IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

Network Intrusion Detection System using Machine Learning

Daroori Raghu Vamshi[1], Dr. Preethi Jeevan[2], Kurva Raja Shekar[3], Kothapally Hemanth[4]
B.TECH Scholars, Department of Computer Science and Engineering[1,3,4]
Associate Professor, Department of Computer Science and Engineering[2]
Sreenidhi Institute of Science & Technology, Hyderabad, India

Abstract: *The incremental increase in the operation of technology has led to an increase in the quantum of data that's being reused over the Internet significantly over the time period. With the huge quantum of data that's being flown over the Internet, comes the script of furnishing security to the data, and this is where an Intrusion Detection System (IDS) comes into the picture and helps in detecting any virtual security pitfalls. Intrusion Detection System (IDS) is a system that monitors and analyzes data to descry any intrusion in the system or network. Interferers that find different ways to access into a network. The IDS which are being proposed is being enforced using technologies such as Machine Learning Algorithms to classify the attacks and detecting them whenever an attack happens and also to find which machine learning algorithm is suitable for detecting the attack. The Intrusion Detection System (IDS) plays an important part in deterring attacks. Still, arising technologies, like cloud computing, Internet of Things etc., induce a large volume of traffics, which may carry the inapplicable attributes that don't have any impact on discovery of assaults. To overcome these issues, features selection approaches (FSA) have been used to remove non- relevant features and find the important ones.*

Keywords: Host, Network, Detection Techniques, Support Vector machine, Machine Learning, Intrusion Detection System

I. INTRODUCTION

Network intrusion detection systems (NIDS) are a crucial component of modern cybersecurity infrastructure. They are designed to monitor network traffic for signs of potential security threats and to alert the administrator when such threats are detected. This can help prevent data breaches and other malicious activity, ensuring the security and integrity of the network. Machine learning is a type of artificial intelligence that involves training algorithms on large datasets to recognize patterns and make predictions. In the context of NIDS, machine learning algorithms can be trained on datasets of normal and malicious network traffic to learn the characteristics that distinguish the two types of traffic. Once trained, the algorithm can be used to classify new, unseen network traffic as normal or malicious.

II. BACKGROUND STUDY (LITERATURE)

Now-a-days internet has become an integral part of our everyday life and businesses, the world of business is linked to the internet. Each day, numerous companies use the Internet to profit from the modern business conception known as e-business. thus, enhancing connectivity of modern businesses. While the Internet has provided companies and individuals the opportunity to connect to a large amount of population and has been profitable it also, comes with a downside that the companies digital infrastructure i.e., databases, network devices, etc., are vulnerable to data breaches, unauthorized access from malicious programs and hackers (or) “intruders”.

These vulnerabilities open up the companies to legal action by their users for data breaches due to lack of proper digital security to protect the customer information.

In addition to data breaches and unauthorized access, not having an IDS can also increase the risk of other forms of malicious activity, such as network attacks and malware infections. Network attacks involve efforts to disrupt or disable a network or system, and they can have serious consequences, including financial losses and damage to an

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

organization's reputation. Malware infections occur when a system is infected with malicious software, such as viruses or trojans, and they can also have serious consequences, including data theft and damage to the system.

III. METHODOLOGY

EXISTING SYSTEM

There are several types of NIDS, including signature-based, anomaly-based, and hybrid systems. Signature-based NIDS rely on a database of known attack patterns or "signatures" to identify potential threats. These systems are effective at detecting well-known attacks, but they may not be able to identify new or previously unseen threats and depend upon the database to be updated regularly. Anomaly-based NIDS, on the other hand, monitor network activity and look for deviations from normal behavior. This allows them to detect previously unseen attacks, but they may have a higher rate of false positives and require more tuning to reduce false alarms. One important aspect of NIDS design is the trade-off between false negatives and false positives. A false negative is when the NIDS fails to detect a real attack, while a false positive is when the NIDS raises an alarm for a benign event. Balancing these two types of errors is a key challenge in NIDS design. It is important to minimize false negatives, as failing to detect a real attack can have serious consequences, but it is also important to minimize false positives, as excessive alarms can lead to "alert fatigue" and hinder the effectiveness of the NIDS.

PROPOSED SYSTEM

In the Proposed Intrusion Detection System (IDS) we use machine learning algorithms to create a model that are trained over the NSL-KDD dataset that is obtained from the Canadian Institute for Cybersecurity. It contains 125,974 samples of normal and abnormal behaviour in the network.

In the proposed system we use the Support Vector Machine (SVM), K-Nearest Neighbours (KNN), Multi-Layer Perceptron (MLP) machine learning algorithms to train the detection model over the NSL-KDD dataset.

We train the dataset over 3 types of machine learning models to improve the true positive rate of our detection system.

IV. ALGORITHM

In our NIDS one of the algorithms, we use is Support Vector Machine (SVM) algorithm with kernel type ‘linear’ for classification. SVM algorithm tries to find a ‘hyperplane’ in an n-dimensional plane that can separate the datapoints of the dataset. The number of dimensions depends upon the number of attributes that are given as input to the algorithm.

The second algorithm that is used is K-Nearest Neighbour (KNN) algorithm which is mostly used for classification problems. KNN does not perform any action on the dataset during the training phase, only storing the dataset. When an input is given to classify then it classifies the input data into data which is similar to it.

The third algorithm used is a deep learning algorithm, a Multi- Layer Perceptron (MLP) using a Sequential model. In MLP there are neurons which are connected to other neurons, these connections are assigned weights to indicate their significance. Neurons activate when activation function threshold is reached by an input. Learning in MLP is done through backpropagation through which each weight is fine- tuned to reach the target attribute.

In our final dataset that given as input to these algorithms there are 93 attributes, but we only select the attributes which have a Pearson Correlation Co-efficient greater than 0.5 with the target attribute ‘intrusion’

Pearson Correlation Coefficient is calculated as:

$$r = \frac{\sum(x_i - \bar{x})\sum(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

The attributes with correlation > 0.5 to the ‘intrusion’ attribute are:

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

count	0.613251
logged_in	0.693770
srv_serror_rate	0.710852
serror_rate	0.712861
dst_host_serror_rate	0.714247
dst_host_same_srv_rate	0.716820
dst_host_srv_serror_rate	0.717387
dst_host_srv_count	0.718579
same_srv_rate	0.798358
intrusion	1.000000

Table 4.1 Attribute correlation

V. ARCHITECTURE

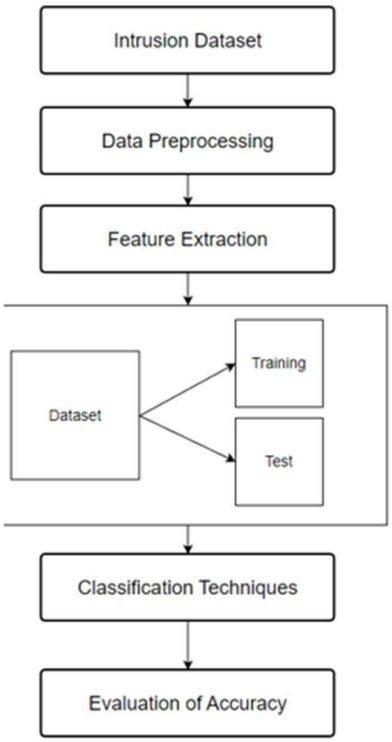


Fig 5.1 System architecture

Steps involved

Step-1: Collecting the dataset.

Step-2: Pre-processing the dataset which involves data normalization using Standard Scaler and encoding the dataset using one-hot-encoder.

Step -3: Extracting the features that the models need to be trained on is done using attributes which have correlation > 0.5 with the target attribute.

Step-4: Split the dataset at ratio of 75/25 of training and test dataset respectively.

Step-5: Training the Machine Learning models using the selected algorithms.

Step-6: Evaluating the accuracy of the trained models with the test dataset using accuracy_score().

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

VI. IMPLEMENTATION

6.1 UML DIAGRAMS

A. Use Case diagram

The Network Intrusion Detection use case diagram represents how users interact with the system. It also represents how each module interacts with the other modules and also the relation between those modules

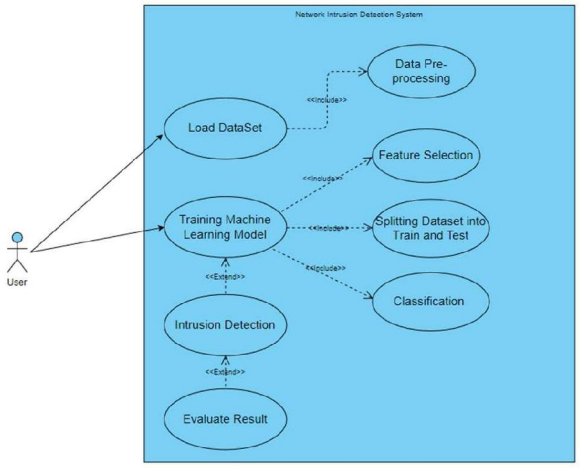


Fig 6.1.1 Use case diagram

B. Sequence diagram

The Network Intrusion Detection sequence diagram shows the sequence in which each process occurs

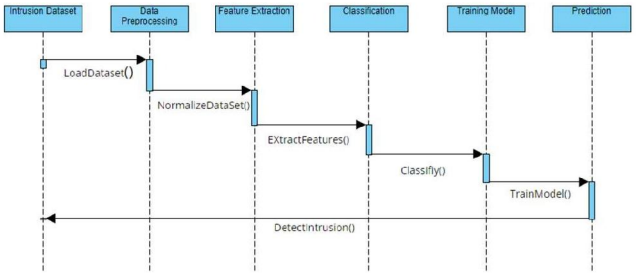


Fig 6.1.2 Sequence diagram

C. Training module:

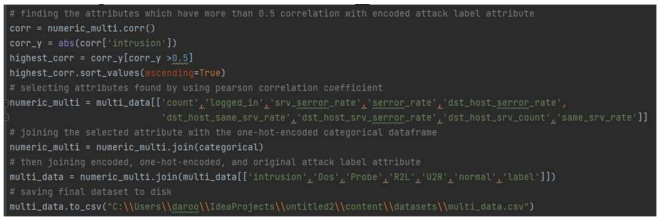
Data Pre-processing

```
# changing attack labels to their respective attack class
def change_label(df):
    df.label.replace(['apache2','back','land','neptune','mailbomb','pod','processtable','smurf','teardrop','udpstorm','worm'],'Dos',inplace=True)
    df.label.replace(['ftp_write','guess_passwd','httptunnel','imap','multihop','named','phf','sendmail',
                      'snmpgetattack','snmpguess','spy','warezclient','warezmaster','xlock','xsnoop'],'R2L',inplace=True)
    df.label.replace(['ipsweep','mscan','nmap','portsweep','saint','satan'],'Probe',inplace=True)
    df.label.replace(['buffer_overflow','loadmodule','perl','ps','rootkit','sqlattack','xterm'],'U2R',inplace=True)
# calling change_label() function


change_label(data)
# selecting numeric attributes columns from data
numeric_col = data.select_dtypes(include='number').columns
# using standard scaler for normalizing
std_scaler = StandardScaler()


def normalization(df,col):
    for i in col:
        arr = df[i]
        arr = np.array(arr)
        df[i] = std_scaler.fit_transform(arr.reshape(len(arr),1))
    return df
# calling the normalization() function


data = normalization(data.copy(),numeric_col)
```

Fig 6.2.1 Data pre-processing

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

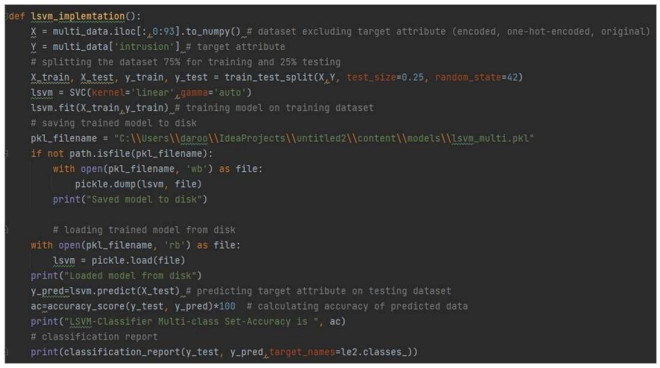
Finding attributes with correlation >0.5 with target attribute and saving the final dataset *multi_data* to csv file.



```
# finding the attributes which have more than 0.5 correlation with encoded attack label attribute
corr = numeric_multi.corr()
corr_y = abs(corr['intrusion'])
highest_corr = corr_y[corr_y >0.5]
highest_corr.sort_values(ascending=True)
# selecting attributes found by using pearson correlation coefficient
numeric_multi = multi_data[['count','logged_in','srv_serror_rate','serror_rate','dst_host_serror_rate',
                            'dst_host_same_srv_rate','dst_host_srv_serror_rate','dst_host_srv_count','same_srv_rate']]
# joining the selected attribute with the one-hot-encoded categorical dataframe
numeric_multi = numeric_multi.join(categorical)
# then joining encoded, one-hot-encoded, and original attack label attribute
multi_data = numeric_multi.join(multi_data[['intrusion','Dos','Probe','R2L','U2R','normal','label']])
# saving final dataset to disk
multi_data.to_csv("C:\\Users\\darog\\IdeaProjects\\untitled2\\content\\datasets\\multi_data.csv")
```

Fig 6.2.2 Feature extraction

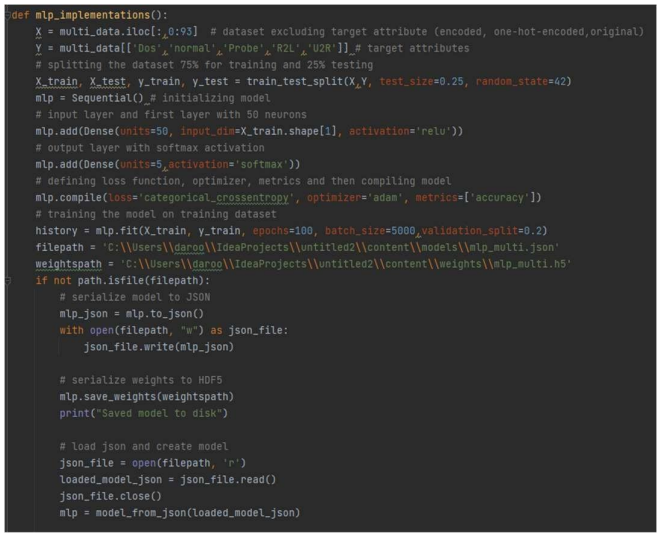
Creating a machine learning model using SVM with kernel mode as linear and saving the created model.



```
def lsvm_implemtation():
    X = multi_data.iloc[:,0:93].to_numpy() # dataset excluding target attribute (encoded, one-hot-encoded, original)
    Y = multi_data['intrusion'] # target attribute
    # splitting the dataset 75% for training and 25% testing
    X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.25, random_state=42)
    lsvm = SVC(kernel='linear',gamma='auto')
    lsvm.fit(X_train,y_train) # training model on training dataset
    # saving trained model to disk
    pkl_filename = "C:\\Users\\darog\\IdeaProjects\\untitled2\\content\\models\\lsvm_multi.pkl"
    if not path.isfile(pkl_filename):
        with open(pkl_filename, 'wb') as file:
            pickle.dump(lsvm, file)
        print("Saved model to disk")

        # loading trained model from disk
    with open(pkl_filename, 'rb') as file:
        lsvm = pickle.load(file)
    print("Loaded model from disk")
    y_pred=lsvm.predict(X_test) # predicting target attribute on testing dataset
    ac=accuracy_score(y_test, y_pred)*100  # calculating accuracy of predicted data
    print("LSVM-Classifier Multi-Class Set-Accuracy is ", ac)
    # classification report
    print(classification_report(y_test, y_pred,target_names=le2.classes_))
```

Fig 6.2.3 Creating and training ML model with SVM

Creating a MLP using a sequential model with adam optimizer and saving the created model.

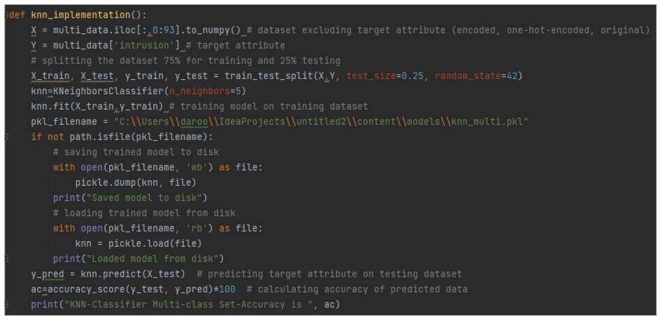


```
def mlp_implementations():
    X = multi_data.iloc[:,0:93]  # dataset excluding target attribute (encoded, one-hot-encoded,original)
    Y = multi_data[['Dos','normal','Probe','R2L','U2R']] # target attributes
    # splitting the dataset 75% for training and 25% testing
    X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.25, random_state=42)
    mlp = Sequential() # initializing model
    # input layer and first layer with 50 neurons
    mlp.add(Dense(units=50, input_dim=X_train.shape[1], activation='relu'))
    # output layer with softmax activation
    mlp.add(Dense(units=5,activation='softmax'))
    # defining loss function, optimizer, metrics and then compiling model
    mlp.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    # training the model on training dataset
    history = mlp.fit(X_train, y_train, epochs=100, batch_size=5000,validation_split=0.2)
    filepath = 'C:\\Users\\darog\\IdeaProjects\\untitled2\\content\\models\\mlp_multi.json'
    weightspath = 'C:\\Users\\darog\\IdeaProjects\\untitled2\\content\\weights\\mlp_multi.h5'
    if not path.isfile(filepath):
        # serialize model to JSON
        mlp_json = mlp.to_json()
        with open(filepath, "w") as json_file:
            json_file.write(mlp_json)

        # serialize weights to HDF5
        mlp.save_weights(weightspath)
        print("Saved model to disk")

        # load json and create model
        json_file = open(filepath, 'r')
        loaded_model_json = json_file.read()
        json_file.close()
        mlp = model_from_json(loaded_model_json)
```

Fig 6.2.4 Creating and training Sequential model

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

Creating a machine learning model using K-NN with *number of neighbors = 5* and saving the created model.



```
def knn_implementation():
    X = multi_data.iloc[:,0:93].to_numpy() # dataset excluding target attribute (encoded, one-hot-encoded, original)
    Y = multi_data['intrusion'] # target attribute
    # splitting the dataset 75% for training and 25% testing
    X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size=0.25, random_state=42)
    knn=KNeighborsClassifier(n_neighbors=5)
    knn.fit(X_train,y_train) # training model on training dataset
    pkl_filename = "C:\\Users\\daroo\\IdeaProjects\\untitled2\\content\\models\\knn_multi.pkl"
    if not path.isfile(pkl_filename):
        # saving trained model to disk
        with open(pkl_filename, 'wb') as file:
            pickle.dump(knn, file)
        print("Saved model to disk")
        # loading trained model from disk
        with open(pkl_filename, 'rb') as file:
            knn = pickle.load(file)
        print("Loaded model from disk")
    y_pred = knn.predict(X_test)  # predicting target attribute on testing dataset
    ac=accuracy_score(y_test, y_pred)*100  # calculating accuracy of predicted data
    print('KNN-Classifier Multi-class Set-Accuracy is ', ac)
```

Fig 6.2.5 Creating and training ML model with K-NN

User Interface module

Using *streamlit* create an interface to upload network status information as a csv file.

```
def main():
    predict_res = ''
    res = ''
    fin_stat = ''
    filename = st.file_uploader(label="Upload the CSV: ", type='csv')
    data = pd.read_csv(filename)
    data.drop(['Unnamed: 0'], axis=1, inplace=True)
    x_fin = data.iloc[:, 0:93].to_numpy()
    option = st.selectbox('which model do you want to use?', ('MLP', 'SVM','KNN'))
    if option == 'MLP':
        predict_res = mlp_prediction(x_fin)
    elif option == 'SVM':
        predict_res = lsvm_prediction(x_fin)
    else:
        predict_res = knn_prediction(x_fin)
    res = attack_dict[predict_res]
    if res == 'Normal':
        fin_stat = 'Status of the system is normal'
    else:
        fin_stat = f'Intrusion Detected of type: {res}'
    st.write(fin_stat)
```

Fig 6.2.6 User Interface using streamlit

VII. RESULTS AND OUTPUTS

Classification Report for SVM model

	Precision	Recall	F1-score	support
Accuracy			**0.96**	**31493**
macro avg	**0.68**	**0.67**	**0.67**	**31493**
Weighted avg	**0.95**	**0.95**	**0.95**	**31493**

Table 7.1 SVM Classification Report

Classification Report for KNN model

	Precision	Recall	F1-score	support
Accuracy			**0.98**	**31493**
macro avg	**0.85**	**0.79**	**0.81**	**31493**
Weighted avg	**0.97**	**0.97**	**0.97**	**31493**

Table 7.2 K-NN Classification Report

Classification Report for MLP (Sequential)

	Precision	Recall	F1-score
Accuracy	**0.98**	**0.96**	**0.97**

Table 7.3 MLP(Sequential) Classification Report

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

User Interface

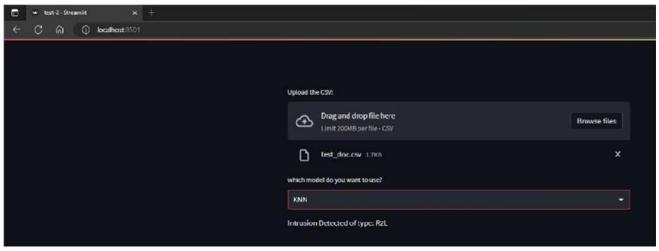


Fig 7.1 Working demo of user interface

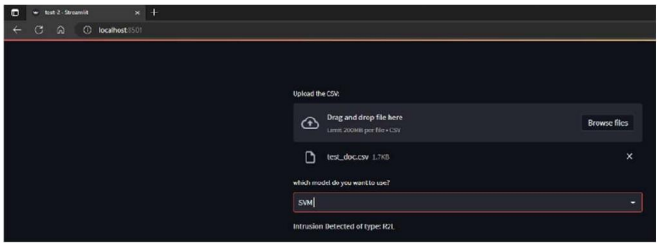


Fig 7.2 Working demo of user interface

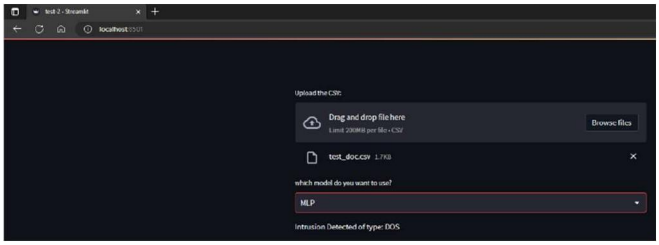


Fig 7.3 Working demo of user interface

VIII. CONCLUSION AND FUTURE WORK

The proposed ML-based Network Intrusion Detection System offers the ability to detect intrusions into the network. The proposed system offers a higher accuracy in detection rate and the ability to detect previously unknown type of intrusions at a higher accuracy than traditional detection systems. Although this system may not completely deter intrusions but, it allows for early detection which can allow organizations to recover quickly from them.

In the future combining the Intrusion Detection System with a Network Intrusion Prevention System which allows to actively prevent an on-going intrusion can bolster network security even more.

REFERENCES

[1]. Thaseen, I. S., Poorva, B., & Ushasree, P. S. (2020, February). Network intrusion detection using machine learning techniques. In 2020 International conference on emerging trends in information technology and engineering (IC-ETITE) (pp. 1-7). IEEE.

[2]. Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches https://doi.org/10.3390/app10051775

[3]. 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).

[4]. Taher, K. A., Jisan, B. M. Y., & Rahman, M. M. (2019, January). Network intrusion detection using supervised machine learning technique with feature selection. In 2019 International conference on robotics, electrical and signal processing techniques (ICREST) (pp. 643-646). IEEE.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 2, February 2024

[5]. Kumar, K., & Batth, J. S. (2016). Network intrusion detection with feature selection techniques using machine-learning algorithms. International Journal of Computer Applications, 150(12).