Project School

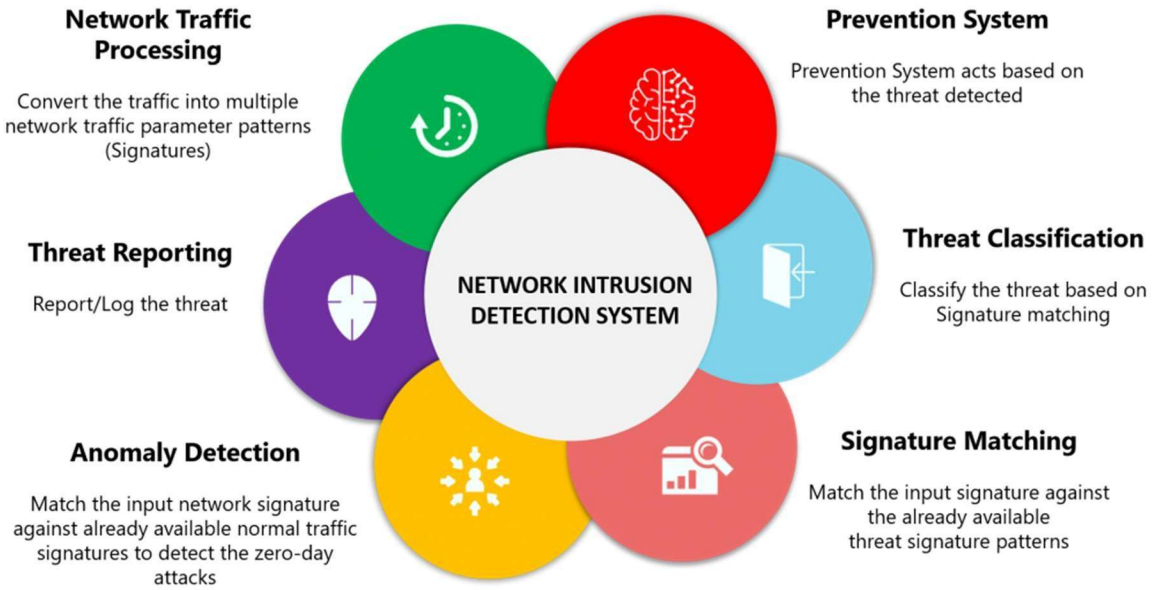
Project Title: NETWORK INTRUSION DETECTION SYSTEM
Faculty Incharge: Dr. Rajasekaran.
Session Duration: 10-09-2022 to 17-12-2022

Name: Shubham Mola.
Roll No:20BD1A057H.
Class: CSE-D.

Signature of student

Signature of Mentor.

Domain:



Project Description:

This web application helps to identify an attack or sense abnormal behavior in the network and send an alert to the user and protect the user. When the user login into the portal he gets the information about the network's accuracy, f1-score, precision. This helps the user to detect how safe his network is for the system.

Introduction:

A network intrusion refers to any forcible or unauthorized activity on a digital network. Attacks can broadly classified into four major categories:

- DOS(Denial of Service)
- Probe
- R2L(Remote-to-User)
- U2R(User-to-Root)

Network intrusion detection systems are placed at a strategic point within the network to examine traffic from all devices on the network. Once it identifies an attack or senses abnormal behavior, it sends an alert to the user.

Technologies Used:

1. Model Building:
    - Preprocessing: LabelEncoder (in sklearn.preprocessing)
    - Normalization: MinMaxScalar (in sklearn.preprocessing)
2. Web Integration:
    - Stack Selected: MERN (MongoDB, Express, React ,Node)
3. Advanced Features:
   Using Google Earth 2.0 for user authentication and storing user details in a hash in MongoDB.
4. Database:
   Data is stored in MongoDB and can be imported and exported. The data is safe and secured.
5. Backend :
   The backend is developed using Python.

Dataset Used:

NSL-KDD

- NSL-KDD is a new version of the KDD'99 data set.
- This is an effective benchmark data set to help researchers compare different intrusion detection methods.

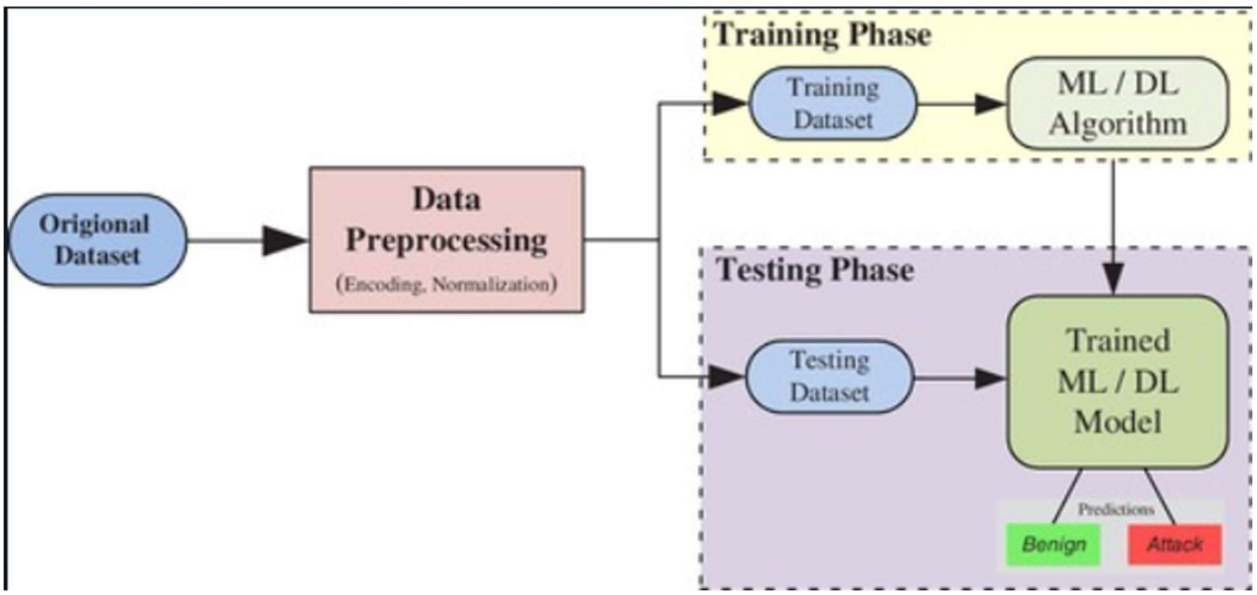
Data files:

- KDDTrain+.ARFF: The full NSL-KDD train set with binary labels in ARFF format.
- KDDTrain+.TXT: The full NSL-KDD train set including attack-type labels and difficulty level in CSV format.
- KDDTrain+_20Percent.ARFF: A 20% subset of the KDDTrain+.arff file.
- KDDTrain+_20Percent.TXT: A 20% subset of the KDDTrain+.txt file.
- KDDTest+.ARFF: The full NSL-KDD test set with binary labels in ARFF format.
- KDDTest+.TXT: The full NSL-KDD test set including attack-type labels and difficulty level in CSV format.
- KDDTest-21.ARFF: A subset of the KDDTest+.arff file which does not include records with a difficulty level of 21 out of 21.
- KDDTest-21.TXT: A subset of the KDDTest+.txt file which does not include records with a difficulty level of 21 out of 2.

Algorithms:

- KNN - K-Nearest Neighbors is one of the simplest Machine Learning algorithms. The K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the category that is most similar to the available categories.
- Random Forest - Random Forest is a popular machine learning algorithm. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.
- CNN - Convolutional Neural Networks are neural networks that share their parameters. It is a Deep Learning Algorithm.
- LSTM - Long Short Term Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step.

Process Flow:



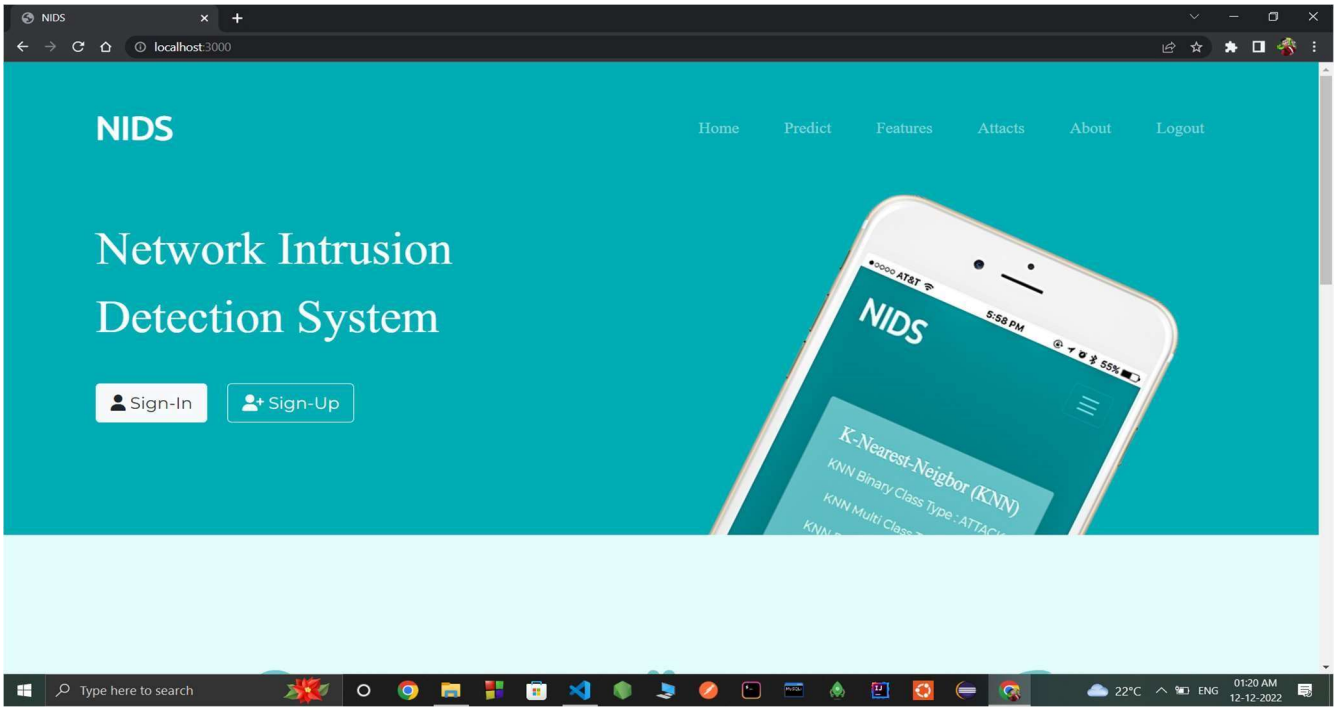
Code:

```python
import numpy as np
import sys
from sklearn.metrics import accuracy_score, confusion_matrix
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
import sklearn
from sklearn.neighbors import KNeighborsClassifier
import os
from sklearn.preprocessing import LabelEncoder
import tensorflow as tf
from sklearn.preprocessing import Normalizer
import pickle
data_Validate=pd.read_csv('fs_new validation project.csv')
columns = (['protocol_type','service','flag','logged_in','count','srv_serror_rate','srv_rerror_rate','same_srv_rate','diff_srv_rate','dst_host_co
data_Validate.columns=columns
protocol_type_le = LabelEncoder()
service_le = LabelEncoder()
flag_le = LabelEncoder()
data_Validate['protocol_type'] = protocol_type_le.fit_transform(data_Validate['protocol_type'])
data_Validate['service'] = service_le.fit_transform(data_Validate['service'])
data_Validate['flag'] = flag_le.fit_transform(data_Validate['flag'])
df_validate=data_Validate.copy(deep=True)
x_validate=df_validate.drop(['attack'],axis=1)
y_validate=pd.DataFrame(df_validate['attack'])
label_encoder = LabelEncoder()
scaler=MinMaxScaler()
x1=x_validate.copy(deep=True)
scaler=MinMaxScaler()
scaler.fit(x1)
scaled_data=scaler.transform(x1)
scaled_data=pd.DataFrame(scaled_data)
```

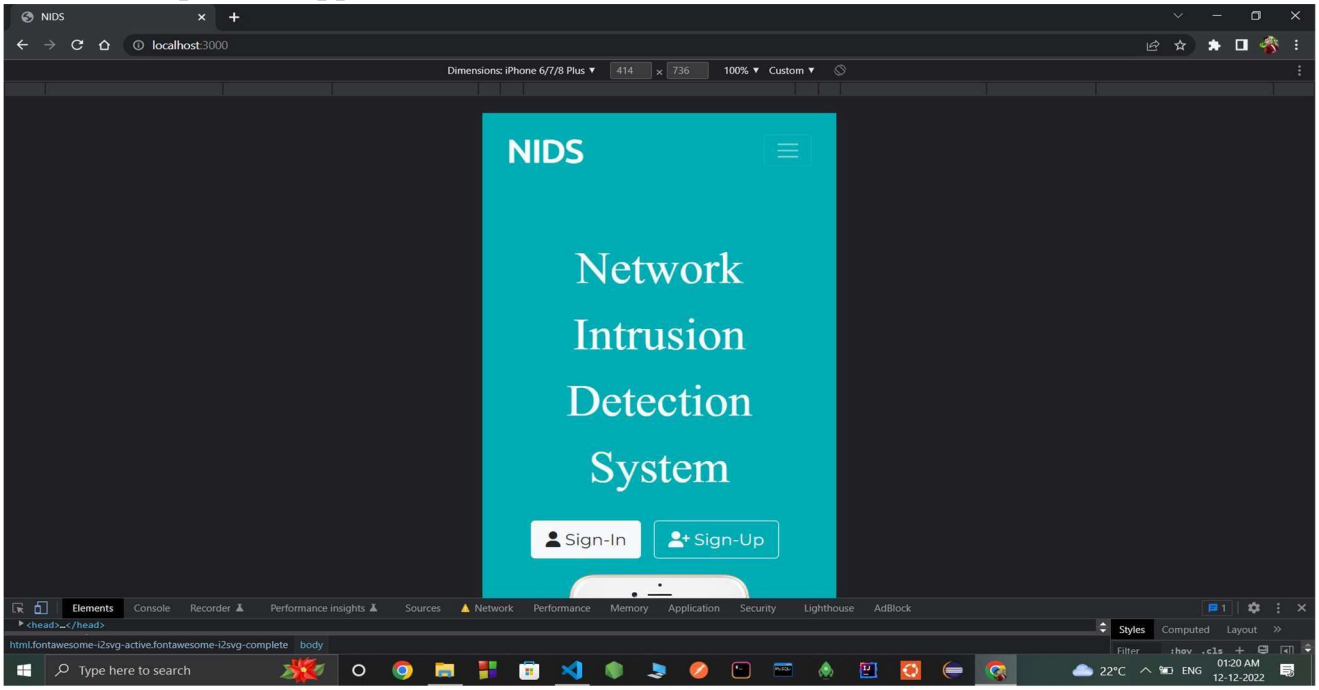
```python
l1 = preprocessing.Normalize(l1)
knn_bin = pickle.load(open('knn_binary_class.sav', 'rb'))
knn_multi = pickle.load(open('knn_multi_class.sav', 'rb'))
randfor_bin = pickle.load(open('random_forest_binary_class.sav', 'rb'))
randfor_multi = pickle.load(open('random_forest_multi_class.sav', 'rb'))
cnn_bin= tf.keras.models.load_model('latest_cnn_bin.h5')
cnn_multi= tf.keras.models.load_model('latest_cnn_multiclass.h5')
lstm_bin= tf.keras.models.load_model('lstm_latest_bin.h5')
lstm_multi= tf.keras.models.load_model('lstm_latest_multiclass.h5')
val_knn=knn_bin.predict(l1)
if(val_knn[0]==0):
    print('KNN algorithm binary class:Normal')
else:
    print('KNN algorithm binary class:Attack')
#print('KNN Algorithm multi class:',knn_multi.predict(l1))
tp_knn=knn_multi.predict(l1)
for i in tp_knn:
    tp_knn=i

print('KNN Multi Class Type : ',tp_knn)
if(tp_knn=='dos'):
   print('KNN  Description : A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessi
elif(tp_knn=='probe'):
   print('KNN  Description : Probing is another type of attack in which the intruder scans network devices to determine weakness in top
elif(tp_knn=='r2l'):
   print('KNN Description : Remote to user (R2L) is a type of computer network attacks, in which an intruder sends set of packets to an
elif(tp_knn=='u2r'):
   print('KNN  Description : User to root attacks (U2R) is an another type of attack where the intruder tries to access the network res
else:
    print('KNN Description : Data is safe')
val_rnd=randfor_bin.predict(l1)
if(val_rnd[0]==0):
```

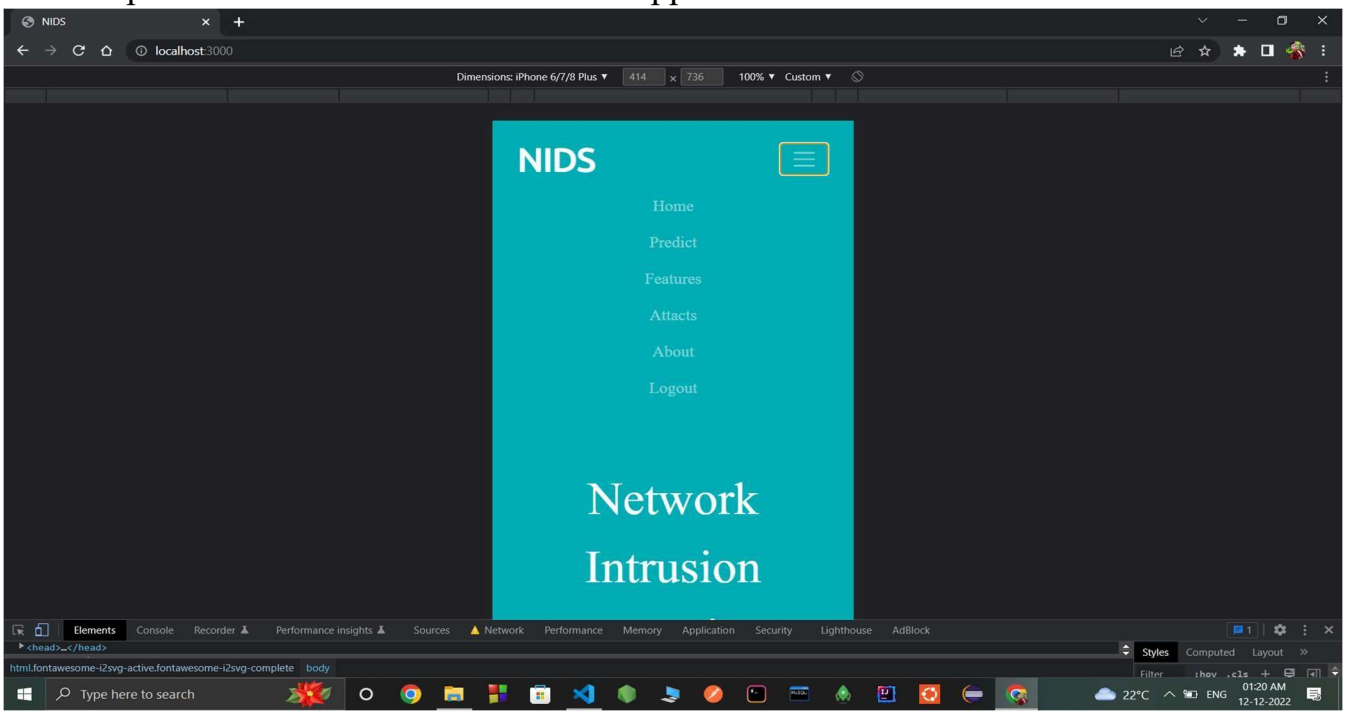
Desktop/Laptop Compatible Application screen:



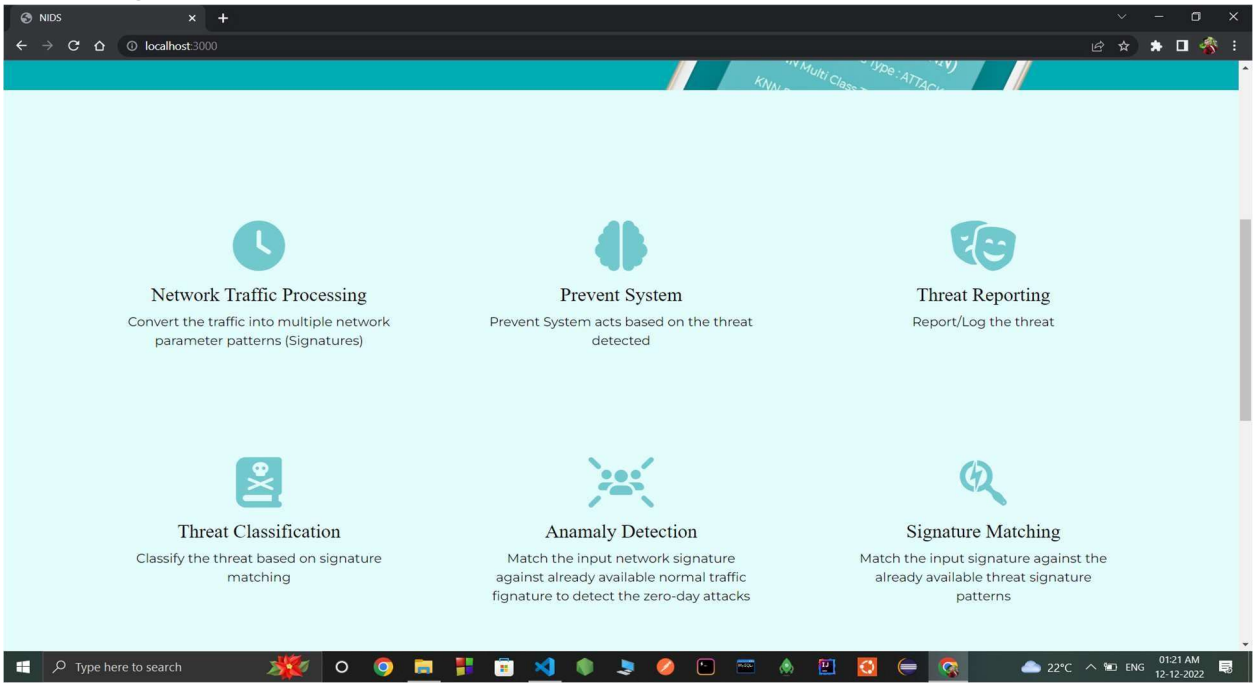
Mobile Compatible Application Screen



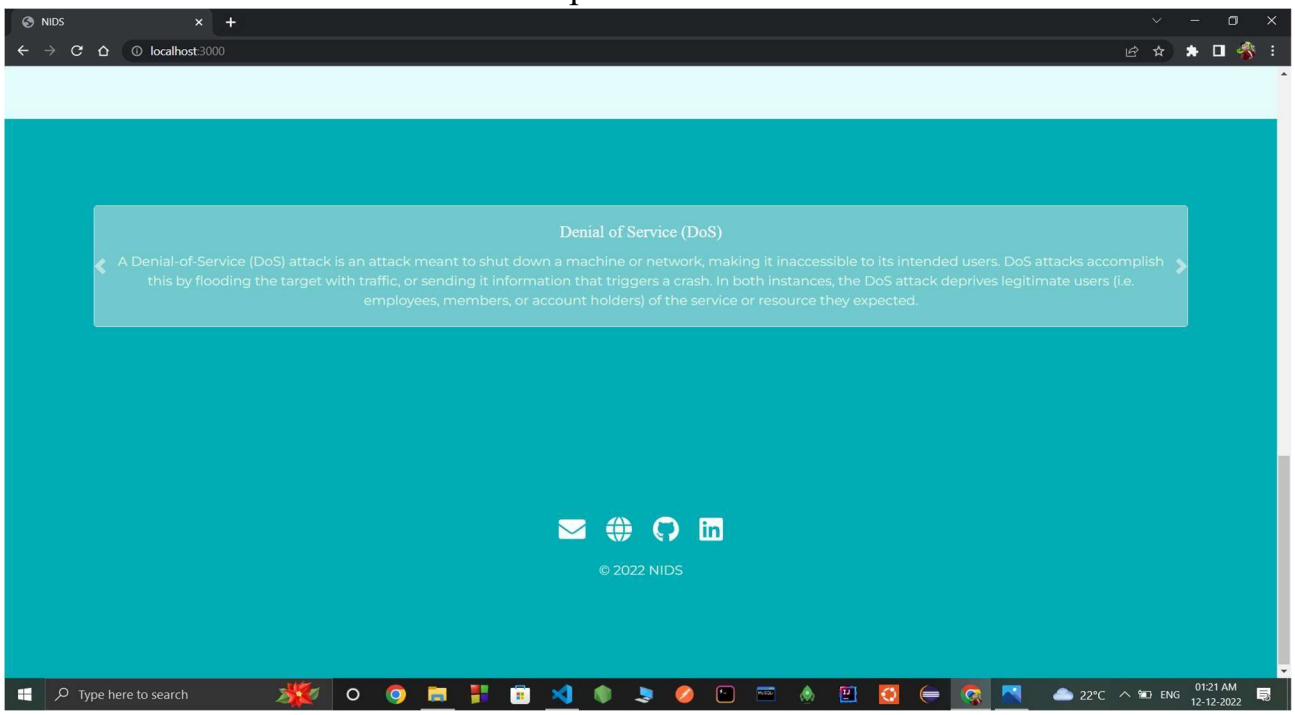
Services provided with a button on mobile app



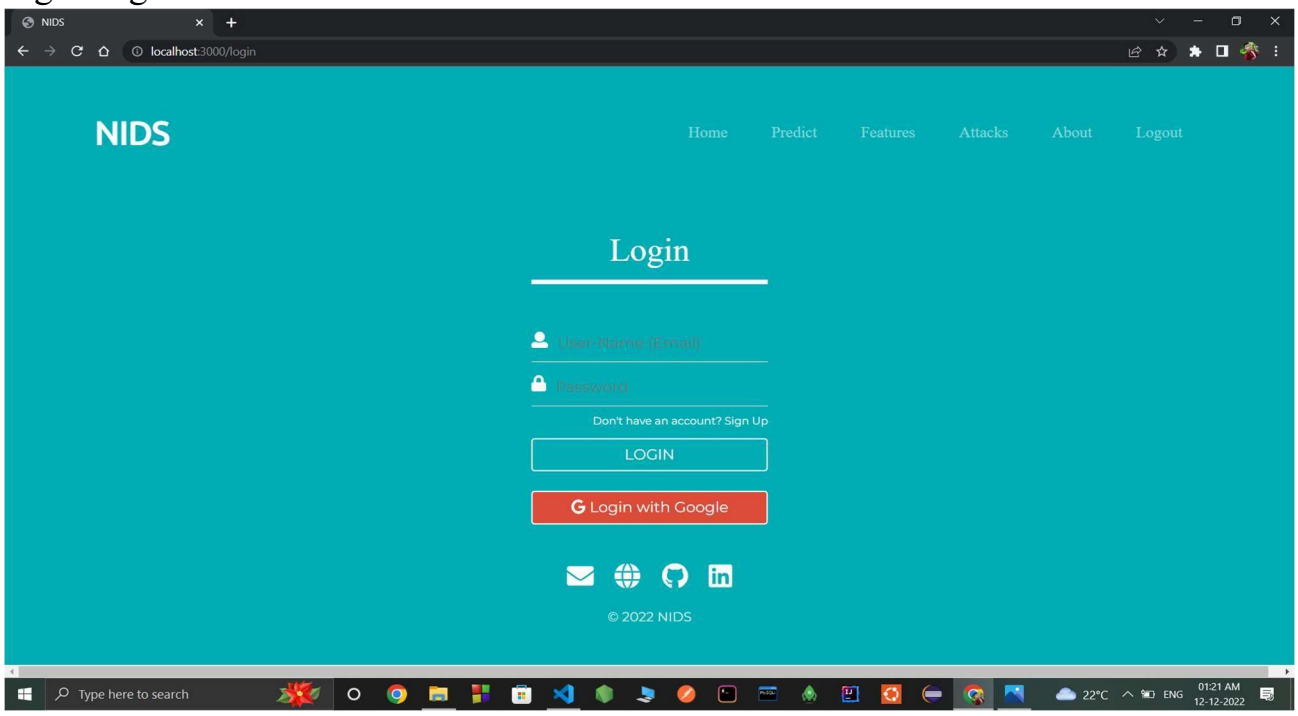
Home Page



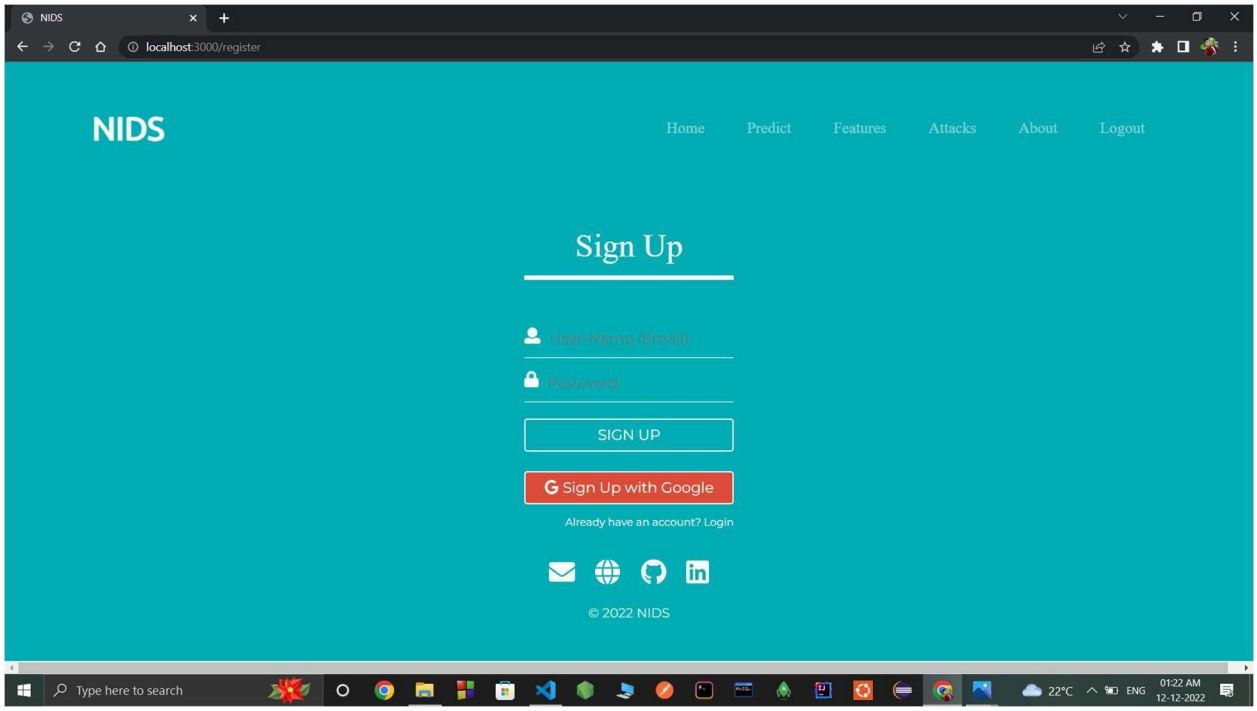
Information of each attack with the help Carousel



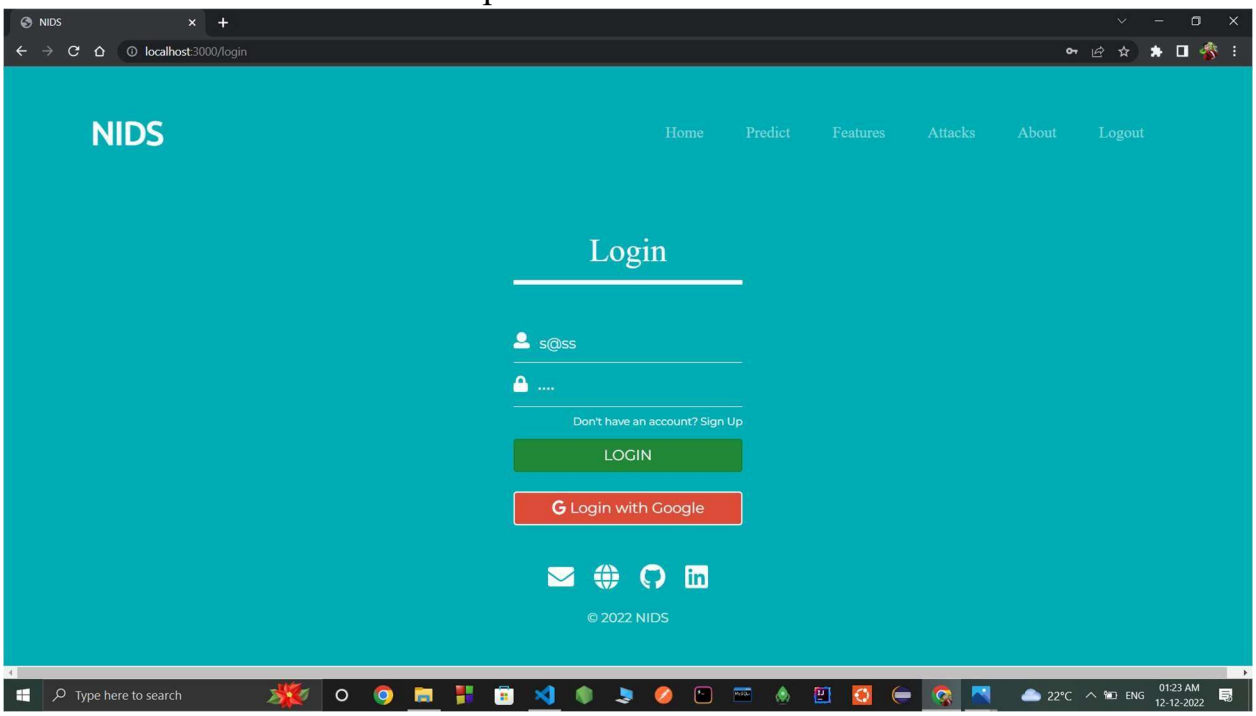
Login Page



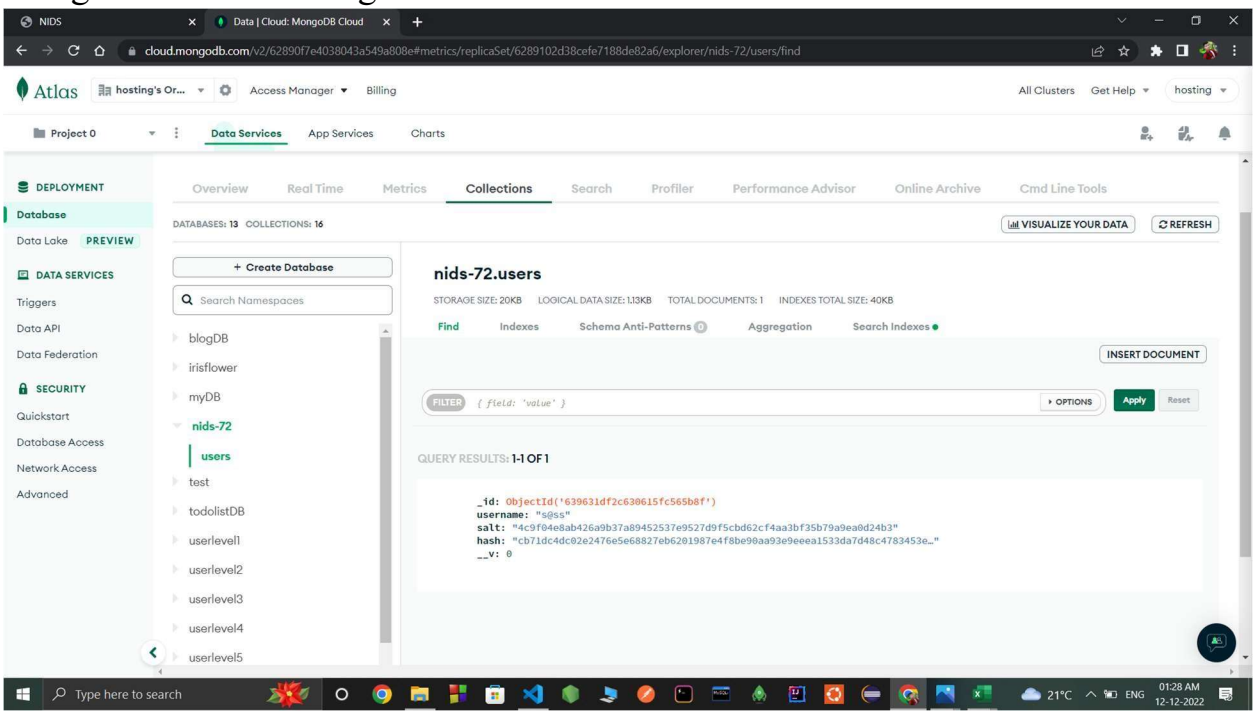
Sign Up Page



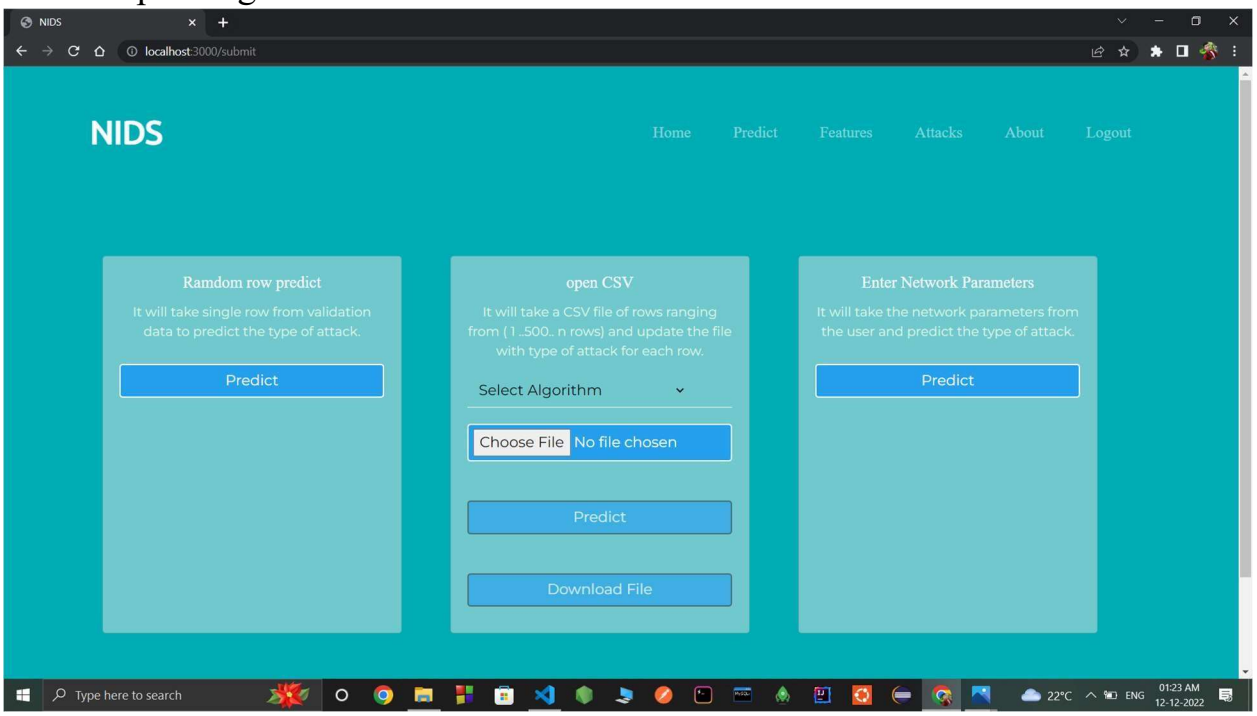
User enter details and validation process



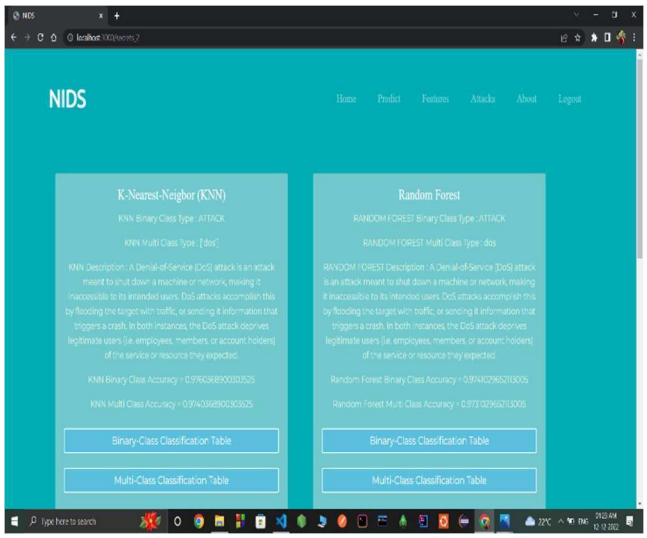
Data gets stored on MongoDB

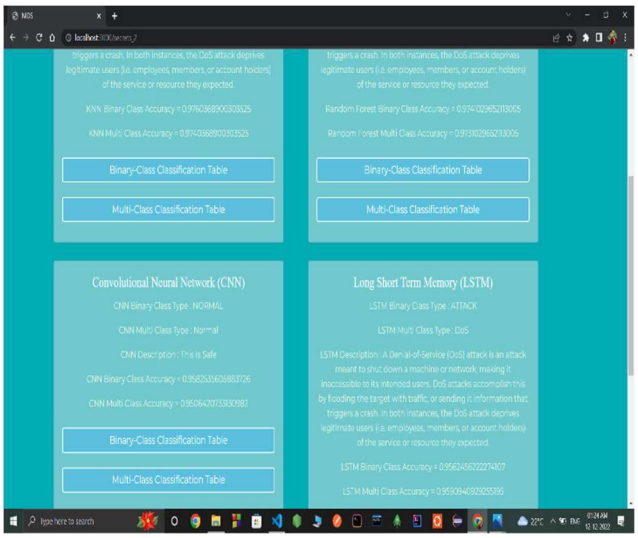


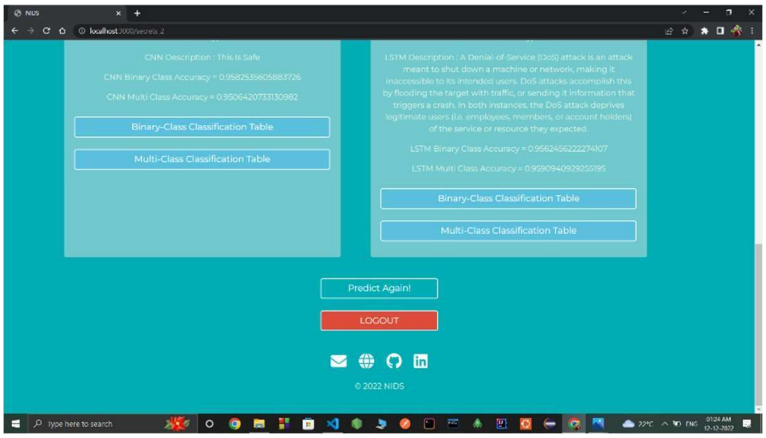
Various options given to user to detect network



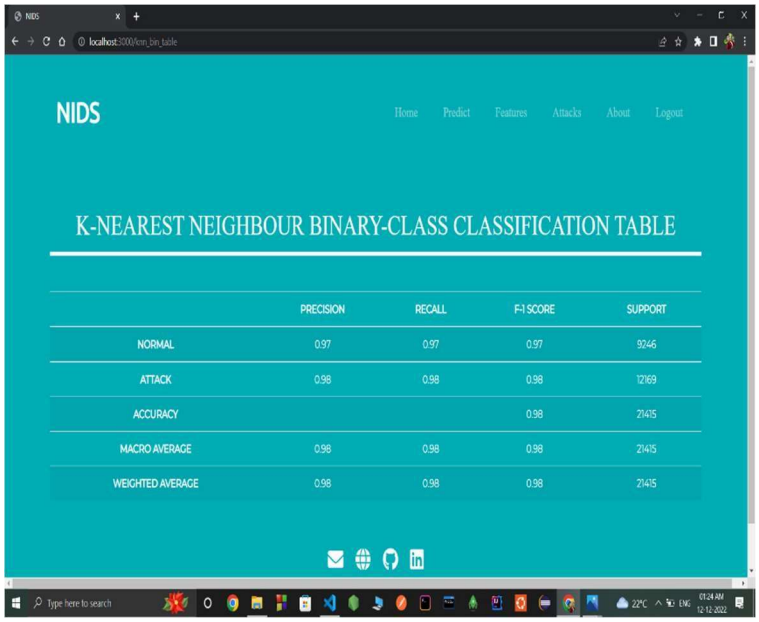
Details about each algorithm and Prediction



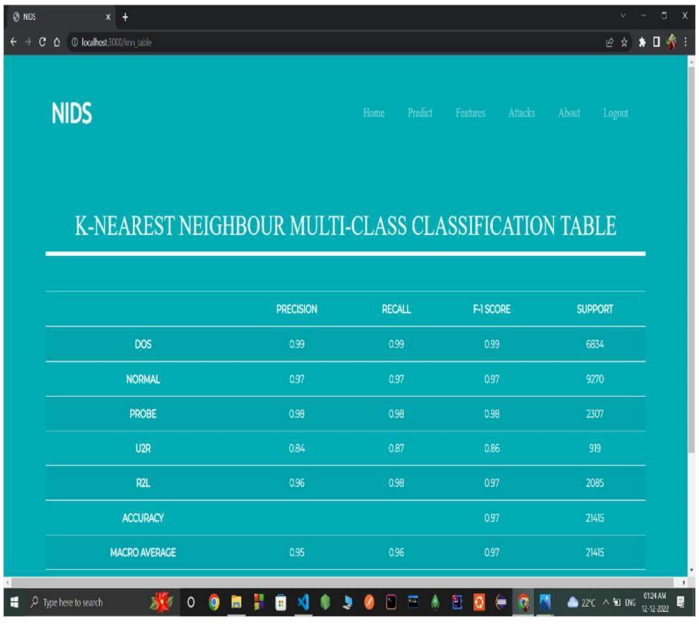




Classification table for the trained and testing data:



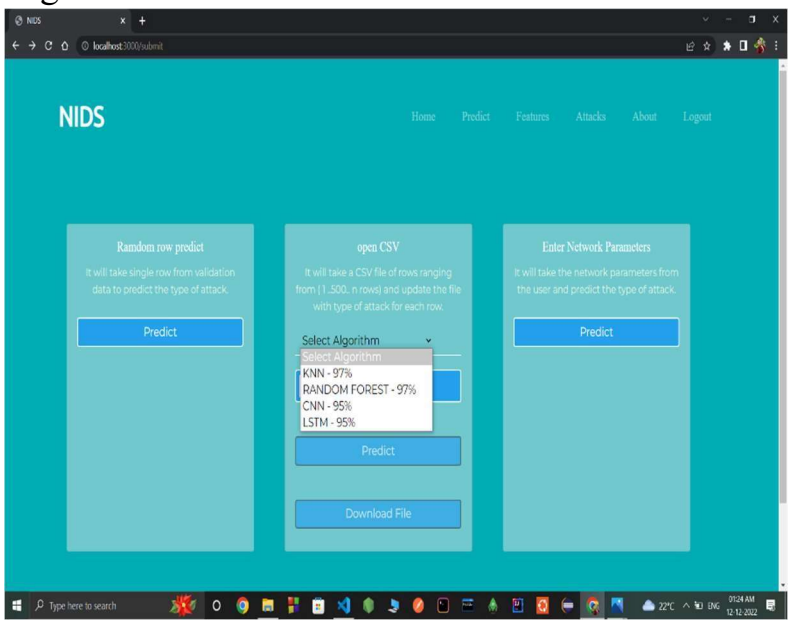
	PRECISION	RECALL	F-1 SCORE	SUPPORT
NORMAL	0.97	0.97	0.97	9246
ATTACK	0.98	0.98	0.98	12169
ACCURACY			0.98	21415
MACRO AVERAGE	0.98	0.98	0.98	21415
WEIGHTED AVERAGE	0.98	0.98	0.98	21415



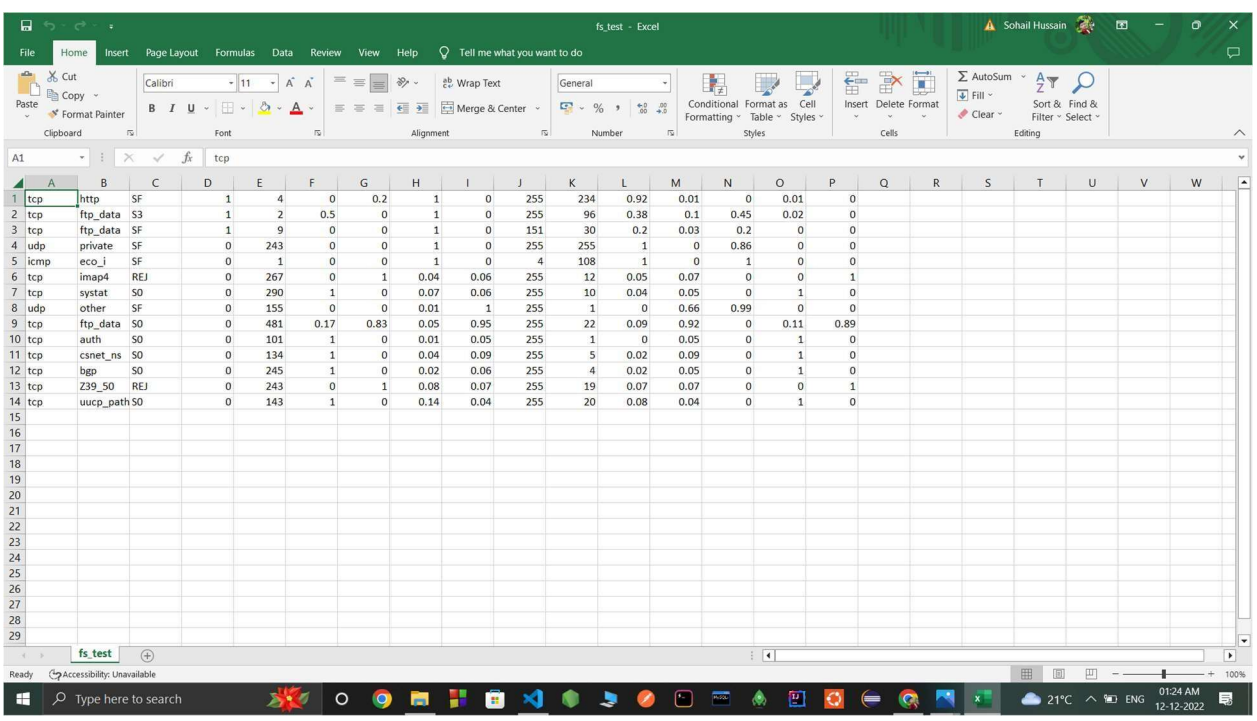
	PRECISION	RECALL	F-1 SCORE	SUPPORT
DOS	0.99	0.99	0.99	6834
NORMAL	0.97	0.97	0.97	9270
PROBE	0.98	0.98	0.98	2307
U2R	0.84	0.87	0.86	919
R2L	0.96	0.98	0.97	2085
ACCURACY			0.97	21415
MACRO AVERAGE	0.95	0.96	0.97	21415

PREDICTION USING CSV:

Algorithm selection:

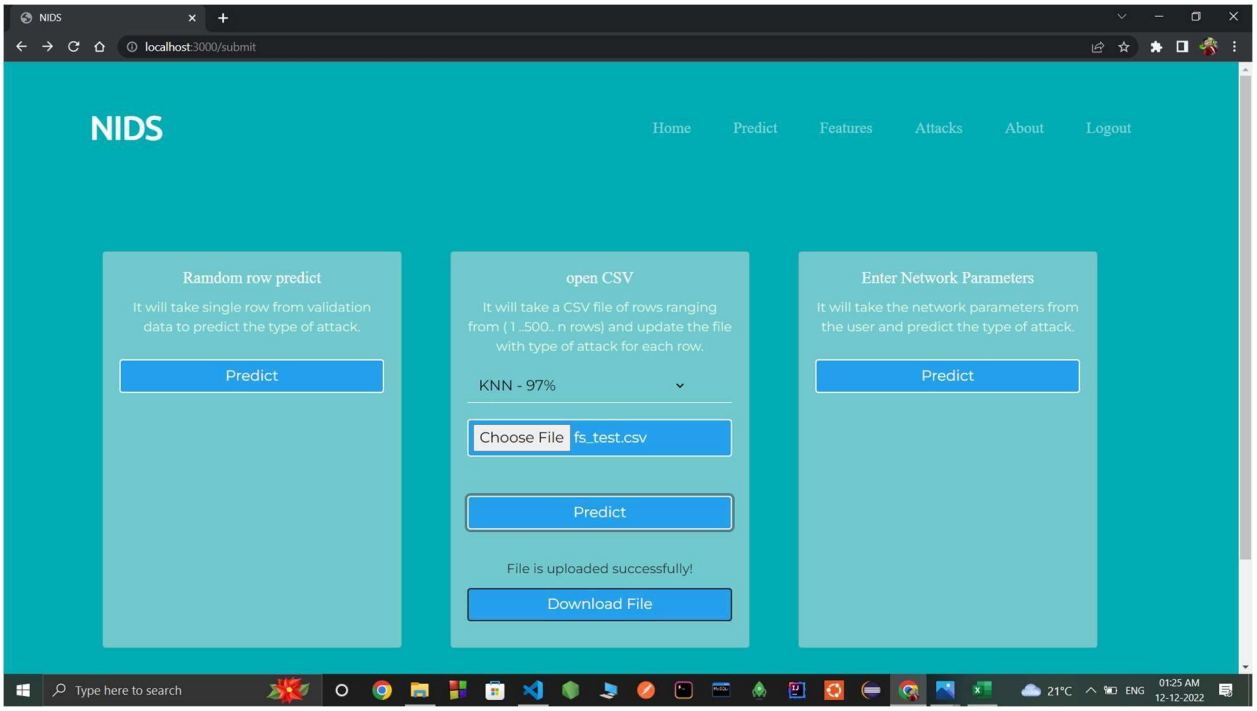


Providing CSV as an input to model:

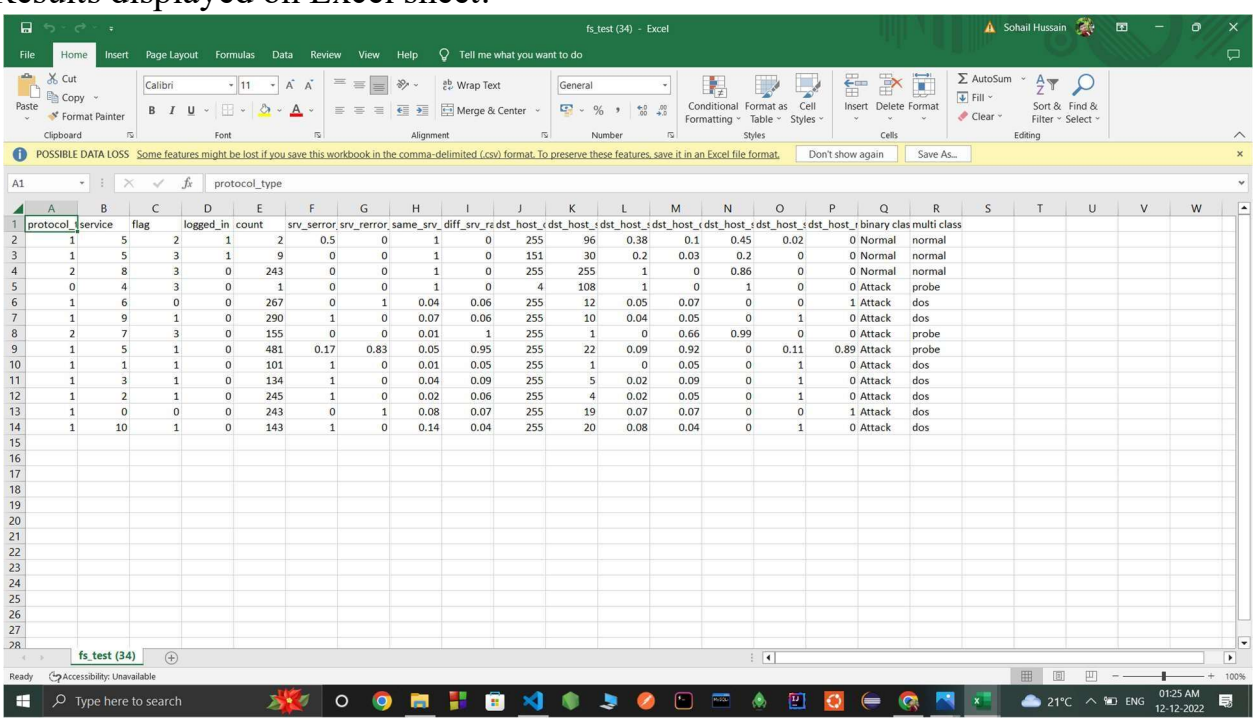


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	tcp	http	SF	1	4	0	0.2	1	0	255	234	0.92	0.01	0	0.01	0
2	tcp	ftp_data	S3	1	2	0.5	0	1	0	255	96	0.38	0.1	0.45	0.02	0
3	tcp	ftp_data	SF	1	9	0	0	1	0	151	30	0.2	0.03	0.2	0	0
4	udp	private	SF	0	243	0	0	1	0	255	255	1	0	0.86	0	0
5	icmp	eco_i	SF	0	1	0	0	1	0	4	108	1	0	1	0	0
6	tcp	imap4	REJ	0	267	0	1	0.04	0.06	255	12	0.05	0.07	0	0	1
7	tcp	systat	S0	0	290	1	0	0.07	0.06	255	10	0.04	0.05	0	1	0
8	udp	other	SF	0	155	0	0	0.01	1	255	1	0	0.66	0.99	0	0
9	tcp	ftp_data	S0	0	481	0.17	0.83	0.05	0.95	255	22	0.09	0.92	0	0.11	0.89
10	tcp	auth	S0	0	101	1	0	0.01	0.05	255	1	0	0.05	0	1	0
11	tcp	csnet_ns	S0	0	134	1	0	0.04	0.09	255	5	0.02	0.09	0	1	0
12	tcp	bgp	S0	0	245	1	0	0.02	0.06	255	4	0.02	0.05	0	1	0
13	tcp	Z39_50	REJ	0	243	0	1	0.08	0.07	255	19	0.07	0.07	0	0	1
14	tcp	uucp_path	S0	0	143	1	0	0.14	0.04	255	20	0.08	0.04	0	1	0

Prediction:

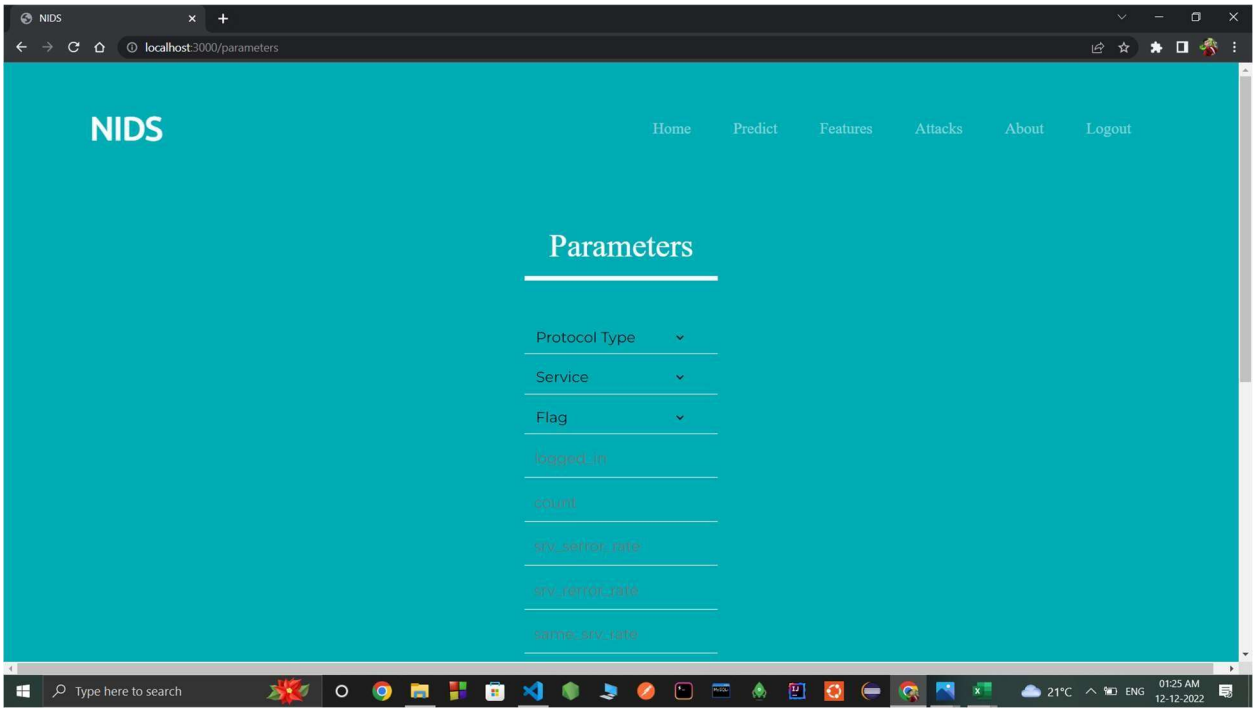


Results displayed on Excel sheet:

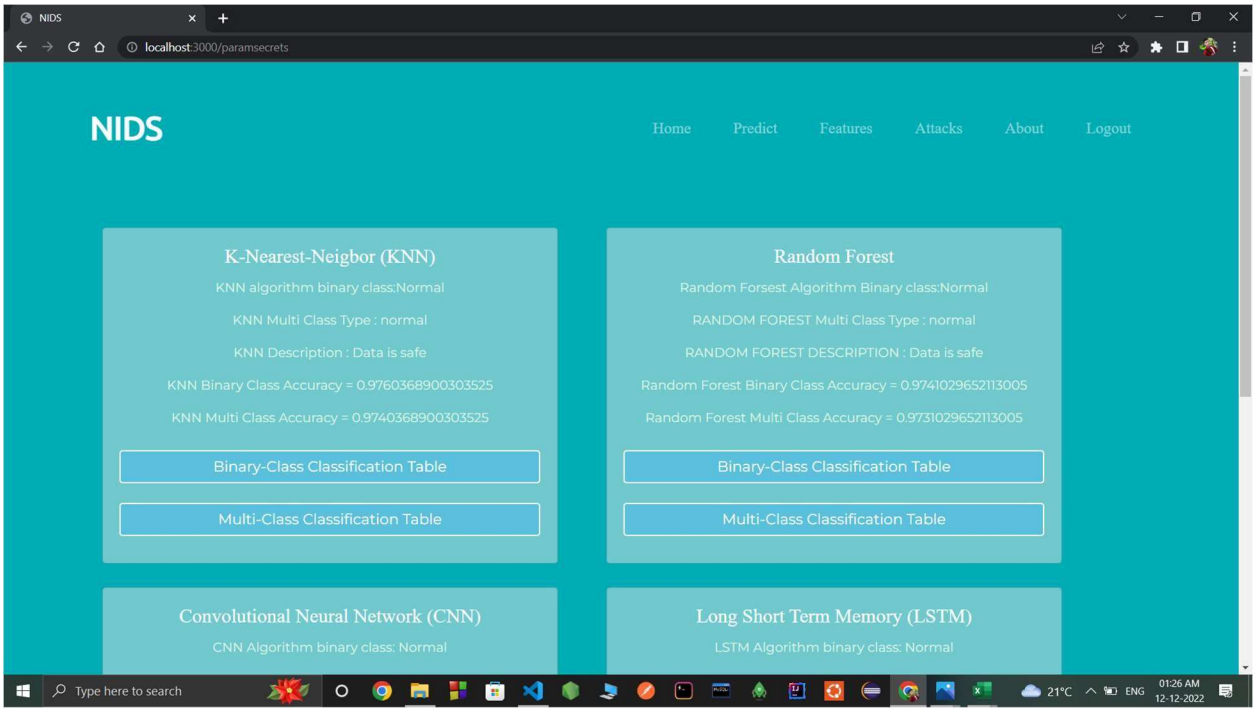


protocol_t	service	flag	logged_in	count	srv_serror	srv_rerror	same_srv_	diff_srv_ra	dst_host_(	dst_host_s	dst_host_s	dst_host_(	dst_host_s	dst_host_s	dst_host_r	binary clas	multi class
1	5	2	1	2	0.5	0	1	0	255	96	0.38	0.1	0.45	0.02	0	Normal	normal
1	5	3	1	9	0	0	1	0	151	30	0.2	0.03	0.2	0	0	Normal	normal
2	8	3	0	243	0	0	1	0	255	255	1	0	0.86	0	0	Normal	normal
0	4	3	0	1	0	0	1	0	4	108	1	0	1	0	0	Attack	probe
1	6	0	0	267	0	1	0.04	0.06	255	12	0.05	0.07	0	0	1	Attack	dos
1	9	1	0	290	1	0	0.07	0.06	255	10	0.04	0.05	0	1	0	Attack	dos
2	7	3	0	155	0	0	0.01	1	255	1	0	0.66	0.99	0	0	Attack	probe
1	5	1	0	481	0.17	0.83	0.05	0.95	255	22	0.09	0.92	0	0.11	0.89	Attack	probe
1	1	1	0	101	1	0	0.01	0.05	255	1	0	0.05	0	1	0	Attack	dos
1	3	1	0	134	1	0	0.04	0.09	255	5	0.02	0.09	0	1	0	Attack	dos
1	2	1	0	245	1	0	0.02	0.06	255	4	0.02	0.05	0	1	0	Attack	dos
1	0	0	0	243	0	1	0.08	0.07	255	19	0.07	0.07	0	0	1	Attack	dos
1	10	1	0	143	1	0	0.14	0.04	255	20	0.08	0.04	0	1	0	Attack	dos

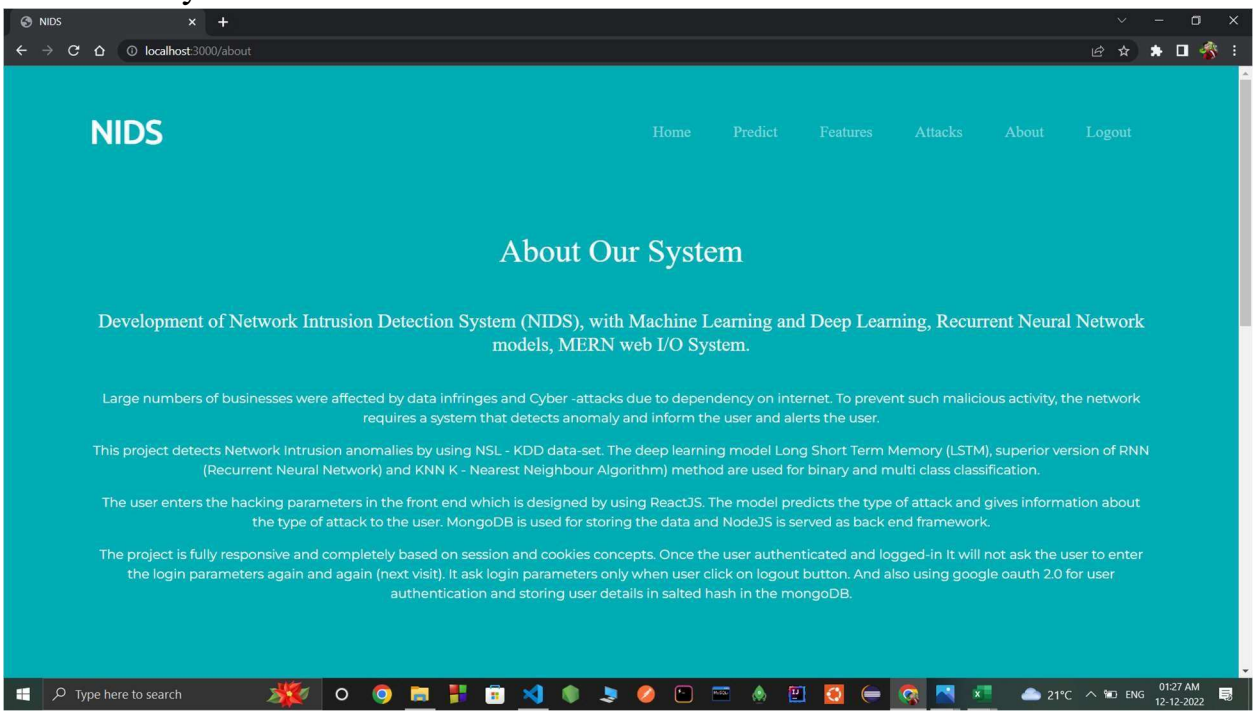
Parameters:



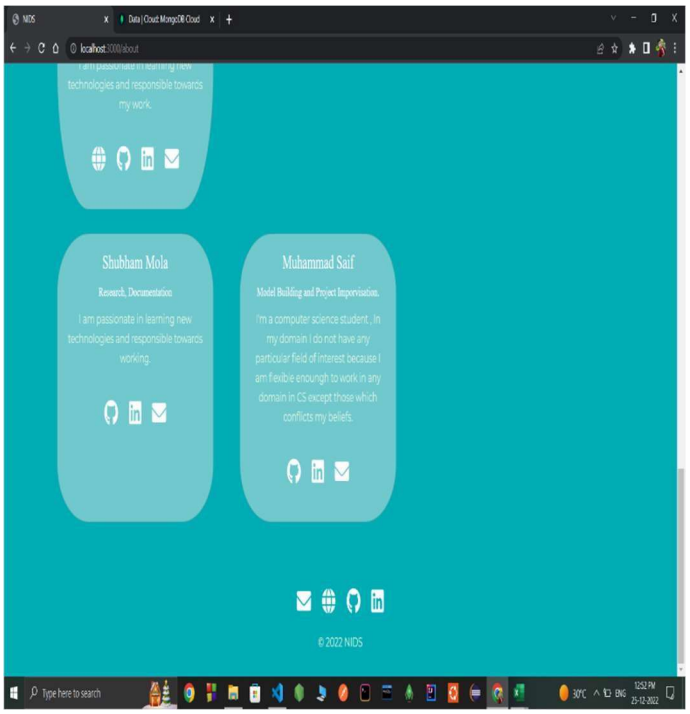
Result



About Our System



Team Details:



Model stats:

NIDS

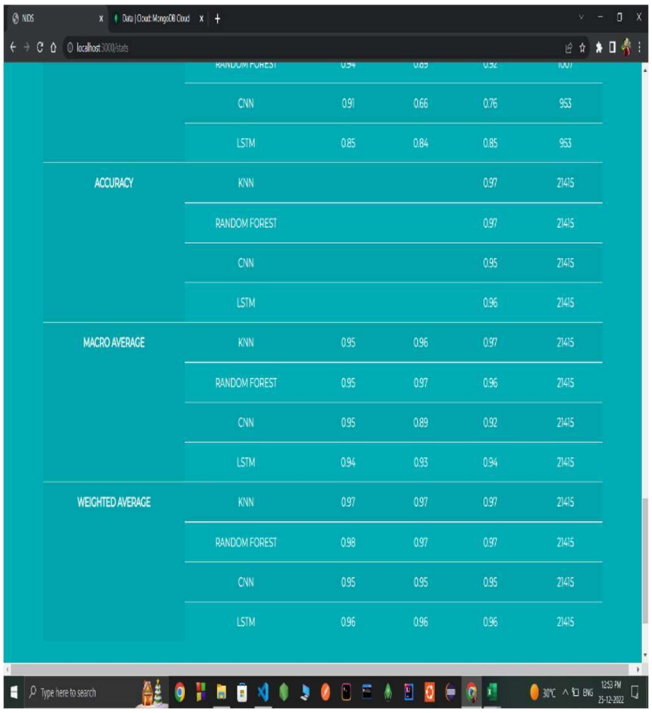
Home Predict Features Attacks About Stats Logout

BINARY-CLASS CLASSIFICATION TABLE

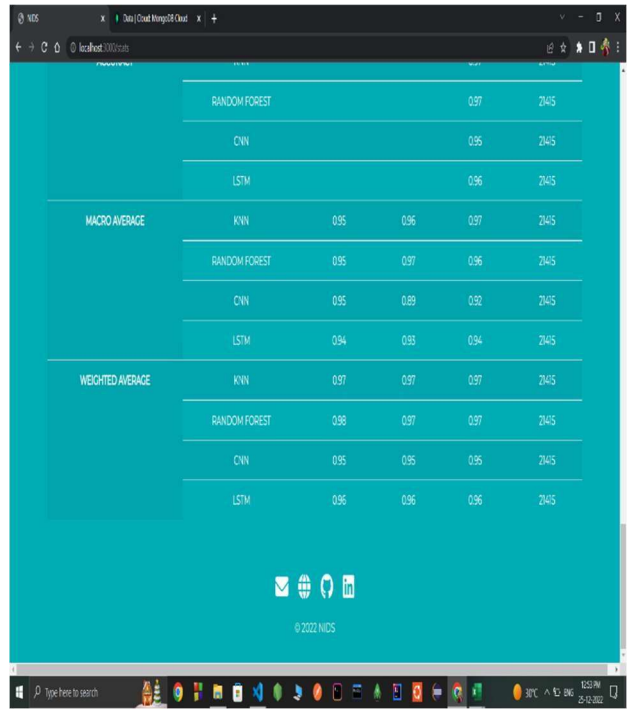
	ALGORITHMS	PRECISION	RECALL	F-1 SCORE	SUPPORT
NORMAL	KNN	0.97	0.97	0.97	9246
	RANDOM FOREST	0.99	0.96	0.97	9847
	CNN	0.94	0.96	0.95	9217
	LSTM	0.94	0.96	0.95	9217
ATTACK	KNN	0.98	0.98	0.98	12169
	RANDOM FOREST	0.97	0.99	0.98	11928
	CNN	0.97	0.96	0.96	12198
	LSTM	0.97	0.96	0.96	12198
ACCURACY	KNN			0.98	21415
	RANDOM FOREST			0.97	21415
	CNN			0.96	21415
	LSTM			0.96	21415
MACRO AVERAGE	KNN	0.98	0.98	0.98	21415
	RANDOM FOREST	0.98	0.97	0.97	21415
	CNN	0.96	0.96	0.96	21415
	LSTM	0.95	0.96	0.96	21415
WEIGHTED AVERAGE	KNN	0.98	0.98	0.98	21415
	RANDOM FOREST	0.97	0.97	0.97	21415
	CNN	0.96	0.96	0.96	21415
	LSTM	0.96	0.96	0.96	21415

MULTI-CLASS CLASSIFICATION TABLE

	RANDOM FOREST	1.00	1.00	1.00	6822
	CNN	0.99	0.97	0.98	6822
	LSTM	1.00	0.97	0.98	6822
NORMAL	KNN	0.97	0.97	0.97	9270
	RANDOM FOREST	0.99	0.95	0.97	9513
	CNN	0.92	0.98	0.95	9217
	LSTM	0.94	0.97	0.96	9217
PROBE	KNN	0.98	0.98	0.98	2307
	RANDOM FOREST	1.00	1.00	1.00	2308
	CNN	0.94	0.97	0.96	2298
	LSTM	0.96	0.98	0.97	2298
U2R	KNN	0.84	0.87	0.86	919
	RANDOM FOREST	0.83	1.00	0.90	1765
	CNN	0.97	0.89	0.93	2125
	LSTM	0.97	0.90	0.94	2125



	CNN	0.91	0.66	0.76	953
	LSTM	0.85	0.84	0.85	953
ACCURACY	KNN			0.97	21415
	RANDOM FOREST			0.97	21415
	CNN			0.95	21415
	LSTM			0.96	21415
MACRO AVERAGE	KNN	0.95	0.96	0.97	21415
	RANDOM FOREST	0.95	0.97	0.96	21415
	CNN	0.95	0.89	0.92	21415
	LSTM	0.94	0.93	0.94	21415
WEIGHTED AVERAGE	KNN	0.97	0.97	0.97	21415
	RANDOM FOREST	0.98	0.97	0.97	21415
	CNN	0.95	0.95	0.95	21415
	LSTM	0.96	0.96	0.96	21415



	RANDOM FOREST			0.97	21415
	CNN			0.95	21415
	LSTM			0.96	21415
MACRO AVERAGE	KNN	0.95	0.96	0.97	21415
	RANDOM FOREST	0.95	0.97	0.96	21415
	CNN	0.95	0.89	0.92	21415
	LSTM	0.94	0.93	0.94	21415
WEIGHTED AVERAGE	KNN	0.97	0.97	0.97	21415
	RANDOM FOREST	0.98	0.97	0.97	21415
	CNN	0.95	0.95	0.95	21415
	LSTM	0.96	0.96	0.96	21415

© 2022 NIDS

Future Scope:

In the future scope there can be a complete application which runs in background while browsing on to the internet and make the application raises an alert while browsing over an unsafe website.

References:

- https://www.techopedia.com/definition/12941/network-based-intrusiondetection-system-nids
- https://www.ijrte.org/wp-content/uploads/papers/v9i1/A1942059120.pdf
- https://www.geeksforgeeks.org/k-nearest-neighbours/
- https://www.geeksforgeeks.org/introduction-convolution-neural-network/ ● https://www.unb.ca/cic/datasets/nsl.html

Conclusion:

The project NIDS is developed considering all key aspects and primarily user data security and privacy. It is efficient and strong enough to handle any type of attack.