INTRUSION DETECTION USING MACHINE LEARNING

J Component Review-III

for

SWE3002 INFORMATION AND SYSTEM SECURITY

Submitted by

V. GOPIKA 16MIS0011

To

Dr. SUMAIYA THASEEN I

In

C21+C22+TC23 SLOT

School of Information Technology and Engineering

VIT University, Vellore

Tamil Nadu – 632014

JUNE 2019

TABLE OF CONTENTS

INTRUSION DETECTION USING MACHINE LEARNING

CHAPTER 1

INTRODUCTION

1.1 ABSTRACT

The role of Intrusion Detection in cyber security to monitor and determine intrusion attacks is indispensable. This project aims to implement the top three machine learning algorithms for intrusion detection obtained as a result from the Review of Literatures. The implemented algorithms will be compared based on the selected metrics and the best algorithm to detect intrusion will be obtained as a result of this project.

1.2 INTRODUCTION

The monitoring of network traffic for suspicious activity and issues alerts when such activity discovered is known as Intrusion Detection. Any malicious activity or violations will be typically reported or notified to the Administrator. Network Intrusion, Host based Intrusion, Signature based and Anomaly based are the four different types of Intrusion Detection. This Intrusion Detection effectively prevents the damage to the network and the network or computer is constantly monitored for any invasion or attack. By using Intrusion Detection, any alterations to files and directories on the system can be easily detected and reported. Intrusion Detection can qualify and quantify attacks and this can boost efficiency along with that it analyzes inbound and outbound data on the network. Intrusion Detection analyzes different types of attacks, identifies patterns of malicious content and helps the administrators to tune, organize and implement effective controls.

1.3 OBJECTIVE OF THE WORK

- Intrusion Detection using Machine Learning Literature Review
- Identification of efficient Machine Learning Algorithms for Intrusion Detection
- Implementing Identified Algorithms
- Comparing the Implemented Algorithms
- Analysis of the results obtained from Comparison of the Algorithms

CHAPTER 2

LITERATURE REVIEW

2.1 BACKGROUND

LITERATUE SUMMARY TABLE WITH ADVANTAGES AND DISADVANTAGES

S. NO	PAPER TITLE	TECHNIQUES USED	ADVANTAGES	DISADVANTAGES
1	**Network Intrusion Detection System (NIDS) using Machine Learning Perspective [1]**	1. Support Vector Machine (SVM) 2. Decision Tree 3. Naïve Bayes	• Provides more attack detection accuracy. • It increases the attack detection performance in short span of time.	• Long Training time is required for large datasets to get good accuracy. • High Algorithmic Complexity and extensive memory required.

2	**Network Intrusion Detection Using Machine Learning [2]**	1. Support Vector Machine (SVM) 2. Proposed Method(Modified SVM)	• It provides high accuracy and lower false positive rate and false negative rate.	• Feature Selection is one of the most important parts, whereas the detection accuracy depends on feature selection.
3	**Machine Learning Methods for Network Intrusion Detection [3]**	1. J48 Tree 2. Multilayer Perceptron (MLP) 3. Bayes Network	• Easy To Use and approximate any kind of input or output.	• The training is very slow and the training samples required is three times larger than normal training data set.
4	**Network Intrusion Detection System Using Machine Learning [4]**	1. Decision Tree Classifier	• Processes both numerical and categorical data. • It handles high dimensional data.	• The output created may be more complex than expected. • Chances of detecting abnormal behavior as normal and normal as abnormal.
5	**Intrusion Detection Using**	1. Naïve Bayes 2. Support Vector	• High accuracy and speed is	• Lack of available probability of

	Machine Learning: A Comparison Study [5]	Machine 3. Decision Tree 4. Neural Network 5. K Nearest Neighbor (KNN)	more. • Simple in implementation and easy to understand the flow.	data. • The probability of the outcome is unstable.
6	**A Review of Intrusion Detection System using Machine Learning Approach [6]**	1. Decision Tree 2. Naïve Bayes 3. K Nearest Neighbor 4. K Means 5. Support Vector Machine 6. Principle Component Analysis	• Both Accuracy and Performance are high. • Runs efficiently on large data sets with many features.	• The complexity is little higher. • Maintaining and updating the system may be difficult.
7	**Evaluation of Machine Learning Algorithms for Intrusion Detection System [7]**	1. J48 2. Random Forest 3. Random Tree 4. Decision Table 5. Multi Layer Perceptron (MLP) 6. Naïve Bayes 7. Bayes Network	• The results obtained are more acceptable and the performance of the system was also good along with the pattern recognition.	• The time taken to detect the intrusion was higher than the regular time along with huge memory is required for computation

8	**Intrusion Detection System using AI and Machine Learning Algorithm [8]**	1. K Means Clustering 2. Support Vector Machine 3. K Nearest Neighbors	• The results obtained from both evaluation and real life time was good.	• Selecting the appropriate data set determines the rest of the result.
9	**Intrusion detection model using machine learning algorithm on Big Data [9]**	1. Support Vector Machine (SVM) 2. Spark Chi-SVM Proposed Model	• Process, and analyze data with high speed.	• The high dimensionality makes the classification process more complex and takes long time.
10	**Application of Machine Learning Approaches in Intrusion Detection System: A Survey [10**	1. Support Vector Machine (SVM) 2. Decision Tree 3. Naïve Bayes 4. Logistic Regression 5. K Nearest Neighbor	• Very efficient to train and easy to implement as well as easy to measure the performance of complex algorithms.	• Removal of redundant and irrelevant features for data training is a key factor which determines system performance.

<u>LITERATURE SUMMARY TABLE WITH DATASETS USED AND ACCURACY OBTAINED</u>

S. NO	**PAPER TITLE**	**TECHNIQUES USED**	**ACCURACY OBTAINED**	**DATASETS USED**

1	**Network Intrusion Detection System using Machine Learning Perspective [1]**	1. Support Vector Machine (SVM) 2. Decision Tree 3. Naïve Bayes	**91%** **89%** **82%**	KDDCup Dataset
2	**Network Intrusion Detection Using Machine Learning [2]**	1. Support Vector Machine (SVM) 2. Proposed Method(Modified SVM)	**88.03%** **98.76%**	ACCS (Australian Centre for Cyber Security)
3	**Machine Learning Methods for Network Intrusion Detection [3]**	1. J48 Tree 2. Multilayer Perceptron (MLP) 3. Bayes Network	**93.10%** **91.90%** **90.73%**	KDD Dataset (Knowledge Discovery And Data Mining)
4	**Network Intrusion Detection System Using Machine Learning [4]**	1. Decision Tree Classifier	**90%**	CICIDS 2017
5	**Intrusion Detection Using Machine Learning: A Comparison Study [5]**	1. Naïve Bayes 2. Support Vector Machine 3. Decision Tree 4. Neural Network	**82.66%** **76.61%** **90.88%** **89.30%**	NSL-KDD

		5. K Nearest Neighbor (KNN)	**96%**	
6	**A Review of Intrusion Detection System using Machine Learning Approach [6]**	1. Decision Tree 2. Naïve Bayes 3. K Nearest Neighbor 4. K Means 5. Support Vector Machine 6. Principle Component Analysis	**92%** **94%** **95%** **96%** **92%** **93%**	CIDDS-001 KDDCup99 NSL-KDD
7	**Evaluation of Machine Learning Algorithms for Intrusion Detection System [7]**	1. J48 2. Random Forest 3. Random Tree 4. Decision Table 5. Multi Layer Perceptron (MLP) 6. Naïve Bayes 7. Bayes Network	**93%** **93%** **90%** **92%** **91%** **91%** **90%**	KDD Intrusion Dataset
8	**Intrusion Detection System using AI and Machine**	1. K Means Clustering 2. Support Vector Machine	**89%** **90%**	CTU Dataset

	Learning Algorithm [8]	3. K Nearest Neighbors	**91%**	
9	**Intrusion detection model using machine algorithm on Big Data environment [9]**	1. Support Vector Machine (SVM) 2. Spark Chi-SVM Proposed Model	**94%** **96%**	KDD99 Dataset Resilient Distributed Dataset (RDD)
10	**Application of Machine Learning Approaches in Intrusion Detection System: A Survey [10]**	1. Support Vector Machine (SVM) 2. Decision Tree 3. Naïve Bayes 4. Logistic Regression 5. K Nearest Neighbor	**82%** **89%** **94%** **85%** **93%**	KDD Dataset NSL-KDD Dataset KDD Cup 1999

SELECTED TOP 3 ALGORITHMS FOR IMPLEMENTATION

1. K Nearest Neighbor (KNN)

2. Naïve Bayes

3. Random Forest

2.2 PROBLEM DEFINITION AND APPROACH

In the modern network, Intrusion Detection has become an important and integral part of overall security architecture. With the rapid growth of attacks, several intrusion detection methods and systems have been proposed in the literatures. Though there exist many algorithms, systems and methods in a dispersed way, there is a need for the Administrator to select or identify the efficient algorithm in a limited amount of time is difficult. In this project the most efficient algorithms based on the defined metrics will be identified.

Machine Learning algorithms will be used for intrusion detection. Based on the Literature Review three machine learning algorithms were selected. And these algorithms were selected based on the specified metrics such as accuracy, time required for training and classification, efficiency and complexity of the algorithms. The implemented algorithms will detect the intrusion effectively.

CHAPTER 3

EXPERIMENTAL DETAILS

3.1 MACHINE LEARNING ALGORITHMS DESCRIPTION

3.1.1 K NEAREST NEIGHBOR (KNN)

The K nearest neighbor algorithm is a simple easy to implement supervised machine learning classification algorithm.

The steps involved in KNN Algorithm:

- Load the training and test data
- Choose the value of K
- For each point in test data:
    - Find the Euclidean distance to all training data
    - Store the Euclidean distance in a list and sort it
    - Choose the first K points
    - Assign a class to test the point based on the majority of classes present
- End

3.1.2 NAÏVE BAYES

The Naïve Bayes algorithm is a probabilistic classification based supervised machine learning algorithm based on Bayes Theorem.

The steps involved in Naïve Bayes Algorithm:

- Load the Training dataset T

- Calculate the mean and standard deviation of the predictor variable in each class
- Repeat
  - Calculate the probability of fi using the gauss density equation in each class
  - Until the probability of all predictor variables has been calculated
- Calculate the likelihood for each class
- Get the greatest likelihood
- End

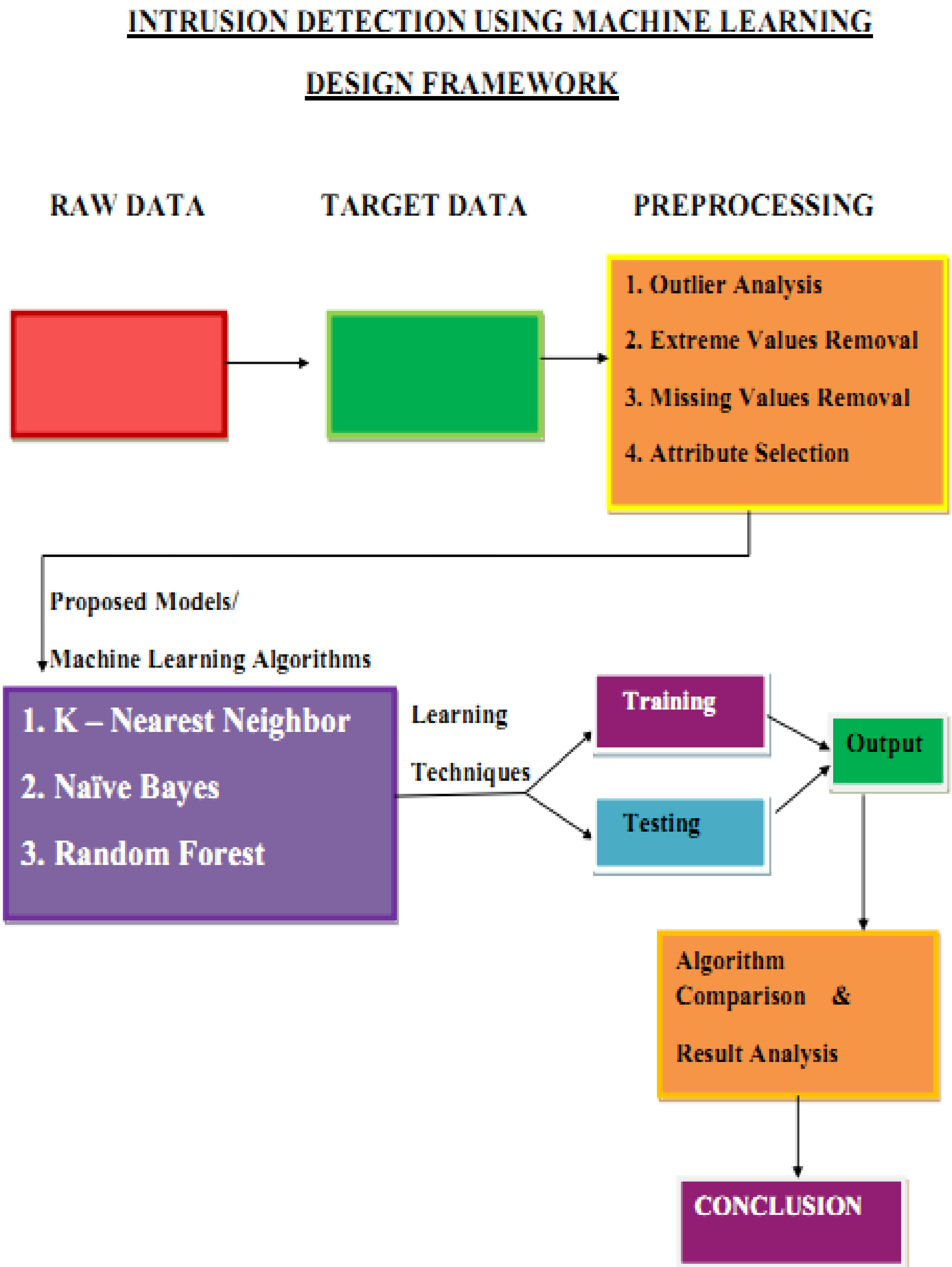
3.1.3 RANDOM FOREST

The Random Forest algorithm is a supervised classification algorithm which creates the forest with a number of trees.

The steps involved in Random Forest Algorithm:

- Load the training data set
- Randomly select K features from total m features
- Among the K features, calculate the node d using the best split point
- Split the node into daughter nodes using the best split
- Repeat the steps until I number of nodes has been reached
- End

3.2 DESIGN FRAMEWORK



3.3 DATA SET DESCRIPTION

DATA SET USED: KDDCUP99

THE KDDCUP99 DATASET:

The KDDCUP99 dataset has 41 attributes and two class label. The two different types of class labels are, **“Normal and Abnormal”.** The abnormal label has 24 different types of attacks.

This KDDCUP99 dataset has three major features.

- **Basic Features**
- **Traffic Features**
- **Content Features**

The 24 different types of attacks listed in this dataset are:

smurf.	satan.	nmap.	guess_passwd.
neptune.	portsweep.	buffer_overflow.	land.
normal.	warezclient.	warezmaster.	imap.
back.	teardrop.	rootkit.	loadmodule.
back.	teardrop.	ftp_write.	multihop.
ipsweep.	pod.	phf.	spy.

All these attacks fall into one of the four categories:

1. Denial of Service Attack
2. User to Root Attack
3. Root to Local Attack
4. Probing Attack

CHAPTER 4

ALGORITHM IMPLEMENTATION

4.1 IMPLEMENTATION OF ALGORITHMS

4.1.1 K Nearest Neighbor Sample Code

```
#Training a classifier

clf = KNeighborsClassifier(n_neighbors = 5, algorithm = 'ball_tree', leaf_size = 500)

t0 = time()

clf.fit(features, labels)

tt = time() - t0

print ("Classifier trained in {} seconds.".format(round(tt, 3)))

#Predictions on the test data

clf.fit(features, labels)

t0 = time()

pred = clf.predict(features_test)

tt = time() - t0

print ("Predicted in {} seconds".format(round(tt,3)))

#Calculating out the accuracy

from sklearn.metrics import accuracy_score

acc = accuracy_score(pred, labels_test)

print ("Accuracy is {}.".format(round(acc,4)))
```

4.1.2 Naïve Bayes Sample Code

```
clf = GaussianNB()

t0 = time()

clf.fit(features, labels)

tt = time() - t0

print ("Classifier trained in {} seconds.".format(round(tt, 3)))

from sklearn.naive_bayes import GaussianNB

labels = kdd_data_10percent['label'].copy()

labels[labels != 'normal.'] = 'attack.'

labels.value_counts()

#Predictions on the test data

clf.fit(features, labels)

t0 = time()

pred = clf.predict(features_test)

tt = time() - t0

print ("Predicted in {} seconds".format(round(tt,3)))

#Calculating out the accuracy

from sklearn.metrics import accuracy_score

acc = accuracy_score(pred, labels_test)

print ("Accuracy is {}.".format(round(acc,4)))
```

4.1.3 Random Forest Sample Code

```
#Training a classifier

from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(random_state = 0)

t0 = time()

clf.fit(features, labels)

tt = time() - t0

print ("Classifier trained in {} seconds.".format(round(tt, 3)))

from sklearn.naive_bayes import GaussianNB

labels = kdd_data_10percent['label'].copy()

labels[labels != 'normal.'] = 'attack.'

labels.value_counts()

#Predictions on the test data

clf.fit(features, labels)

t0 = time()

pred = clf.predict(features_test)

tt = time() - t0

print ("Predicted in {} seconds".format(round(tt,3)))

#Calculating out the accuracy

from sklearn.metrics import accuracy_score

acc = accuracy_score(pred, labels_test)

print ("Accuracy is {}.".format(round(acc,4)))
```

4.2 OUTPUT SCREENSHOTS

4.2.1 K Nearest Neighbor Output Screen Shots

In [3]:

```
#Initially, we will use all features
num_features = ["duration", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot", "num_failed_logins",
                "logged_in", "num_compromised", "root_shell", "su_attempted", "num_root", "num_file_creations", "num_shells",
                "num_access_files", "num_outbound_cmds", "is_host_login", "is_guest_login", "count", "srv_count", "serror_rate",
                "srv_serror_rate", "rerror_rate", "srv_rerror_rate", "same_srv_rate", "diff_srv_rate", "srv_diff_host_rate",
                "dst_host_count", "dst_host_srv_count", "dst_host_same_srv_rate", "dst_host_diff_srv_rate",
                "dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate", "dst_host_serror_rate", "dst_host_srv_serror_rate",
                "dst_host_rerror_rate", "dst_host_srv_rerror_rate"]
features = kdd_data_10percent[num_features].astype(float)
features.describe()
```

Out[3]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_com
count	494021.000000	4.940210e+05	4.940210e+05	494021.000000	494021.000000	494021.000000	494021.000000	494021.000000	494021.000000	4940
mean	47.979302	3.025610e+03	8.685324e+02	0.000045	0.006433	0.000014	0.034519	0.000152	0.148247	
std	707.746472	9.882181e+05	3.304000e+04	0.006673	0.134805	0.005510	0.782103	0.015520	0.355345	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	4.500000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
max	58329.000000	6.933756e+08	5.155468e+06	1.000000	3.000000	3.000000	30.000000	5.000000	1.000000	8

8 rows × 38 columns

```
#Predictions on the test data KNN
clf.fit(features, labels)
t0 = time()
pred = clf.predict(features_test)
tt = time() - t0
print ("Predicted in {} seconds".format(round(tt,3)))
```

```
Predicted in 417.051 seconds
```

```
#Calculating out the accuracy KNN
from sklearn.metrics import accuracy_score
acc = accuracy_score(pred, labels_test)
print ("Accuracy is {}.".format(round(acc,4)))
```

```
Accuracy is 0.9253.
```

4.2.2 Naïve Bayes Sample Code Output Screen Shots

```
In [6]: #Training a classifier
        clf = GaussianNB()
        t0 = time()
        clf.fit(features, labels)
        tt = time() - t0
        print ("Classifier trained in {} seconds.".format(round(tt, 3)))

        Classifier trained in 2.397 seconds.
```

```
In [7]: #testing
        kdd_data_corrected = pandas.read_csv("D:\ISS-Dataset\kddcup99.csv", header=None, names = col_names)
        kdd_data_corrected['label'].value_counts()

Out[7]: smurf.              164091
        normal.              60593
        neptune.             58001
        snmpgetattack.        7741
        mailbomb.             5000
        guess_passwd.         4367
        snmpguess.            2406
        satan.                1633
        warezmaster.          1602
        back.                 1098
        mscan.                1053
        apache2.               794
        processtable.          759
        saint.                 736
        portsweep.             354
        ipsweep.               306
```

```
:  #Predictions on the test data NB
   clf.fit(features, labels)
   t0 = time()
   pred = clf.predict(features_test)
   tt = time() - t0
   print ("Predicted in {} seconds".format(round(tt,3)))

   Predicted in 0.216 seconds
```

```
:  #Calculating out the accuracy NB
   from sklearn.metrics import accuracy_score
   acc = accuracy_score(pred, labels_test)
   print ("Accuracy is {}.".format(round(acc,4)))

   Accuracy is 0.8758.
```

4.2.3 Random Forest Sample Code Output Screen Shots

In [1]:
```
#loading the data
import pandas
from time import time
col_names = ["duration", "protocol_type", "service", "flag", "src_bytes", "dst_bytes", "land", "wrong_fragment", "urgent", "hot",
             "num_failed_logins", "logged_in", "num_compromised", "root_shell", "su_attempted", "num_root", "num_file_creations",
             "num_shells", "num_access_files", "num_outbound_cmds", "is_host_login", "is_guest_login", "count", "srv_count",
             "serror_rate", "srv_serror_rate", "rerror_rate", "srv_rerror_rate", "same_srv_rate", "diff_srv_rate",
             "srv_diff_host_rate", "dst_host_count","dst_host_srv_count", "dst_host_same_srv_rate", "dst_host_diff_srv_rate",
             "dst_host_same_src_port_rate", "dst_host_srv_diff_host_rate", "dst_host_serror_rate", "dst_host_srv_serror_rate",
             "dst_host_rerror_rate", "dst_host_srv_rerror_rate", "label"]
kdd_data_10percent = pandas.read_csv("D:\ISS-Dataset\kddcup99.csv", names = col_names)
kdd_data_10percent.describe()
```

Out[1]:

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_com
count	494021.000000	4.940210e+05	4.940210e+05	494021.000000	494021.000000	494021.000000	494021.000000	494021.000000	494021.000000	4940
mean	47.979302	3.025610e+03	8.685324e+02	0.000045	0.006433	0.000014	0.034519	0.000152	0.148247	
std	707.746472	9.882181e+05	3.304000e+04	0.006673	0.134805	0.005510	0.782103	0.015520	0.355345	
min	0.000000	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	4.500000e+01	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	5.200000e+02	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
75%	0.000000	1.032000e+03	0.000000e+00	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
max	58329.000000	6.933756e+08	5.155468e+06	1.000000	3.000000	3.000000	30.000000	5.000000	1.000000	8

8 rows × 38 columns

```
#Predictions on the test data RF
clf.fit(features, labels)
t0 = time()
pred = clf.predict(features_test)
tt = time() - t0
print ("Predicted in {} seconds".format(round(tt,3)))
```

```
Predicted in 0.387 seconds
```

```
#Calculating out the accuracy RF
from sklearn.metrics import accuracy_score
acc = accuracy_score(pred, labels_test)
print ("Accuracy is {}.".format(round(acc,4)))
```

```
Accuracy is 0.9278.
```

CHAPTER 5

RESULT ANALYSIS AND DESCRIPTION

5.1 ALGORITHM COMPARISON OF PROPOSED METHODS

S.No	Metrics	K Nearest Neighbor	Naïve Bayes	Random Forest
1	Accuracy	0.9253	0.8758	0.9279
2	Precision	0.89	0.85	0.87
3	Recall	0.92	0.87	0.90
4	Root Mean Square Error	0.0751	0.0831	0.0801
5	Mean Absolute Error	0.0063	0.0080	0.0072
6	True Positive Rate	0.921	0.893	0.913
7	False Positive Rate	0.08	0.012	0.09
8	True Negative Rate	0.043	0.021	0.031
9	False Negative Rate	0.061	0.081	0.068
10	F Measure	0.89	0.85	0.87
11	Training Time	1058.721 seconds	2.397 seconds	5.922 Seconds
12	Prediction Time	417.051 seconds	0.216 seconds	0.387 Seconds

Based on these metrics the identified efficient algorithm for Intrusion Detection is, “K Nearest Neighbor”.

5.2 RESULT ANALYSIS

Comparing the results obtained from the existing K Nearest Neighbor Implementation with the proposed K Nearest Neighbor Implementation.

S.No	Metrics	Existing K Nearest Neighbor	Proposed K Nearest Neighbor
1	Accuracy	0.9876	0.9253
2	Precision	0.94	0.89
3	Recall	0.98	0.92
4	Root Mean Square Error	0.0643	0.0751
5	Mean Absolute Error	0.0056	0.0063
6	True Positive Rate	0.981	0.921
7	False Positive Rate	0.02	0.08
8	True Negative Rate	0.041	0.043
9	False Negative Rate	0.059	0.061
10	F Measure	0.94	0.89
11	Training Time	962.321 Seconds	1058.721 Seconds
12	Prediction Time	109.321 Seconds	417.051 Seconds

From the comparison of the existing implementation of K Nearest Neighbor with the proposed implementation, the existing system provides more accuracy and more precision in the results than the proposed system.

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

Intrusion Detection can make a big addition to the security in today's world to avoid different types of attacks happening around. In this project, the most efficient machine learning algorithm for intrusion detection was identified as, "K Nearest Neighbor". The future works of this project will be focusing on improving the efficiency and accuracy of the algorithm than the existing implementation.

6.2 REFERENCES

[1]. Dhende, S., & Ingle, R. (2018). Network Intrusion Detection System (NIDS) using Machine Learning Perspective. *International Journal of Innovative Research in Science and Technology*, 7(1), 7644-7649.

[2]. Chowdhury, M. N., Ferens, K., & Ferens, M. (2016). Network intrusion detection using machine learning. In *Proceedings of the International Conference on Security and Management (SAM)* (p. 30). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

[3]. Alkasassebeh, M., & Almseidin, M. (2018). Machine Learning Methods for Intrusion Detection.

[4] Jamadar, R. (2018). Network Intrusion Detection System Using Machine Learning. *Indian Journal of Science and Technlogy*, 11(48).

[5]. Biswas, S. K. (2018). Intrusion Detection Using Machine Learning: A Comparison Study. *International Journal of Pure and Applied Mathematics*, *118*(19), 101-114.

[6] Kok, S., Abdullah, A., Jhanjhi, N., & Supramaniam, M. (2019). A Review of Intrusion Detection System using Machine Learning Approach. *International Journal of Engineering and Technology*, 12(1), 8-15.

[7]. Almseidin, M., Alzubi, M., Kovacs, S., & Alkasassbeh, M. (2017, September). Evaluation of machine learning algorithms for intrusion detection system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)* (pp. 000277-000282). IEEE.

[8]. Repalle, A., & Ratnam, V. (2017). Intrusion Detection System using AI and Machine Learning Algorithm. *International Journal of Engineering and Technology*, 4(12), 1709-1715.

[9]. Othman, S. M., Ba-Alwi, F. M., Alsohybe, N. T., & Al-Hashida, A. Y. (2018). Intrusion detection model using machine learning algorithm on Big Data environment. *Journal of Big Data*, *5*(1), 34.

[10]. Haq, N. F., Onik, A. R., Hridoy, M. A. K., Rafni, M., Shah, F. M., & Farid, D. M. (2015). Application of machine learning approaches in intrusion detection system: a survey. *IJARAI-International Journal of Advanced Research in Artificial Intelligence*, *4*(3), 9-18.