In this step the model is evaluated by using metrics in sklearn. The accuracy is measured by '.accuracy_score()','.confusion_matric()', '.classification_report()' functions from metrics module .The train data is given to check the model efficient .

```
In [23]: from sklearn import metrics
         models = []
         models.append(('KNeighborsClassifier', KNN_Classifier))
         models.append(('LogisticRegression', LGR_Classifier))

         for i, v in models:

             accuracy = metrics.accuracy_score(Y_train, v.predict(X_train))
             confusion_matrix = metrics.confusion_matrix(Y_train, v.predict(X_train))
             classification = metrics.classification_report(Y_train, v.predict(X_train))
             print()
             print('============================  Model Evaluation ============================'.format(grpclass, i))
             print()
             print ("Model Accuracy:" "\n", accuracy)
             print()
             print("Confusion matrix:" "\n", confusion_matrix)
             print()
             print("Classification report:" "\n", classification)
             print()
```

Fig 4.11 Evaluating model with trained data

The KNN classifier model got the accuracy of 99% accuracy with trained data and the other metrics are shown below

```
============================  Model Evaluation ========

Model Accuracy:
 0.9977577476500898

Confusion matrix:
 [[67287    56]
 [  246 67097]]

Classification report:
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00     67343
         1.0       1.00      1.00      1.00     67343

    accuracy                           1.00    134686
   macro avg       1.00      1.00      1.00    134686
weighted avg       1.00      1.00      1.00    134686
```

Fig 4.12 Output of KNeighbor classifier Model

The Logistic regression classifier model got 96 % of accuracy with trained data and other metrics are shown below

```
Model Accuracy:
 0.8525248995282194

Confusion matrix:
 [[5843 1615]
 [ 917 8794]]

Classification report:
              precision    recall  f1-score   support

         0.0       0.86      0.78      0.82      7458
         1.0       0.84      0.91      0.87      9711

    accuracy                           0.85     17169
   macro avg       0.85      0.84      0.85     17169
weighted avg       0.85      0.85      0.85     17169
```

Fig 4.13 Output of Logistic Regression Model

4.3. Testing model:

In this step the model is evaluated by using metrics in sklearn and test dataset. The different metrics used are accuracy score, confusion matrix, and classification report.

```
In [36]: models = []
         models.append(('KNeighborsClassifier', KNN_Classifier))
         models.append(('LogisticRegression', LGR_Classifier))

         for i, v in models:

             accuracy = metrics.accuracy_score(Y_test, v.predict(X_test))
             confusion_matrix = metrics.confusion_matrix(Y_test, v.predict(X_test))
             classification = metrics.classification_report(Y_test, v.predict(X_test))
             print()
             print('============================  Model Evaluation ============================'.format(grpclass, i))
             print()

             print ("Model Accuracy:" "\n", accuracy)
             print()
             print("Confusion matrix:" "\n", confusion_matrix)
             print()
             print("Classification report:" "\n", classification)
             print()
```

Fig 4.14 Evaluating model with test data

The KNN classifier model got the accuracy of 89% accuracy and the other metrics are

```
Model Accuracy:
 0.8946356805871046

Confusion matrix:
 [[5893 1565]
 [ 244 9467]]

Classification report:
               precision    recall  f1-score   support

          0.0       0.96      0.79      0.87      7458
          1.0       0.86      0.97      0.91      9711

     accuracy                           0.89     17169
    macro avg       0.91      0.88      0.89     17169
 weighted avg       0.90      0.89      0.89     17169
```

Fig 4.15 Output of KNN classifier Model

The Logistic regression classifier model got 85 % of accuracy and other metrics are

```
Model Accuracy:
 0.8525248995282194

Confusion matrix:
 [[5843 1615]
 [ 917 8794]]

Classification report:
               precision    recall  f1-score   support

          0.0       0.86      0.78      0.82      7458
          1.0       0.84      0.91      0.87      9711

     accuracy                           0.85     17169
    macro avg       0.85      0.84      0.85     17169
 weighted avg       0.85      0.85      0.85     17169
```

Fig 4.16 Output of Logistic Regression Model

Ie.The accuracy of the models of the system is

KNN classifier	89% accuracy
Logistic Regression	85% accuracy

Full code is available at: https://github.com/neerajsinghchowhan/INTRUSION-DETECTION/blob/main/Network%20Intrusion%20Detection%20System.ipynb

5. CONCLUSION

In the above system, the classification models gave the accuracy of 85.25% through KNN classifier and 89.46% through Logistic regression. A similar system is implemented by J.S.Sirisha (178297601004) and obtained an accuracy of 82.88% using decision tree algorithm and 86.97% using Naive Bayes algorithm. The logistic regression model turned to be more efficient for this system followed by Naive Bayes algorithm.

This project can be improved by providing the real life data packets in network, directly to the model to detect intrusions. The same system may be further improved by using XG-boost, artificial neural networks and other advanced algorithms.

6. BIBLIOGRAPHY

Intrusion Detection System Introduction, Abstract -
https://en.wikipedia.org/wiki/Intrusion_detection_system

https://www.geeksforgeeks.org/intrusion-detection-system-ids/

Dataset-

https://www.kaggle.com/sampadab17/network-intrusion-detection

Anaconda environment-

https://www.anaconda.com/

K-Nearest Neighbor -
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

Logistic régressions -
https://machinelearningmastery.com/logistic-regression-for-machine-learning/