

A Study on the Efficacy of Machine Learning Models in Intrusion Detection Systems

Praveen Kumar Shukla¹, Dr C S Raghuvanshi², Dr Hari Om Sharan³

¹Department of computer science & Engineering, Rama University, Kanpur 209217, INDIA

Email Id: praveenshukla26@gmail.com

²Department of computer science & Engineering, Rama University, Kanpur 209217, INDIA

Email Id: drcsraghuvanshi@gmail.com

³Department of computer science & Engineering, Rama University, Kanpur 209217, INDIA

Email Id: deanengineering@ramauniversity.ac.in

Article History:

Received: 30-03-2024

Revised: 20-05-2024

Accepted: 01-06-2024

Abstract:

The electronics industry has seen a rise in demand for faster and more affordable delivery due to developments in information technology. Technology is developing quickly, which simplifies living but also presents a number of security issues. As the Internet has grown over time, so too have the amount of online attacks. The intrusion detection system (IDS) is one of the supporting layers that can be utilized for information security. IDS avoids questionable network activity and provides a pristine environment for conducting business. In the process of building an e-commerce system, the most challenging aspect is ensuring user security during online transactions. Security methods for intrusion detection were investigated in this study. The need for ongoing intrusion detection monitoring stems from the need for continued technological adaptation, which leads to a comparison of adaptive artificial intelligence-based intrusion detection systems. This paper demonstrates the use of reinforcement learning (RL) and regression learning-based intrusion detection systems (IDS) to very challenging problems, including resource allocation and input feature selection.

Keywords: reinforcement learning, linear regression, logistic regression, intrusion detection, KDD dataset.

1. INTRODUCTION

Electronic commerce (E-commerce) is the only way to meet the needs of business organizations in the modern, digital age. Delivery times are accelerated by e-commerce, which provides premium products at reduced costs. Sometimes it's referred to as paperless business information sharing. Due to its many features, including non-cash payments, round-the-clock services, improved marketing management, pre- and post-sale services, and prompt and trustworthy customer communication, business partners prefer it. It acts as a single platform for information exchange, maintaining strategies, and assisting with international company operations. Apart from these advantages, e-commerce has two main non-technical and technological drawbacks. The main technological shortcoming of the system is its security, which is the result of poor algorithm implementation. Providing security for online transactions on the user's end is a difficult task. The owner and user will suffer an unsupportable financial loss. The reason that causes this loss in e-commerce is the

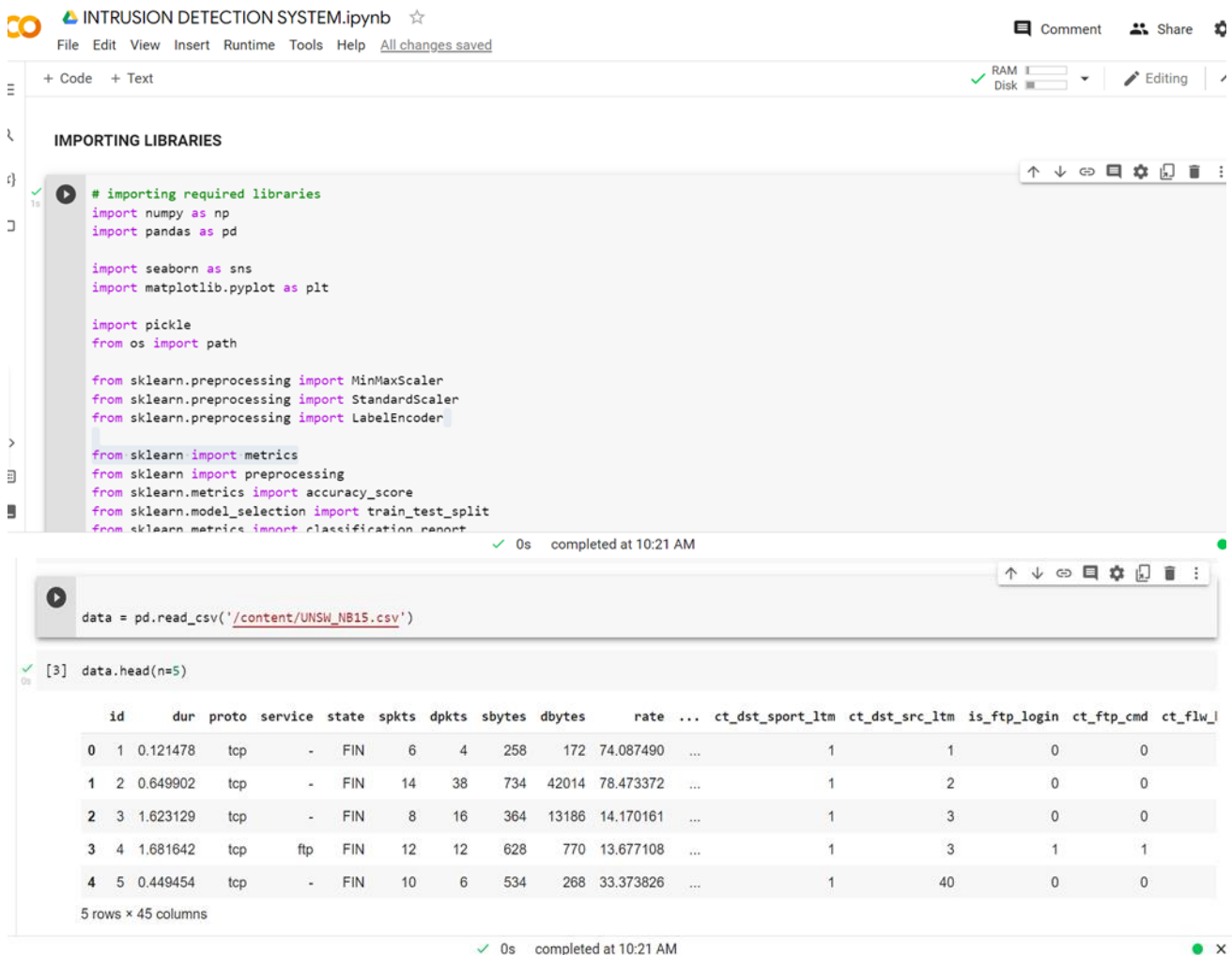
downtime brought on by an intrusion attack. As a result, this study uses intrusion detection algorithms to analyse security algorithms.

2. LITERATURE REVIEW

PAPER TITLE	RESEARCH TECHNIQUE	FUTURE SCOPE
Semi-supervised multi-layered clustering model for intrusion detection. Digital Communications and Networks 4(4): 277-286.	In this paper SMLC (an ensemble model of multiple randomized layers using K-means algorithm) is made and then this model result is compared with the known ML algorithms such as random forest, adaboost, bagging to detect the accuracy of the model. The SMLC model reduces the computational cost and can process large amounts of data.	SMLC has a relatively high testing time. In future work should be done to improve the scalability of SMLC, automation of its parameter tuning process and reduction of its testing time.
Performance Evaluation of Supervised Machine Learning Algorithms for Intrusion Detection- Procedia Computer Science 89(1): 117-123.	In this paper four ML algorithms are used on a NSL-KDDdataset and their accuracy, F-scores, precision, recalls are compared and out of the four models Random forest model has given the best results.	In future work can be extended by considering the classifiers for multiclass classification and considering only the important attributes for the intrusion detection
IDS for IoT-based smart environments: a survey Journal of Cloud Computing 7 (1): 21	In this paper the IDS system is integrated with the IOT smart applications to prevent attacks and security issues. Here the PCA (Principal component analysis) algorithm is used in IDS as it is a lightweight algorithm for IOT- based environment and descriptive multivariate method for handling quantitative data and can be extended to deal with mixed measurement level data.	In future work can be done in the areas to reduce the monitored traffic and increase the processing time and automate the detection process without human intervention.
Intrusion detection in smart cities using Restricted Boltzmann Machines.” Journal of Network and Computer Applications 135 (1): 76-83.	In this paper two main methods are used. One is the RBM(Restricted Boltzmann model) for feature extraction and the other is classifier model for attack detection.	In future work can be done to reduce the error percentage and detect variety of attacks.

3. MATERIALS AND METHODOLOGY

Step1: Import libraries and load dataset



The screenshot shows a Jupyter Notebook titled "INTRUSION DETECTION SYSTEM.ipynb". The first cell, titled "IMPORTING LIBRARIES", contains the following code:

```
# importing required libraries
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

import pickle
from os import path

from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder

from sklearn import metrics
from sklearn import preprocessing
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

The second cell contains the code to load the dataset:

```
data = pd.read_csv('/content/UNSW_NB15.csv')
```

The output of the second cell shows the first five rows of the dataset:

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cmd	ct_flw_l
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	0	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	0	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	0	
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	1	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	0	

The output also indicates "5 rows x 45 columns".

Step2: Data pre-processing

Scaling numerical Attributes: Taking away the mean and scaling to one variance will normalize the characteristics. Statistics regarding the cases in the training phase are independently targeted and adjusted by computing each attribute. The data that follows are then modified using the standard error and average that were created earlier. Many approximations require dataset uniformity; if the specific qualities do not match standard data that is normally distributed, they may perform poorly.

Pre-processing of Data

- Dataset had **45 attributes** and **175341 rows**.
- The dataset has 81173 rows and 45 attributes after the null values were removed.
- Utilizing given data type information from the features dataset, the data type of attributes is converted.

```
[4] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 175341 entries, 0 to 175340
Data columns (total 45 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   id               175341 non-null int64
1   dur              175341 non-null float64
2   proto            175341 non-null object
3   service          175341 non-null object
4   state            175341 non-null object
5   spkts            175341 non-null int64
6   dpkts            175341 non-null int64
7   sbytes           175341 non-null int64
8   dbytes           175341 non-null int64
9   rate             175341 non-null float64
10  sttl             175341 non-null int64
11  dttl             175341 non-null int64
12  sload            175341 non-null float64
13  dload            175341 non-null float64
14  sloss            175341 non-null int64
15  dloss            175341 non-null int64
```

```
40 ct_src_ltm      175341 non-null int64
41 ct_srv_dst      175341 non-null int64
42 is_sm_ips_ports 175341 non-null int64
43 attack_cat      175341 non-null object
44 label           175341 non-null int64
dtypes: float64(11), int64(30), object(4)
memory usage: 60.2+ MB

[5] data[data['service']!='-']
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp_cm
0	1	0.121478	tcp	-	FIN	6	4	258	172	74.087490	...	1	1	0	
1	2	0.649902	tcp	-	FIN	14	38	734	42014	78.473372	...	1	2	0	
2	3	1.623129	tcp	-	FIN	8	16	364	13186	14.170161	...	1	3	0	
4	5	0.449454	tcp	-	FIN	10	6	534	268	33.373826	...	1	40	0	
5	6	0.380537	tcp	-	FIN	10	6	534	268	39.417980	...	1	40	0	
...
175125	175126	0.653375	tcp	-	FIN	10	8	564	354	26.018748	...	1	1	0	

Step3: Data cleansing, normalization, and null value detection

In Python Label Encoding, the category value must be changed to a numeric value in the range of 0 to the class labels' total number, minus one. if the outcome of the category variable.

Data Normalization

- Normalization of Data 58 Data Frame's numeric columns are scaled from 0 to 1 using the Min Max Scaler.

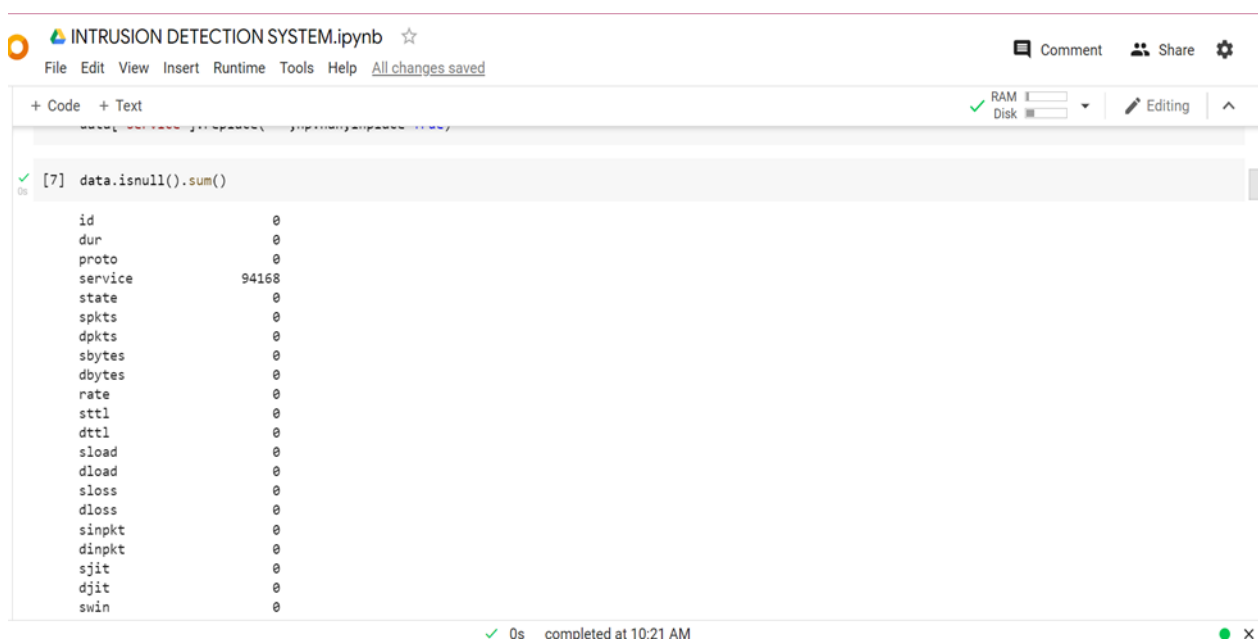
Binary Classification Preparation

- The Data Frame is duplicated for the Binary Classification process.
- The 'label' property is divided into two groups: 'normal' and 'abnormal'.

- Using Label Encoder (), the column 'label' is encoded, and the resulting encoded labels (0,1) are preserved in the column itself.
- 61 columns and 81173 rows in a binary dataset

Multi-class Classification Preparation

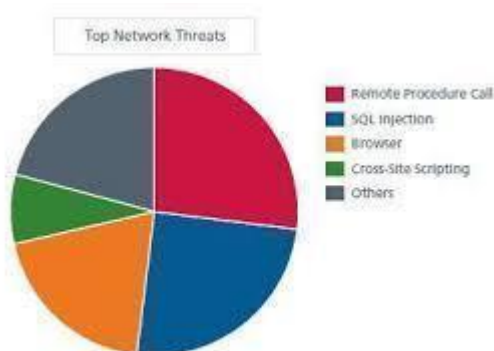
- The Data Frame is duplicated for the Multi-class Classification process.
- Nine categories are listed under the 'attack_cat' attribute: 'Analysis', 'Backdoor', 'DoS', 'Exploits', 'Fuzzers', 'Generic', 'Normal', 'Reconnaissance', and 'Worms'.
- Label Encoder () is used to encrypt attack_cat, and the accompanying encoded labels (0,1,2,3,4,5,6,7,8) are kept in the label attribute.
- attack_cat is encoded using one-hot.
- Multi-class Dataset — 81173 rows, 69 columns



```
[7]: data.isnull().sum()

id          0
dur         0
proto       0
service     94168
state       0
spkts       0
dpkts       0
sbytes      0
dbytes      0
rate        0
sttl        0
dttl        0
sload       0
dload       0
sloss       0
dloss       0
sinpkt      0
dinpkt      0
sjit        0
djit        0
swin        0
```

Step4: Classify the type of attacks



```
[9] data.dropna(inplace=True)
```

```
[10] data.shape
```

```
(81173, 45)
```

```
[11] data['attack_cat'].value_counts()
```

Generic	39496
Normal	19488
Exploits	16187
DoS	1791
Fuzzers	1731
Reconnaissance	1703
Analysis	564
Worms	114
Backdoor	99

```
[12] data['state'].value_counts()
```

0s completed at 10:21 AM

```
data
```

	id	dur	proto	service	state	spkts	dpkts	sbytes	dbytes	rate	...	ct_dst_sport_ltm	ct_dst_src_ltm	is_ftp_login	ct_ftp
3	4	1.681642	tcp	ftp	FIN	12	12	628	770	13.677108	...	1	3	1	
11	12	2.093085	tcp	smtp	FIN	62	28	56329	2212	42.520967	...	1	2	0	
15	16	0.000002	udp	snmp	INT	2	0	138	0	500000.001300	...	1	4	0	
17	18	0.393556	tcp	http	FIN	10	8	860	1096	43.195886	...	1	2	0	
21	22	0.338017	tcp	http	FIN	10	6	998	268	44.376468	...	1	1	0	
...
175335	175336	0.000006	udp	dns	INT	2	0	114	0	166666.660800	...	17	45	0	
175336	175337	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	13	24	0	
175338	175339	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	3	13	0	
175339	175340	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	14	30	0	
175340	175341	0.000009	udp	dns	INT	2	0	114	0	111111.107200	...	16	30	0	

0s completed at 10:21 AM

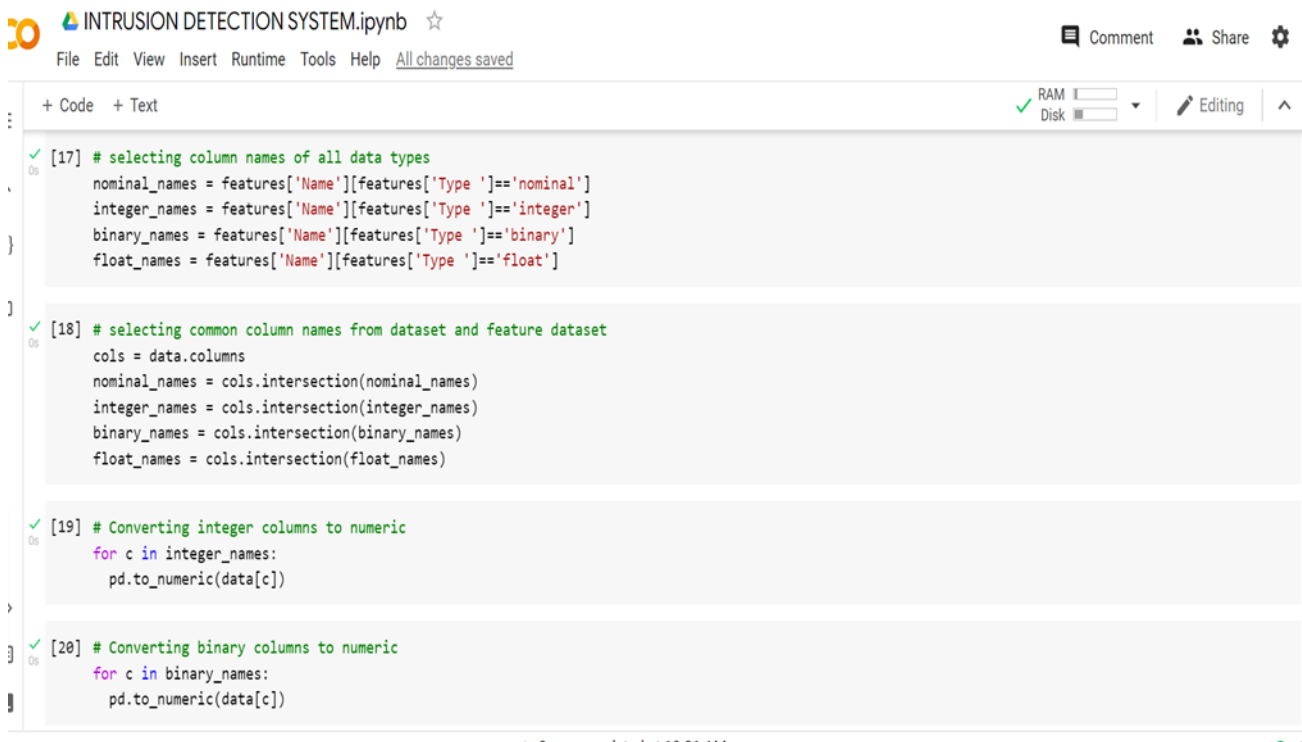
Step5: Feature Extraction:

This is the procedure of reducing the number of variables in a statistical model. The number of variables should be kept to a minimum to decrease overall computation costs and, in some cases, to improve model performance. The input parameters with the strongest correlation to the target attribute are selected after statistical analysis is used to assess the relationship between each input parameter and the goal parameter. Although the statistical measures utilized are influenced by the type of data in both the dependent and independent variables, these processes can be simple and efficient. The random forest gets estimates from each decision tree and calculates the right answer based on the number of votes cast, as opposed to relying just on one decision tree. Compared to other techniques, training takes less time with this one. Even with a large dataset, it runs quickly and

accurately predicts the outcome. Even in the absence of a sizable volume of data, accuracy can still be maintained.

Selection of features

- 'bin_data' has 61 characteristics total.
- 'multi_data' has 69 properties total.
- For feature extraction, the Pearson Correlation Coefficient approach is employed.
-
- The attributes with a correlation coefficient of more than 0.3 to the target attribute label were chosen.
- The attributes with a correlation coefficient of more than 0.3 to the target attribute label were chosen
- Number of 'bin_data' attributes after feature selection: 15
- "rate," "sttl," "sload," "dload," "ct_srv_src," "ct_state_ttl," "ct_dst_ltm," "ct_dst_src_ltm," "ct_src_ltm," "ct_srv_dst," "state_CON," "state_INT," and "label."
- After feature selection, the number of "multi_data" attributes is 16
- "dttl," "swin," "dwin," "tcprrt," "synack," "ackdat," "label," "proto_tcp," "proto_udp," "service_dns," "state_CON," "state_FIN," "attack_cat_Analysis," "attack_cat_DoS," "attack_cat_Exploits," and "attack_cat_Normal."



```
[17] # selecting column names of all data types
nominal_names = features['Name'][features['Type']=='nominal']
integer_names = features['Name'][features['Type']=='integer']
binary_names = features['Name'][features['Type']=='binary']
float_names = features['Name'][features['Type']=='float']

[18] # selecting common column names from dataset and feature dataset
cols = data.columns
nominal_names = cols.intersection(nominal_names)
integer_names = cols.intersection(integer_names)
binary_names = cols.intersection(binary_names)
float_names = cols.intersection(float_names)

[19] # Converting integer columns to numeric
for c in integer_names:
    pd.to_numeric(data[c])

[20] # Converting binary columns to numeric
for c in binary_names:
    pd.to_numeric(data[c])
```

Step 6: Splitting the Dataset into the Train & Test sets:

we separate our sample into a test set and a train set. This important data pre-processing step increases our classification model's efficacy. Suppose we train our classification model on a single sample and validate it on a second sample. Consequently, our model will struggle to comprehend

how the two models relate to one another. The model's performance will deteriorate if we give it a new dataset after it has received enough training and has a high training accuracy. A substantial amount of effort has been devoted to the development of a classification model that exhibits strong performance on the test & training datasets alike. The train set was included in the sample used to develop the classification model, and the result is now known for these samples. The term "test set" refers to a subset of the sample that is used to forecast output when the model is tested.

Splitting Dataset into Training and Testing

- Randomly splitting the **bin_data** in **80% for training** and **20% for testing**.
- Randomly splitting the **multi_data** in **70% for training** and **30% for testing**.
- Target feature — **label**

The screenshot shows a Jupyter Notebook interface with two code cells. The first cell demonstrates one-hot encoding of categorical variables using pandas. The second cell shows a minmax scaler function for normalizing numerical data.

Code Cell 1:

```
# one-hot-encoding categorical attributes using pandas.get_dummies() function
data_cat = pd.get_dummies(data_cat, columns=cat_col)
```

Output of Cell 1:

```
[30] data_cat.head()
```

	proto_tcp	proto_udp	service_dhcp	service_dns	service_ftp	service_ftp-data	service_http	service_irc	service_pop3	service_radius	service_smtp	s
3	1	0	0	0	1	0	0	0	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	1
15	0	1	0	0	0	0	0	0	0	0	0	0
17	1	0	0	0	0	0	1	0	0	0	0	0
21	1	0	0	0	0	0	1	0	0	0	0	0

Code Cell 2:

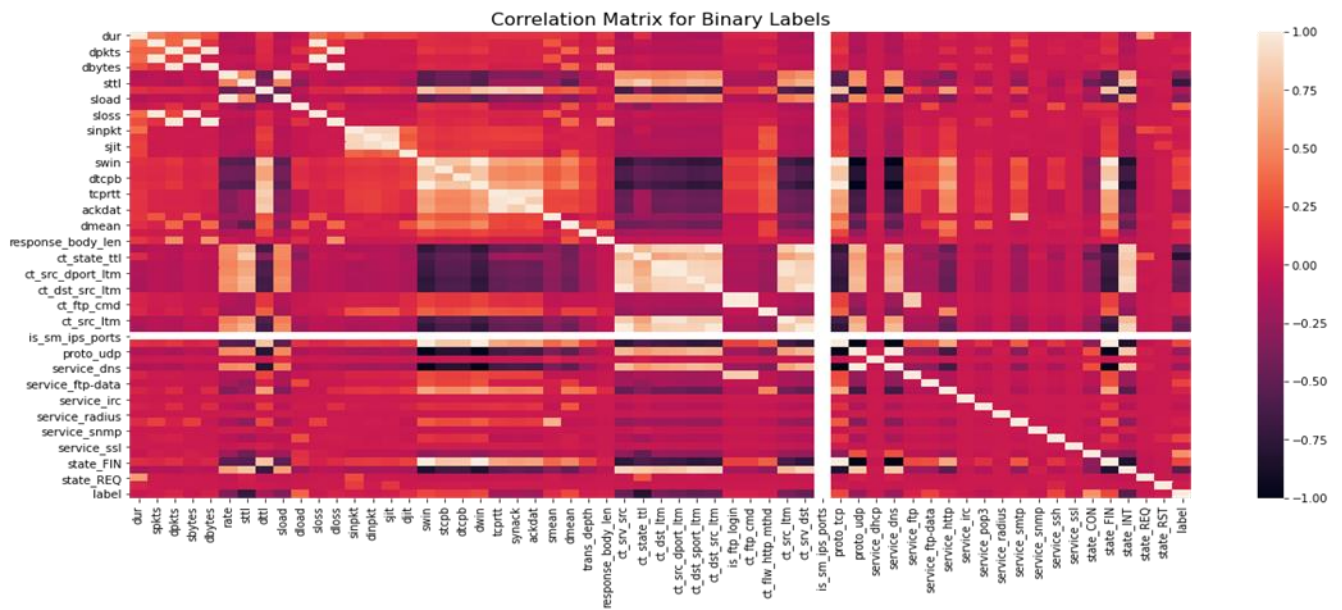
```
# using minmax scaler for normalizing data
minmax_scale = MinMaxScaler(feature_range=(0, 1))
def normalization(df,col):
    for i in col:
        arr = df[i]
        arr = np.array(arr)
        df[i] = minmax_scale.fit_transform(arr.reshape(len(arr),1))
    return df
```

Output of Cell 2:

```
[38] # data before normalization
data.head()
```

	id	dur	spkts	dpkts	sbytes	dbytes	rate	sttl	dttl	sload	...	service_radius	service_smtp	service_snmp	service_ssh	ser
3	4	1.681642	12	12	628	770	13.677108	62	252	2.740179e+03	...	0	0	0	0	
11	12	2.093085	62	28	56329	2212	42.520967	62	252	2.118251e+05	...	0	1	0	0	
15	16	0.000002	2	0	138	0	500000.001300	254	0	2.760000e+08	...	0	0	1	0	
17	18	0.393556	10	8	860	1096	43.195886	62	252	1.573347e+04	...	0	0	0	0	

completed at 10:27 AM



INTRUSION DETECTION SYSTEM.ipynb ☆

File Edit View Insert Runtime Tools Help All changes saved

RAM 16 GB Disk 16 GB Editing

```

# finding the attributes which have more than 0.3 correlation with encoded attack label attribute
corr_ybin = abs(corr_bin['label'])
highest_corr_bin = corr_ybin[corr_ybin > 0.3]
highest_corr_bin.sort_values(ascending=True)

```

sload	0.334562
dload	0.343910
rate	0.344535
ct_src_ltm	0.368486
ct_dst_ltm	0.387358
ct_src_dport_ltm	0.444874
ct_srv_dst	0.459984
ct_srv_src	0.463153
ct_dst_src_ltm	0.463735
ct_dst_sport_ltm	0.497234
state_INT	0.546631
state_CON	0.552505
sttl	0.707337
ct_state_ttl	0.801403
label	1.000000

Name: label, dtype: float64

Step7: Comparison of Linear Regression, Logistic Regression and RL.

Linear regression models

It is possible to look at the link between one or more independent variables and a continuous dependent variable. If there is only one independent variable and one dependent variable, then simple linear regression is used. When there are more than two factors, though, multiple linear regression is used instead of simple linear regression. In all types of linear regression, the least squares method is used to find the line that fits a set of data the best. Logistic regression, like linear regression, can be used to foretell numbers for categorical variables. This method is used to evaluate the correlation between several predictors and a single outcome. Categorical variables can take on the values true, false, yes, no, 1, 0, and other values. The S-curve is transformed into a straight line by the logit

function, and the unit of measurement in linear regression is a probability. In regression analysis, both models are used to make predictions about the future. However, linear regression is often easier to understand. In addition, whereas a smaller sample size is sufficient for linear regression to accurately represent values across all answer categories, a larger sample size is necessary for logistic regression. Should a more extensive, representative sample be lacking, the model's statistical potential to identify a noteworthy effect may be compromised.

Linear Regression Model

Binary Classification

- Accuracy — **97.80720665229443**
- Mean Absolute Error — **0.021927933477055742**
- Mean Squared Error — **0.021927933477055742**
- Root Mean Squared Error — **0.1480808342664767**
- R2 Score — **88.20923868071647**

Multi-class Classification

- Accuracy — **95.12976346911958**
- Mean Absolute Error — **0.06824901445466491**
- Mean Squared Error — **0.12146846254927726**
- Root Mean Squared Error — **0.3485232596962178**
- R2 Score — **91.82055676180129**

```
LINEAR REGRESSION

✓ [67]
Ds X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=0.20, random_state=50)

✓ [68] lr_bin = LinearRegression(normalize=False)
Ds lr_bin.fit(X_train, y_train)

/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_base.py:155: FutureWarning: 'normalize' was deprecated in version 1.0 and will be removed
FutureWarning,
LinearRegression(normalize=False)

✓ [69] y_pred = lr_bin.predict(X_test)
```

```
[69] y_pred = lr_bin.predict(X_test)
```

```
[70] round = lambda x:1 if x>0.6 else 0
vfunc = np.vectorize(round)
y_pred = vfunc(y_pred)
```

```
[71] print("Mean Absolute Error - ", metrics.mean_absolute_error(y_test, y_pred))
print("Mean Squared Error - ", metrics.mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error - ", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
print("R2 Score - ", metrics.explained_variance_score(y_test, y_pred)*100)
print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```

Mean Absolute Error - 0.021927933477055742
 Mean Squared Error - 0.021927933477055742
 Root Mean Squared Error - 0.1480808342664767
 R2 Score - 88.20923868071647
 Accuracy - 97.80720665229443

```
[72] cls_report= classification_report(y_true=y_test, y_pred=y_pred,target_names=le1.classes_)
```

INTRUSION DETECTION SYSTEM.ipynb ☆

File Edit View Insert Runtime Tools Help *All changes saved*

+ Code + Text RAM
Disk Editing ^

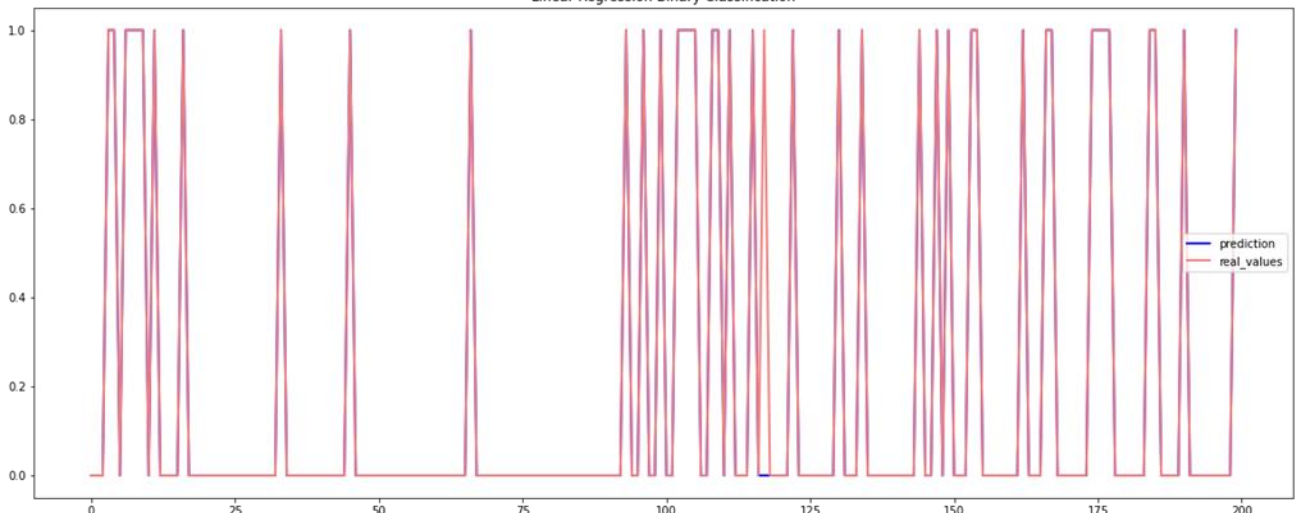
```
[73] print(cls_report)
```

	precision	recall	f1-score	support
abnormal	0.97	1.00	0.99	12326
normal	0.99	0.91	0.95	3909
accuracy			0.98	16235
macro avg	0.98	0.96	0.97	16235
weighted avg	0.98	0.98	0.98	16235

```
[74] lr_bin_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
lr_bin_df.to_csv('lr_real_pred_bin.csv')
lr_bin_df
```

	Actual	Predicted
159889	0	0
125506	0	0
158979	0	0

Linear Regression Binary Classification



Logistic regression

Among the family of supervised machine learning models is logistic regression. It is also considered a discriminative model, meaning that it aims to differentiate between multiple classes. Naive Bayes and other generative algorithms can make examples of the class they are trying to predict, but this one can't (for example, a cat picture). We already talked about how the model's beta values are made by logistic regression, which maximises the log likelihood function. This slightly alters when looking at it from a machine learning standpoint. In machine learning, the global maximum is found using gradient descent, with the loss function being the negative log likelihood. This is merely a different method to get the same approximations as before. Moreover, logistic regression may lead to overfitting, especially if the model has a large number of predictor variables. In high dimensionality models, regularisation is usually employed to penalise parameters with large coefficients. For additional information on the logistic regression machine learning model, go visit Scikit-learn (link is external to IBM).

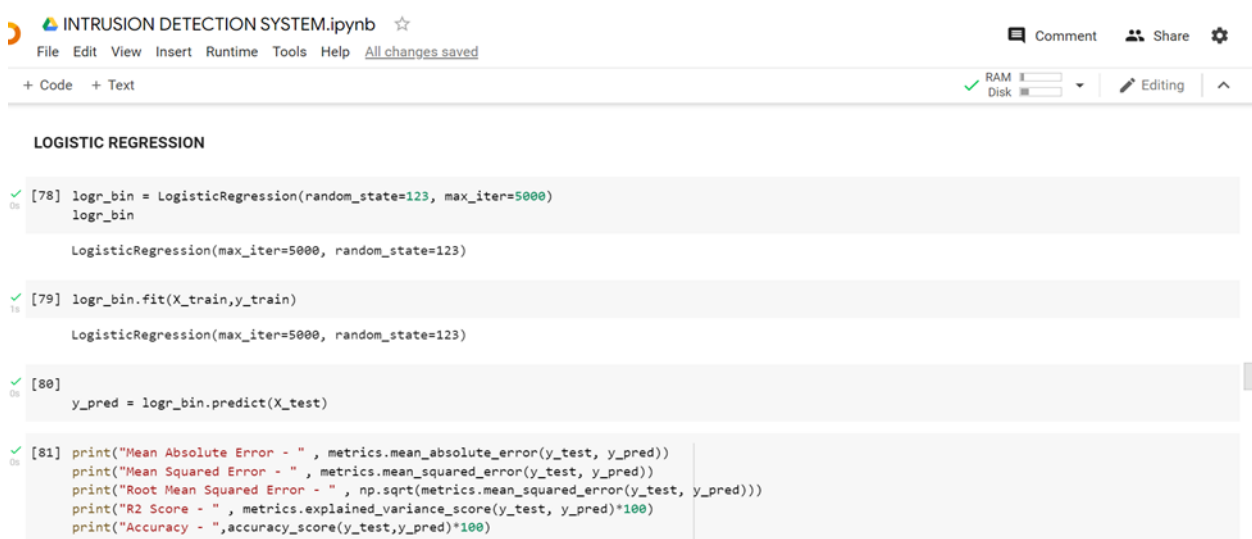
Logistic Regression Model

Binary Classification

- Accuracy — **97.80104712041884**
- Mean Absolute Error — **0.02198952879581152**
- Mean Squared Error — **0.02198952879581152**
- Root Mean Squared Error — **0.1482886671186019**
- R2 Score — **88.17947258428785**

Multi-class Classification

- Accuracy — **97.58952036793693**
- Mean Absolute Error — **0.060077201051248356**
- Mean Squared Error — **0.18056011826544022**
- Root Mean Squared Error — **0.42492366169165047**
- R2 Score — **87.87674567880146**



```
LOGISTIC REGRESSION

[78] logr_bin = LogisticRegression(random_state=123, max_iter=5000)
      logr_bin
      LogisticRegression(max_iter=5000, random_state=123)

[79] logr_bin.fit(X_train,y_train)
      LogisticRegression(max_iter=5000, random_state=123)

[80]
      y_pred = logr_bin.predict(X_test)

[81] print("Mean Absolute Error - ", metrics.mean_absolute_error(y_test, y_pred))
      print("Mean Squared Error - ", metrics.mean_squared_error(y_test, y_pred))
      print("Root Mean Squared Error - ", np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
      print("R2 Score - ", metrics.explained_variance_score(y_test, y_pred)*100)
      print("Accuracy - ",accuracy_score(y_test,y_pred)*100)
```



Logistic regression is commonly applied to problems involving classification and forecasting through the use of cases. These use cases consist of:

•Fraud detection:

Teams can use logistic regression models to find data abnormalities that point to fraudulent activity. Banking and other financial organisations may discover that specific behaviours or qualities are more often linked to fraudulent operations in an effort to better protect their consumers. SaaS-based organisations also employ these techniques to eliminate fictitious user accounts from datasets during business performance data analysis.

•Disease prediction:

This analytics technique can be implemented in medicine to predict the likelihood that a given population will become unwell or sick. Health care facilities can arrange for prophylactic care for individuals who are more susceptible to a specific disease.

•Churn prediction:

Certain behaviours may signal churn in different organisational activities. If high achievers are in danger of leaving the organisation, human resources and management teams may be interested in learning about it. Information of this kind could start conversations regarding the company's compensation policies or culture. Alternatively, the sales staff would like to know which of its clients might choose to work with another company. Teams might be inspired by this to develop a retention plan in order to prevent revenue loss.

Types of logistic regression Logistic regression models based on categorical responses come in three varieties.

•Binary logistic regression:

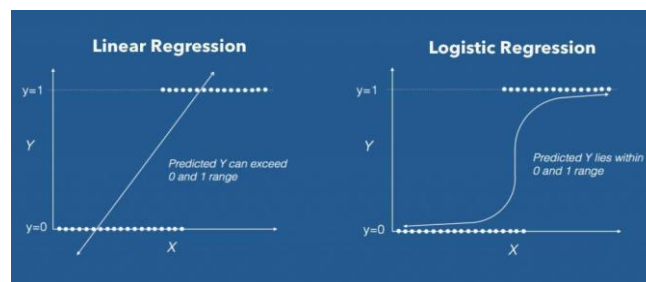
This method's dependent variable, or answer, is dichotomous, meaning that there are only two possible outcomes (e.g., 0 or 1). It can determine the malignancy of a tumour or whether an email is spam, among other applications. This technique is widely used in the field of logistic regression and is widely recognised as a top-tier classifier for binary data.

•Multinomial logistic regression:

Within this particular variant of logistic regression, the dependent variable is open to a maximum of three values, none of which are sequentially arranged. To improve their ability to sell their films, for instance, film studios endeavour to predict the genre of film that a given audience is most likely to enjoy. To determine the degree to which age, relationship status, gender, and age influence the genre of film that an individual prefers, the studio may implement a multinomial logistic regression model. As a result, the studio may opt to focus its marketing endeavours on the specific demographic that is relevant to a given film.

•Ordinal logistic regression:

Despite the set order of values shown, this type of logistic regression model is used when the response variable has three or more possible outcomes. Ordinal responses can be expressed as grading schemes or rating scales ranging from 1 to 5.



Multi-class Classification

- Accuracy — **97.59362680683311**
- Mean Absolute Error — **0.059912943495400786**
- Mean Squared Error — **0.17941031537450722**
- Root Mean Squared Error — **0.42356854861345317**
- R2 Score — **87.93449282205455**

• **SVM algorithm**

- 1. Input:** The algorithm takes a labelled dataset as input, where each data point consists of a set of features (X) & a corresponding binary class label (Y) indicating one of two classes (e.g., 0 or 1).
- 2. Data Preparation:** If needed, feature scaling and data normalization can be performed to ensure that all features have a similar scale, which can improve the convergence and performance of the algorithm.
- 3. Feature Space Transformation Kernel Trick - Optional:** The initial feature space can be changed into a higher-dimensional space using a kernel function in SVM. This enables SVM to identify a decision boundary that may have been nonlinear in the original space but is now linear in the modified space. The linear kernel, polynomial kernel, sigmoid kernel, and radial basis function RBF kernel are examples of common kernel functions.
- 4. Margin Maximization:** Finding the hyperplane with the largest margin between the two classes is the primary goal of SVM. The margin is the distance between the hyperplane and the closest data points of each class support vectors. The optimal hyperplane is the one that achieves the largest margin while correctly classifying the data points.
- 5. Soft Margin Optional:** SVM introduces slack variables to allow for a soft margin in situations where the data is not perfectly separable. These slack variables allow some data points to be misclassified or fall within the margin region, but with a penalty for misclassification. The objective is to balance the margin maximization and the classification error.
- 6. Optimization Problem:** The SVM technique frames the issue as a convex optimisation problem in order to determine the ideal weights and biases for the hyperplane. The optimisation seeks to maximise the margin and minimise the classification error of incorrectly categorised data points.
- 7. Kernel Parameters Optional:** In the event that a kernel function is employed, the SVM algorithm may further entail fine-tuning the kernel parameters, such as the polynomial kernel's degree or the RBF kernel's width, in order to maximise performance on the training set.
- 8. Training:** In the training phase, the algorithm finds the ideal hyperplane parameters by solving the optimisation issue using methods such as quadratic programming or Sequential Minimal Optimisation (SMO).
- 9. Prediction:** The model can be used to predict the class label for fresh data points once it has been trained. The predicted class label is determined by comparing the signed distance of the data point to the hyperplane. If the signed distance is positive, the data point belongs to one class; otherwise, it belongs to the other class.
- 10. Output:** For binary classification tasks, the Support Vector Machine technique provides an output of the learnt model parameters (weights and bias). It is well known that SVM can handle high-dimensional data and find difficult decision boundaries, making it a reliable classification

system. When the data are clear and the categories are distinct, this method shines. Using techniques such as One-vs-One (OvO) or One-vs-Rest OvR strategies, SVM can be adapted to perform multiclass classification. In addition, the Support Vector Regression SVR variant of SVM can be used for regression assignments.

4. RESULTS AND DISCUSSION

The assessment of the adaptive neural network approach to intrusion detection involved conducting three experiments using the prototype application. The effectiveness of several algorithms, such as SVM, linear regression, & logistic regression, was assessed in these trials.

The results clearly demonstrated that the support vector machine consistently provided higher accuracy compared to both linear and logistic regression models. This suggests that the adaptive neural network approach, when implemented using a support vector machine, holds significant promise for improving IDS ' effectiveness and accuracy.

5. CONCLUSION

IDS are essential for protecting computer and network-based systems because they continuously monitor protocol traffic for possible breaches and assaults. The adaptability of IDS allows for adjustments in response to both external and internal threats. In this study, we conducted a comparative analysis of IDS using linear logistic regression & SVM.

In conclusion, the experiments conducted using the prototype application to assess the viability of the adaptive neural network approach for intrusion detection revealed valuable insights. Specifically, the support vector machine algorithm consistently outperformed linear and logistic regression models in terms of accuracy. This finding underscores the potential effectiveness of the adaptive neural network approach for enhancing IDS.

These findings imply that more investigation and advancement in this area may result in IDS that are more reliable and accurate. Implementing adaptive neural networks, particularly with SVM, holds promise for addressing the evolving challenges in cybersecurity and bolstering our ability to detect and respond to intrusions effectively. However, additional investigations and real-world testing are warranted to fully harness the capabilities of this approach and validate its practicality in various security contexts.

ACKNOWLEDGMENT

In America, the word "acknowledgment" is most often spelled without a "g" following the "e." Steer clear of the stuttering phrase "One of us (R. B. G.) thanks." Say "R.B.G. thanks..." instead. Include acknowledgements of sponsors in the first page's unnumbered footnote.

REFERENCES

- [1] Aburomman, A. A., &Reaz, M. B. I. (2016) "Ensemble of binary SVM classifiers based on PCA and LDA feature extraction for intrusion detection."Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC): 636-640.
- [2] Al-Jarrah, O. Y., Al-Hammdi, Y., Yoo, P. D., Muhaidat, S., & Al-Qutayri, M. (2018) "Semi-supervised multi-layered clustering model for intrusion detection." Digital Communications and Networks 4(4): 277-286.

- [3] Al-Yaseen, W. L., Othman, Z. A., & Nazri, M. Z. A. (2017) "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system." *Expert Systems with Applications* 67(1): 296-303.
- [4] An, X., Su, J., Lü, X., & Lin, F. (2018) "Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system." *EURASIP Journal on Wireless Communications and Networking* 249 (1): 1-9.
- [5] Belavagi, M. C., & Muniyal, B. (2016) "Performance evaluation of supervised machine learning algorithms for intrusion detection." *Procedia Computer Science* 89(1): 117-123.
- [6] Elrawy, M. F., Awad, A. I., & Hamed, H. F. (2018) "Intrusion detection systems for IoT-based smart environments: a survey." *Journal of Cloud Computing* 7 (1): 21
- [7] Elsaedy, A., Munasinghe, K. S., Sharma, D., & Jamalipour, A. (2019) "Intrusion detection in smart cities using Restricted Boltzmann Machines." *Journal of Network and Computer Applications* 135 (1): 76-83.
- [8] Ganapathy, S., Kulothungan, K., Muthurajkumar, S., Vijayalakshmi, M., Yogesh, P., & Kannan, A. (2013) "Intelligent feature selection and classification techniques for intrusion detection in networks: a survey." *EURASIP Journal on Wireless Communications and Networking* (1): 271.
- [9] Gautam, R. K. S., & Doegar, E. A. (2018) "An Ensemble Approach for Intrusion Detection System Using Machine Learning Algorithms." *International Conference on Cloud Computing, Data Science & Engineering (Confluence)*: 14-15.
- [10] Ghasempour, A. (2019). *Internet of Things in Smart Grid: Architecture, Applications, Services, Key Technologies, and Challenges*. *Inventions* 4(1): 22.