MACHINE LEARNING BASED INTERNET FIREWALL

Information Security Management - CSE 3502

J-Component Project – Final Project Report

Winter Semester 2022

Team Members:

Moukhik Misra

19BCE2190

B. Tech. Computer Science and Engineering

Mohit Suhasaria

19BCE2167

B.Tech. Computer Science and Engineering

School of Computer Science and Engineering

Vellore Institute of Technology

Vellore

April 2022

I. ABSTRACT

1. Motivation, aim and objective of the project.

Technology has seen a tremendous growth in the recent years, in all sectors from software to hardware. However, this growth has led to a new domain for malicious activities to take place. It means that we must protect our data and resources to the best of our capabilities. The most important thing in information security is to keep the private data secure against unauthorized access and unwanted breaches. There are various tools used for different purposes to build a secure system with various levels of security and authorization. One method to protect private networks is to use a firewall. Firewalls are based on a specified set of security rules determined by the network security admin. These rules are used to monitor network traffic in the form of incoming and outgoing data. The firewall acts as a membrane between the external and internal aspects of data communication. The internal network and its resources must be protected from malicious attacks. The firewall filters network traffic to protect the system from malicious software attacks and viruses while allowing the flow of normal or valid traffic. There are two types of firewalls: hardware and software. Hardware firewalls are physical components in the network that filter network traffic and act as a barrier whereas software firewalls are the firewalls that run on the system and search through data to find malicious code.

The main aim of our project is to study and analyze the contents of various data packets processed by an internet firewall. This data is available to us in the form of a dataset containing over 65,000 entries where the information like source port, destination port, NAT source port, NAT destination port, bytes, action, packets sent, packets received, time elapsed etc. is available. Based on these features of a data packet as a part of the network traffic analysis done by the internet firewall, certain actions are performed to regulate this traffic. These actions include; allow, drop, reset and deny. This gives huge importance to the statistical analysis of these features to recognise trends in malicious traffic as compared to normal traffic.

Our main objective is to perform extensive analysis on the internet firewall dataset to generate as much information about the contents of data packets in a network traffic through the use of statistical visualisations. We also want to emphasise on the usefulness of a machine learning based internet firewall that can accurately regulate the flow of network traffic. In this project, we will use the Random Forest Classifier for classifying the action performed by the firewall based on 11 features such as source port, destination port, bytes etc. The random forest classifier consists of multiple decision trees which generate class predictions individually and the majority prediction becomes the output of the classifier. The model will be evaluated using accuracy score, mean squared error and r squared. This is accompanied by the threat prediction and classification in which attacks such as sql injection, xss, command injection and path traversal attacks are classified using the ECML/PKDD , Http parameters and the XSS datasets. Data exploration is performed, and the distribution of the targets are visualized. Features are calculated using TFIDF on characters and SVM is used for classification. The SVM model uses both Linear and RBF kernels. Grid Search hyperparameter optimisation and the model is then evaluated using popular classification metrics.

2. About Methodology.

The first stage of our project would be to recognise the necessary tools and technologies to be used such as Google Colab, Jupyter, Python and associated libraries like numpy, pandas, seaborn and matplotlib.

The dataset would be loaded onto main memory for exploratory analysis. The exploration should reveal the basic information about the dataset like its size, shape, features, data types and missing or null values.

The dataset used is the Internet Firewall Dataset from UCI repository where 65532 instances of internet traffic logs through Firat University network are available. The logs have 12 features in total out of which one feature corresponds to the action performed by the firewall. We will consider this feature as the class for our analysis where the action can be classified as; allow, deny, drop, and reset-both. The remaining 11 features will be fed to the model for classification. The features of the dataset are Source Port, Destination Port, NAT Source Port, NAT Destination Port, Bytes which are divided into Bytes Sent and Bytes Received, Packets, Elapsed Time in seconds.

All the features available for each entry might not have the same relevance in the outcome when the data packet is filtered by the firewall. Some features might be a lot more relevant in deciding the action performed on the packet than other features. We must also take into consideration the balance of the dataset. This can be done by plotting a distribution of outcomes for the whole dataset to observe any skew towards a specific class or multiple classes. This means that if a large number of entries available to us are classified as a particular class then it becomes easier to predict a given packet as belonging to that class, and in turn the classes with lower entries in the dataset will have a lower probability of being predicted. So spotting a skew in the dataset before fitting any model on it or deriving any kind of information from it is absolutely necessary.

We then move on to feature selection because similar to the reasons mentioned above, all features don't have the same relevance when deciding the classification of a data packet. To help us in feature selection, a logarithmic distribution of features can be made from the numeric features in the dataset. Similarly, the categorical features are visualised through frequency distributions. A target versus feature cross table will also be made for the categorical features to make the frequency distributions useful as numeric values.

The pre-processing, feature exploration and feature selection stages are performed to prepare the data to be split into training and testing sets for the purpose of fitting a machine learning model to it. We have chosen the random forest classification model for this project. It consists of multiple decision trees with various hyperparameters. These decision trees come up with individual classification outcomes and the majority outcome becomes the prediction generated by the Random Forest classifier.

The utility of a model only exists if it is accurate so we shall evaluate the performance of our model using various parameters such as accuracy, mean squared error, root mean squared error and r squared.

This is followed by the threat prediction and classification in which attacks such as sql injection, xss, command injection and path traversal attacks are classified using the ECML/PKDD , Http parameters and the XSS datasets. Data exploration is performed, and the

distribution of the targets are visualized. Features are calculated using TFIDF on characters and SVM is used for classification. The SVM model uses both Linear and RBF kernels. A pipeline using these two methods is used as a model. Grid Search hyperparameter optimisation and the model is then evaluated using popular classification metrics such as accuracy, precison, recall and f1 score as well as an accompanying

3. Expected Outcome.

- We expect to get a detailed insight on the contents of various data packets flowing through a network.
- Understanding the working of an internet firewall and the features that affect the actions performed on data packets.
- Detailed information on the selected dataset about its features and other characteristics like distribution.
- Closer look at inter-class relationships between attributes of data packets.
- Characteristics of target versus feature distribution of categorical features in the selected dataset.
- Detailed analysis of the actions performed by an Internet Firewall based on the different characteristics of network traffic.
- Machine learning based data traffic classifier that can accurately predict the action to be performed on data packets captured through internet traffic using Random Forest.
- Performance Evaluation of implemented model using metrics such as accuracy, logloss, mse, rsme and r^2.
- Prediction of common attacks such as sql injection, xss, path traversal or command injection attacks.
- Extracting and classification of common attack types from string data using TFIDF technique and classification using SVM
- Using Grid Search Cross Validation for optimisation of hyperparameters.
- Evaluation and comparison of created models with existing methods with advantages and disadvantages.

4. Efficiency Obtained

For evaluating the developed packet classification and threat prediction modules, multiple different metrics have been provided. For packet classification, it is observed that the Random Forest model that has been implemented has a mean accuracy of 99.81% and mean logloss of 0.04444 for 5 Cross Validations. Other metrics have been evaluated such as Mean Squared Error (mse) with a mean value of 0.0068 and r^2 has a mean score of 0.989098 and rmse has mean score 0.08279.

For threat detection, metrics such as accuracy, precision, recall and f1-score have been considered for the implemented ML pipeline consisting of TF-IDF and SVM. We have obtained an accuracy of 99.579% and the precision, recall and f1-score of the different targets has also been provided in the classification report.

KEYWORDS: Packet Classification, Threat Detection, Random Forest, Logloss, Cross Validation, Pipeline, TF-IDF, SVM, GridSearchCV

II. INTRODUCTION

1. Overall Idea about the Project

The primary objective of our project is to create an Internet Firewall using Machine Learning. An internet firewall is a network security application that is used to monitor traffic on a particular network. This includes both incoming and outgoing network traffic. Based on certain characteristics of the traffic, which is in the form of packets of data, both incoming and outgoing, the firewall decides to mainly block or accept these packets of data. It can also decide to drop a packet altogether or reset it based on requirements and packet characteristics. Firewalls have been the first line of defence for networks for a long time. However only recently have the concepts of Machine Learning begun to be applied in the design and functioning of firewall systems.

In our project, we will be applying a Machine Learning classifier called the Random Forest Classifier to classify whether a particular packet should either be allowed, denied, dropped or reset. The dataset used by us will the Internet Firewall Data dataset from the UCI Machine Learning Repository. This dataset which contains packet data from Firat University, Turkey contains 12 features in which there is one Action feature. This action feature signifies the 4 classes or outcomes for a particular packet. These classes are allow, deny drop and reset-both actions. The dataset contains 65532 records or instances which is a fairly large sample size for creating an ML model.

The basic idea behind the project is to first import the necessary libraries in python, followed by investigation of the data present in the dataset and obtaining information about the dataset. The dataset contains featres such as Source Port, Destination Port, NAT Source Port, NAT Destination Port, Bytes which are divided into Bytes Sent and Bytes Received, Packets, Elapsed Time in seconds. The packets are further classified into packet sent and packets received. All these features have an effect in deciding the outcome of the network traffic. Before implementing the model, it is important to have a look at how balanced the dataset it. This can be achieved by looking at the frequency of each of the 4 outcomes possible in the dataset. From this we can generally get a good idea of whether a class will be easy or difficult to predict. This is followed by Feature Exploration step in which a logarithmic distribution of the numeric features in the dataset can be obtained. It helps in feature selection. Next, we take a look at the categorical features present in the dataset such as source port, destination port, NaT Source and Destination ports. We will be visualizing the frequency distribution of these features. For categorical features its also important to visualize the target vs feature crosstables for the top levels so as to get a numeric representation of frequency distribution.

After visualizing and understanding the data, it is time to implement the Random Forest Classifier. The dataset will be split into training and test sets and the appropriate hyperparameters will be set to ensure best possible classification. The model will be fit with the data and predict the outcome for the packets in the dataset. It is also important to understand the effectiveness of our model and we can obtain the following by looking at performance metrics of the implemented model. Evaluation of the performance of the model is an essential step for the functioning of the firewall in real world application.

This is followed by the threat prediction and classification in which attacks such as sql injection, xss, command injection and path traversal attacks are classified using the ECML/PKDD , Http parameters and the XSS datasets. Data exploration is performed, and the distribution of the targets are visualized. Features are calculated using TFIDF on characters and SVM is used for classification both of which are set in an ML pipeline. The SVM model uses both Linear and RBF kernels. Grid Search hyperparameter optimisation is used for 2 cross validations and the model is then evaluated using popular classification metrics.

2. Background of the Project.

A firewall is an essential tool in the defence of our computer systems and networks from malicious activity from connected networks and the internet. Firewall is often considered the first line of defence against malicious internet activity. In essence, a firewall is an application whose primary functionality is to monitor incoming and outgoing traffic in computer networks and to discern whether said traffic falls under acceptable conditions and can be allowed to pass or go or whether the traffic should be denied or dropped or reset owing to its characteristics. Firewalls are not a new software having been in existence many years ago. There are many types of firewalls as well such as proxy firewall, stateful inspection firewall, UTM firewalls, NGFW firewalls, threat focused NGFW firewalls and other such types of firewalls. These firewalls serve different purposes. However, the world of cyber threats and cybercrimes is every expanding and newer techniques to infect our systems with malware are always popping up. Recently a lot of different types of cyber-attacks occurred causing massive damages to both large scale organizations as well as to individuals in the form of infecting their systems with malware ranging from small scale effect to straight up taking control over people's systems.

To combat this a firewall must be dynamic and able to come up with solutions to such malicious actors. That is the firewall must be capable of learning the different types of malicious data and be able to use this knowledge to prevent further such malicious activity. Machine learning techniques have only recently been used in Internet Firewalls to prevent malicious activity. Even still the application of machine learning techniques in firewalls is in a nascent stage with various rooms for improvements and new technologies. The main problem with using ML in firewalls is the acute need of control of network traffic and management of different types of networks. This poses a huge challenge as each network is different and it is overseen by different standards. This make creating a unified ML model for solving firewall problems nigh on impossible. The networks are also continuously changing and evolving and the patterns which an ML model needs for predictions are not always followed. Also, different types of devices maybe connected to the network, and this keeps on changing and these devices have different characteristics which also poses a challenge for unified ML models. However, recently there is enough computational power in today's machines to be able to use ML techniques to learn patterns of malicious activity in the background of one's systems to be able to use these techniques in real world detection scenarios.

With an eye on this, we decided to develop our own solution to an ML based Internet Firewall. Our objective is to build a robust and accurate model that can correctly predict whether certain traffic should be allowed, denied, reset or dropped based on the features that accompany it. To achieve our objectives, we will be using Random Forest Classifier for our ML model. The aim is to create a firewall which is capable of accurately determining the correct outcome for

network traffic based in the training data that is provided to it. Additionally, with the growing number of attack types and methods available to hackers today, it is essential to detect and properly classify attack types and put blocking and defence mechanisms in place. To detect and classify different types of attacks such as sqli, xss, cmdi or path traversal and others we have created a machine learning pipeline consisting of TF-IDF vectorizer followed by and SVM model (both linear and rbf kernels) used alongside Grid Search Cross Validation to find the best parameters (hyperparameter optimisation). The created model is evaluated with metrics such as accuracy, f1-score, precison and recall.

3. Statistics Related to Methods Used

i) Random Forest Classifier

Random forest is an ensemble learning machine learning technique that consists of a large number of decision trees operating together in a specified way. Basically in random forest, each decision tree gives out a prediction for the classification problem which in this case is the firewall, whether to accept, deny, drop or reset traffic and the prediction class with the most votes becomes the final prediction. It uses majority voting technique to get final prediction. The main reason that random forest gives better performance is that a group of unrelated or uncorrelated trees (or ML models) operating in the form of a group or committee statistically outperforms any individual model (tree). Low correlation between the models is a very important factor as when multiple of these uncorrelated parts come together, they produce a combined prediction better than individual models as they protect each other from incorrect decisions or errors. The totality of the tress move together in the right direction even though some tress come to the wrong decision.

Random Forest classifier requires there to be features with noticeable predictive power. It wont help if we put in features totally unrelated to the outcome. It also requires, as mentioned above, the trees in the forest and their predictions to be uncorrelated. This can be attained by proper feature selection and hyperparameter tuning.

Decision Trees are extremely sensitive to changes in training data. Any small change in training data gives different tree structures. This is an advantage of random forest as it allows each tree to randomly sample from the dataset with certain changes resulting in a different tree structure . this is a process known as bagging. It allows random forest to be receptive of changes in data whilst being accurate in predictions. The next important aspect of random forest is feature randomness where in each tree in random forest classifier is capable of only picking from a random subset of features. This allows for the least correlation possible between the formed trees allowing us to obtain the fairest result from the majority voting. Random forest uses different decision trees that are trained on different subsets of data and use different features for predictive purposes. This is what makes random forest a very effective ML model, hence why we are using it in our project.

Feature Importance Formula

$$fi_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} ni_j}{\sum_{k \in all\ nodes} ni_k}$$

- fi sub(i)= the importance of feature i
- ni sub(j)= the importance of node j

Normalization between 0 and 1:

$$normfi_i = \frac{fi_i}{\sum_{j \in all\ features} fi_j}$$

Random Forest Formula:

$$RFfi_i = \frac{\sum_j normfi_{ij}}{\sum_{j \in all\ features, k \in all\ trees} normfi_{jk}}$$

- RFfi sub(i)= the importance of feature i calculated from all trees in the Random Forest model
- normfi sub(ij)= the normalized feature importance for i in tr

ii) Performance Evaluation:

Accuracy:

(TP+TN)/(TP+TN+FP+FN)

TP: True positives

TN: True Negatives

FP: False Positives

FN: False Negatives

Logloss:

Logloss is a cross-entropy based metric that measures the quality of predictions as compared to accuracy. It is a gauge of additional error coming from the predictions as compared to the true values. Logloss value of 0 implies a perfect model. We will be using logloss as a performance metric in our project.

MSE (Mean Square Error):

Mean squared error is used to measure the average of the square of errors. MSE like logloss is a measure of the quality of predictor or estimator. We will be using MSE as a performance metric in our project.

Root Mean Square Error (RMSE):

$$\mathrm{RMSE}_{fo} = [\sum_{i=1}^{N} (z_{f_i} - z_{o_i})^2 / N]^{1/2}$$

where $(z_{fi} – z_{oi})2$ = error, squared ad N is sample size

TF-IDF (Term Frequency – Inverse Document Frequency)

The text vectorizer Term frequency-inverse document frequency converts the text into an useable vector. Term Frequency (TF) and Document Frequency (DF) are combined in this idea (DF).

The term frequency refers to the number of times a term appears in a document. The frequency of a term in a document reflects how essential it is. Term frequency displays each text in the data as a matrix with the number of documents in the rows and the number of distinct terms in the columns.

The amount of papers that include a specific term is known as document frequency. The frequency of a term in a document reflects how common it is.The weight of a term is determined by the inverse document frequency (IDF), which seeks to minimise the weight of a term if the term's occurrences are dispersed throughout all documents. The following formulas can be used to determine the TF-IDF:

$$idf_i = \log(\frac{n}{df_i})$$

$$w_{i,j} = tf_{i,j} \times idf_i$$

Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm, used for classification problems, but it can also be used for regression problems. It finds a hyperplane in an n-dimensional space, where n is the number of features, that optimally classifies the data points. It may be possible to find multiple hyperplanes that separate the classes but the objective of SVM is to find the hyperplane that distinctly separates the classes with maximum margin on both sides. A hyperplane is a boundary plane that acts as a separator between classes so a data

point may exist on one side of the hyperplane. Each side of this plane can be considered as the region for which the data points inside belong to a particular class.

Support vectors are the data points close to the hyperplane, that is, these are the data points that have the greatest impact on the position and orientation of the plane. So, these support vectors are used while maximizing the margins of the hyperplane for a classification problem. By maximizing the margins around a hyperplane, the authors ensure higher accuracies of prediction as data points exist away from the plane and can be distinctly identified with the associated class. Linear SVM is used for linearly separable data, which means that if a dataset can be classified into two classes using only a single straight line, it is called linearly separable data, and the classifier employed is called Linear SVM. A kernel is a function that takes the original non-linear problem and transforms it into a linear one within the higher-dimensional space. RBF kernels are the most generalized form of kernelization and is one of the most widely used kernels due to its similarity to the Gaussian distribution. The RBF kernel function for two points X_1 and X_2 computes the similarity or how close they are to each other. This kernel can be mathematically represented as follows:

$$K(X_1, X_2) = exp(-\frac{||X_1 - X_2||^2}{2\sigma^2})$$

where,

1. ’σ‘is the variance and our hyperparameter
2. $||X_1 - X_2||$ is the Euclidean (L_2-norm) Distance between two points X_1 and X_2

4. Advantages and disadvantages of the various methods along with our method.

Advantages of our method

Using the random forest method reduces overfitting in decision trees and helps to improve the accuracy, Random Forest method is also very effective in both classification and regression problems, and it works well with both categorical and continuous values. Missing values are also automated and are hence not a problem in Random Forest method. The performance of the model in terms of accuracy or quality measures such as errors or logloss is also very good as it uses an ensemble of uncorrelated decision trees to make its decision hence making the prediction accurate and with less errors. . Hence it works extremely well with our dataset containing both numeric and categorical values.

TF-IDF is easy to calculate. It is an easy way to extract the most descriptive keywords in a document and measures the uniqueness and relevance of the content (in this case attack strings). SVM works relatively well when there is a clear margin of separation between classes. It is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples and is relatively memory efficient.

Disadvantages of our method

Random Forest algorithm has a few drawbacks such as a high computational cost and requirement owing to the fact that it uses multiple decision trees. It also requires a lot of time to train as it needs to get the best decision from a committee of decision trees. The dataset that we are using is also specific and hence the model requires a broader range of data for more effective and wide-scale predictions. TF-IDF is only useful as a lexical level feature. Synonymities are neglected. It doesn't capture semantic features. SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.

Advantages and Disadvantages of other methods

The random forest model is scalable whereas other models used in existing literature are not suitable for large datasets where target classes overlap which is quite possible in the data for network traffic. However, machine learning algorithms like SVM are relatively more memory efficient and work well when the number of features with distinct separation increases.

Advanced methods like Genetic algorithms are more efficient at processing datasets with high noise and solve loosely formed solution spaces well but finding suitable fitness functions, defining the representation of the given problem and choosing various other parameters like mutation rate, population size and crossover rate make it very complicated and infeasible to generalise for various datasets especially for network traffic data which is quite dynamic in nature.

III. LITERATURE SURVEY

	Paper Name	Methodology	Technical Parameters	Improvements & Future Work
1	Mathematical Validation of Proposed Machine Learning Classifier for Heterogeneous Traffic and Anomaly Detection	The authors propose and develop an ML classifier for heterogenous traffic classification and categorizing collected events within networks. The proposed ML layer consists of an input layer followed by a hidden layer and an output layer. The methodology involves first a preprocessing phase where the events of the network traffic are divided into trining and testing sets. This is followed by training of the proposed algorithm and then testing. New traffic events are then fed to the final classifier model to determine whether an event is normal or malicious and intrusive in nature.	The technical parameters of the proposed system include a three-layer perceptron. The train test split is 80-20. Numerical variables are encoded (successful as +1 and anomaly as -1). Optimization model is used by defining a function to determine weights of the model.	The large dimensions and heterogeneity of data proves to be a limitation for effective classification. Future work involves designing an effective IDS and firewall using proposed classifier to improve network security.
2	Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection	In this paper, the authors propose a novel multistage optimized ML framework for NIDS. The proposed framework reduces computational complexity while keeping performance. The authors study the effects of oversampling techniques on training data and compare two feature selection techniques (correaltion and information gain-based approaches) and outline their effect on time and computational complexity as well as detection power. The two IDS datasets used are the CIDCS 2017 and the UNSW-NB 2015 datasets. The proposed framework successfully reduces the training sample size by upto 74% and the feature set size by about 50%. Detection accuracies range over 99% for both the datasets	The authors use Z-score and SMOTE techniques for data pre-processing. Feature selection is done using Information gain-based approach as well as correlation based approach. Hyperparameter optimisation is done by Random Search, meta heuristic algorithms (PSO , GA) and Bayes optimization methods (Gaussian and TPE). The stages are combined to build the optimized classifier that determines normal or attack instances.	An area of improvement is for more insights to be extracted from high volume network data and continuously changing environments. Future work mentioned by the authors are investigating the impact of combining supervised and unsupervised ML techniques for better IDS and anomaly detection.

3	A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM	In this paper, the authors propose a Double-Layeres Hybrid Approach (DLHA) for Intrusion Detection. Principal Component Analysis (PCA) was used to study the common characteristics of different attack types (even more uncommon attacks such as R2L and U2R). Using PCA many variables were created of the different attak types to maximize variance. The DLHA uses Naïve Bayes Classifer in the first layer for DoS and Probe attacks. Layer 2 is SVM for R2l and U2R. The NSL-KDD dataset was used for the proposed model. DLHA shows a detection rate of 96.67% for R2L and 100% for U2R as well as an overall detection rate of 93.11%.	Data preparation phase involves divind into 2 groups for the different kinds of attacks. Data Transformation techniques used are Intersectional Correlated Feature selection followed by Normalization, One hot encoding and PCA. Downsampling of frequent instances is also performed. The ML classifiers used are Naive Bayes for Group 1 and SVM with 10 fold cross validation for Group 2. Evaluation metrics used are Accuracy, F1 Score, Precision, Recall and False Alarm Rate.	A needed improvement pointed out by the authors are a need for better detection of all types of attacks. Future work mentioned are application of the proposed approach on a network environment to categorize more than 4 types of attacks,
4	A Hybrid Unsupervised Clustering-Based Anomaly Detection Method	In this paper, the authors present an unsupervised anomaly detection method which combines Sub-Space Clustering (SSC) and One Class Support Vector Machine (OCSVM) to detect attacks without any prior knowledge. The dataset used is the NSL-KDD dataset. First the categorical features are converted to numeric using one-hot encoding. The dataset is split into 4 one-class attack subsetsand a mixed attack subset. F-test and feaure selection is then performed followed by Normalization with Standard Scalar. This is followed by Parameter tuning and detection threshold calculation. Performance evaluation is then performed.	The technical steps used by the authors are one-hot encoding, F-test, Data normalization using Standard Scalar. The proposed SSC-OCSVM algorithm involves the initialisation stage followed by the clustering and learning stage and the evidence accumulation stagewhich update the dissimilarity vector on each partition for every sub space (based on EA clustering). Final step is Anomaly detection. Performance measures in the form of Confusion matrix, Recall, FAR and the ROC Curve	Future Work mentioned by the authors are implementing parallelization of the proposed algorithm as future research.
5	Web-APT-Detect: A Framework For Web-based Advanced	In this paper, the authors propose an unsupervised anomaly detection algorithm, Web APT Detect (WAD) which uses self translation through an encoder	The technical parameters involved in the proposed framework are token parsing, building vocabulary for HTTP	The authors speculate that different self-translation machine

	Persistent Threat Detection Using Self-translation Machine With Attention	decoder by means of attention mechanism. The purpose of the proposed framework is to detect malicious patterns in HTTP requests. The dataset used is the CSIC 2010 dataset. The F1 Score obtained via testing is 0.9844. The training phase to process HTTP request has the following steps: token parse, building vocabulary, training model. The main steps for anomaly detection are token parse, vocabulary, and BLEU metric.	request, optimizer using back-propagation algorithm for parameter adjustments. BLEU metric is used to measure the difference between the input token and the translated token sequence. Self-translation is also proposed using attention mechanism which consists of an encoder and a decoder. The proposed framework uses LSTM layers with Adam Optimizer with weight decay of 1e-9 and 30 epochs. The main performance metric is the F1-Score	structures can be used to further improve detection performance. Future work is intended to be on further optimizing the design of the network structure and trying out more advanced attention mechanisms.
6	TIDCS: A Dynamic Intrusion Detection and Classification System Based Feature Selection	In this paper, the authors propose two models for intrusion detection and classification scheme based IDS and Classification System – Accelerated (TIDCS-A) for network security. TIDCS performs feature selection by reducing the number of features in input data using a new algorithm. The high ranked features are used for the classification. The authors propose a periodic system cleansing for renewing trust relationships as well as a dynamic algorithm for cleansing states and reducing exposure of nods. The two datasets used are the NSL-KDD and the UNSW dataset. The proposed system incorporates the nodes past behaviour with ML algorithm. The results obtained are 91% accuracy of detection for UNSW with the proposed TIDCS	For feature selection Linear Discriminant Analysis (LDA) and Principal Component Analysis (PCA) is used. The LogitBoost based algorithm is used as IDS using ensemble classification approcach. Heuristic search is used with genetic algorithm to give the results. Filter and Wrapper methods are also used to enhance performance of the model. Various pahes include the problem formulation phase, the random phase, improvement phase, classifer decision and storing and combined decision phase and accelerated detection phase.	The authors propose further research into improvements of the various phases of the proposed system such as the accelerated decision phase for improvements and future work.
7	Deep Learning Approach	In this paper the authors propose a deep learning and self-taught	The data preprocessing step consists pf 1 to n	The authors propose future

	Combining Sparse Autoencoder With SVM for Network Intrusion Detection	learning IDS based on STL framework. It has applications in feature learning and dimensionality reduction. Training and testing time are reduced and the prediction accuracy of SVM is also increased considerably. Spare autoencoder mechanism is used for unsupervised feature representation. The new features are then entered into the SVM for detection and classification. The obtained results are compared to existing models such as Naïve Bayes, RF and standard SVM. The proposed method outperforms them. Data set used is the NSL-KDD dataset.	Numericalization followed by Normalization. The STL phase uses a sparse autoencoders which is fed with the normalized data from the pre-processing step. The first encoder gives a trained parameter set which is fed to a second sparse auto-encoder. The resultant features are fed into SVM Classifier for Classification and Detection.	work and further improvement by using multiple stages of STL and a hybrid feature learning model for better feature representation and dimensionality reduction mechanisms. Another improvement is for parallel implementation or GPU acceleration of the proposed system.
8	Enhanced Network Anomaly Detection Based on Deep Neural Networks	In this paper the authors explore the suitability of deep learning techniques for anomaly-based IDS. They propose anomaly detection models based on different DNN structures including CNNs, Autoencoders and RNNs. The datset used was the NSL-KDD dataset. The experiments performed were on a GPU test bed. Conventional ML techniques were also implemented namely Extreme ML, KNN, DT, RF, SVM, Naïve Bayes and quadratic discriminant analysis. Well know classification metrics were used for evaluation such as ROC, AUC, PR curve, accuracy, and others.	The DNN structures used by the authors are Autoencoder, LSTM, CNN. The different types of autoencoders used are sparse autoencoder, denoising autoencoders, contractive autoencoders and convolutional autoencoders. There is 1 Input Layer followed by two hidden layers following which there is a bottleneck layer which diverges to two more hidden layers and an output layer or to a fully connected layer and then to an output layer. Evaluation metrics are ROC, AUC< accuracy and others.	The authors mention future work in the direction of investigating deep learning as feature extraction tool for deficient data representation for anomaly detection problems.
9	Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning	In this paper the authors look at various classification techniques for intrusion detection systems. Techniques such as multi layer perceptron, SVM and others have been explained here. An efficient classification algorithm	Data preprocessing is done by excluding the non-numeric features. The ML models that are applied to the NSL-KDD dataset are SVM, RF and	Future work mentioned by the authors ae to further explore ELM to investigate its performance in

	Machine for Intrusion Detection	is required for IDS as it is used in analysing huge traffic data. Other techniques used in this paper are Random Forest (RF) and Extreme Learning Machine (ELM). The dataset used is the NSL KDD dataset. The data is also preprocessed well. Evaluation is done using metrics like accuracy, precison and recall.	ELM. The SVM model uses the RBF kernel. Evaluation metrics are Accuracy, Precison and Recall. Train test split is 90-10.	feature selection and feature transformation techniques.
10	A Machine-Learning-Driven Evolutionary Approach for Testing Web Application Firewalls	In this paper, the authors propose an ML-Driven approach and an evolutionary algorithm for detecting defects in web application firewalls that can lead to possible SQL Injection attacks. The ML-Driven automatically generates a large set of attacks and tests the firewall system. The ML bbased system also learns which patterns are better to break the system defence and evolves to generate new such attacks. ML is used to learn attack patterns from the results of previous attacks. The created tool is tested using an service called ModSecurity which is an opensource WAF.	The ML-Driven tool generates a diverse range of attacks. The different types of attacks are: • SQL Injection attacks – Boolean Attacks, Union attacks and Piggy-backed attacks • Grammar Based Random Attack Generation (RAN) • Machine Learning Guided Attack Generation The ML-Driven system prforms and learn various types of attacks from the results of previous attacks.	The authors propose investigating automated approaches to generate effective patches for WAF as future work. They also plan to research strategies to improve the effectiveness and the efficiency of the ML-Driven system by fixing data imbalance issues.
11	Enhancing the effectiveness of Web Application Firewalls by generic feature selection	In this paper the authors signify the importance of feature selection in HTTP traffic filtering for WAFs. The paper focuses on generic feature selection measure which is tested on low level package filters. Datasets used is the ECML/PKDD 2007 dataset and the CSIC 2010 dataset. The statistical properties of both datasets are analysed and more insight is provided about their nature and quality. Appropriate instances of GeFS feature selection are also determined. Different classifiers are used to test accuracies. The results suggest that 63% data could be	There are different types of GeFS measures used namely the mRMR feature selection measure, the CFS measure and the Mahalanobis distance. The GeFS measure is optimized by adding an additional positive variable. Various classifiers are used such as C4.5, CART, Random Tree and Random Forest and the performance is noted.	Future Work involves testing the GeFS measure with more classification techniques as well as trying out combinations of GeFS measures.

		removed without damaging performance.		
12	Performance Modeling and Analysis of Network Firewalls	In this paper, the authors propose an analytical queuing method baed on embedded Markov chain. This method is used to study and analyze the performance of rule based firewall systems when subjected to normal traffic flows and DoS attacks. The DoS attacks target different rule positions and this method can counter that. The equation for key features and performance measures of engineering and design significance are derived. The important features are throughput, packet loss, packet delay, CPU utilization. The authors also verify and validate the model using simulation and experimental means.	A finite queuing model is created to represent the behaviour and study the performance of the firewall. Model Analysis and Solutions for derivation of equation for key features is performed using mathematical models. DoS and DDoS attacks with multiple flows are also handled by the model The problem of infinite queuing is also handled. Heuristic used is MCR (Marginal Confidence Rule).	Improvement to the proposed method involves more flexibility in the firewall ruleset as well as allowing for larger rulesets for protection against more complex algorithmic attacks. Future work involves study of performance of firewalls when implementing the mitigation solution of dynamic re ordering of rulesets.
13	On Dynamic Optimization of Packet Matching in High-Speed Firewalls	In this paper, the authors present a novel algorithm for maximizing early rejection of unwanted traffic flows and they also propose a novel packet filtering technique using dynamic optimization that uses statistical search trees to utilize traffic characteristics and minimize the average packet matching time. The search data structure is optimized to enable timely adaptations to change in network traffic. The methods propose employ effective techniques. Evaluation is extensive using Internet traces and memory and time requirements are considered and optimized.	Early Traffic Rejection optimization is done by two methods – Rejection Address-Space-Based Optimization and Dynamic Rule Selection. Statistical optimization of filtering trees is don eby the exploring the locality of matching properties in firewall filtering and to check packet flow properties and packet field properties. Other methods are Statistical Matching Tree construction and Matching Tree Construction using Alphabetic Trees as well as Policy Matching Algorithms using Alphabetic Tree and Tree Reconstruction	Further optimizations to Tree constructions have been mentioned by the authors. Future work involves further exploration of statistical methods for minimization of packet matching time and optimization of memory requirements.
14	Analysis of intrusion	In this research work, two diverse Machine Learning	Feature extraction is done using deep learning	False positives mustbe decreased.

	detection in cyber attacks using DEEP learning neural networks	techniques are prepared which include both supervised and unsupervised, for Network Intrusion Detection. Naive Bayes (supervised learning) and Self Organizing Maps (unsupervised learning) are the presented techniques. Deep learning techniques such as CNN is used for feature extraction. These remain provisional chances adaptation technique and pointer variables transformation. The two machine learning procedures are prepared on both kind of transformed dataset and afterward their outcomes are looked at with respect to the correctness of intrusion detection.	through CNN and the models used for the detection of attacks include supervised and unsupervised learning models, Naive Bayes and Self Organising Maps respectively.	Of these sorts of slips, the false negatives are more hazardous for IDS, on the grounds that it does not recognize a caution as an occurrence, and gives it a chance to pass through the sifting technique. This proposed research work can be extended by hybridizing deep learning algorithms and measuring the performance
15	Research Trends in Network-Based Intrusion Detection Systems: A Review	The authors of this article present a review of the research trends in network-based intrusion detection systems (NIDS), their approaches, and the most common datasets used to evaluate IDS Models. The analysis presented in this paper is based on the number of citations acquired by an article published, the total count of articles published related to intrusion detection in a year, and most cited research articles related to the intrusion detection system in journals and conferences separately. Based on the published articles in the intrusion detection field for the last 15 years, the authors of this article also discuss the state-of-the-arts of NIDS, commonly used NIDS, citation-based analysis of benchmark datasets, and NIDS techniques used for intrusion detection. A citation and publication-based comparative	Keywords and terminologies related to IDS, datasets, methodologies and issues are used along with the number of citations of the papers. Filters are used for intrusion detection systems based on false positive rates, true positive rate and F1 score.	In the future, the authors may implement the less cited bio-inspired approaches that have few publication count values for articles related to network intrusion detection systems for future subsequent work. They may determine whether the less cited approaches with fewer counts of published articles are equally applicable to achieve an efficient and effective intrusion detection model.

		analysis to quantify the popularity of various approaches are also presented by the authors.		
16	A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions	In this paper, the authors survey the intrusion detection problem by considering algorithms from areas such as ML, DL, and SWEVO. The survey is a representative research work carried out in the field of IDS from the year 2008 to 2020. The paper focuses on the methods that have incorporated feature selection in their models for performance evaluation. The paper also discusses the different datasets of IDS and a detailed description of recent dataset CIC IDS-2017. The paper presents applications of IDS with challenges and potential future research directions.	Feature selection, model, performance measures, application perspective, challenges and future directions.	the future work may reinforce the study to explore different capabilities for using and leveraging the information provided in context with attack detection. Though ML techniques have showcased promising results for intrusion detection, still their scal-ability with real-time intrusion detection needs to be envisioned.
17	A Distributed Deep Learning System for WebAttack Detection on Edge Devices	In this article, based on distributed deep learning, the authors propose a web attack detection system that takes advantage of analyzing URLs. The system is designed to detect web attacks and is deployed on edge devices. The cloud handles the above challenges in the paradigm of the Edge of Things. Multiple concurrent deep models are used to enhance the stability of the system and the convenience in updating. The authors have implemented experiments on the system with two con-current deep models and compared the system with existing systems by using several datasets.	Feature representation, data normalisation, Data and feature discrimination, models for classification such as RNN, GRU, Naive Bayes, LSTM and X-Means.	This system has the potential to be useful in the real world because of its automatic feature selection, convenience of updating, and stability of protecting from attacks against distributed deep models.
18	A new WAF architecture with machine learning for	The authors aim to show a distributed WAF architecture, using ML classifiers as one of its components. Instead of having an enforcement point that	Linear Support Vector Machine, Perceptron and Logistic Regression models are used. Evaluation is done using	Future work could be to expand classifiers' use for a more in-depth analysis of the

	resource-efficient use	analyzes the complete HTTP protocol for violations in this architecture, the authors have a trained classifier to detect them. The first part of this work verifies the viability of using classifiers based on their metrics, such as accuracy and recall. The authors analyze two datasets and make comparisons about their use. The second part of this paper compares ML models' prediction processing time and a rules-based engine's processing time. The authors also show that a classifier can find errors in the classification of a dataset generated by a WAF based on rules. They present samples and experimental codes to show the difference in approaches.	accuracy, precision and recall.	request headers and body. Addition- ally, the authors may see how to use other mechanisms in conjunction with classifiers, such as reputation mechanisms.
19	The Analysis of Firewall Policy Through Machine Learning and Data Mining	In this paper, an automated model based on machine learning and high performance computing methods is proposed by the authors for the detection of anomalies in firewall rule repository. To achieve this, firewall logs are analysed and the extracted features are fed to a set of machine learning classification algorithms including Naive Bayes, kNN, Decision Table and HyperPipes. F-measure, which combines precision and recall, is used for performance evaluation. In the experiments, kNN has shown the best performance. Then, a model based on the F-measure distribution was envisaged. 93 firewall rules were analysed via this model. The model anticipated that 6 firewall rules cause anomaly. These problematic rules were checked against the security reports prepared by experts and each of them are verified to be an anomaly.	Machine learning classification algorithms including Naive Bayes, kNN, Decision Table and HyperPipes are used and F-measure, which combines precision and recall, is used for performance evaluation.	The researchers who will carry out their studies about this subject are bound to encounter vast amounts of data. Because of this, they may prefer high performance computing methods in order that they can shorten the duration of analysis. In future work, whether machine learning methods can be adjusted so that they can operate on a firewall at real-time can be studied.

20	Intrusion Detection System Through Advance Machine Learning for the Internet of Things Networks	In this paper, a two-stage hybrid method is proposed. First, the genetic algorithm (GA) is applied to select appropriate features to improve the accuracy of the proposed framework. Next, the well-known machine learning (ML) algorithm, including the support vector machine (SVM), ensemble classifier, and decision tree are employed.	Genetic Algorithm for feature selection and SVM, Decision Tree and ensemble classifier for detection.	Future research work may focus to detect anomalies in IoT and IoMT (Inter- net of Medical Things) networks using hybrid features selection, such as PSO-GA and multiclass classifica- tion methods.
21	Feature analysis of encrypted malicious traffic	In this paper, the authors apply three machine learning techniques to the problem of distinguishing malicious encrypted HTTP traffic from benign encrypted traffic and obtain results comparable to previous work. The authors then consider the problem of feature analysis in some detail. Previous work has often relied on human expertise to determine the most useful and informative features in this problem domain. They demonstrate that such feature-related information can be obtained directly from machine learning models themselves.	Support Vector Machine, Random Forest and XGBoost are used which are evaluated using F1-score, AUC, Precision and recall.	For further research, it would be useful to have a larger dataset that could be mined for more subtle features. Another direction for related research would be to field a system based on the ma- chine learning techniques considered here, so that the effective- ness and robustness of such an approach could be analyzed under real-world and changing conditions.
22	DNS Firewall Based on Machine Learning	The authors of this work provide a study to implement a DNS firewall solution based on ML and so improve the detection of malicious domain requests on the fly. For this purpose, a dataset with 34 features and 90 k records was created based on real DNS logs. The data were enriched using OSINT sources. Exploratory analysis and data preparation steps were carried out, and the final dataset submitted to different Supervised	The data was enriched using OSINT sources. Models like SVM, Logistic Regression and K-Nearest Neighbours were used and evaluated using accuracy, precision, recall and F1-score.	This work was carried out in a laboratory environment, so there is a great limitation in terms of tests in real scenario. With the creation of a prototype and testing in a real environment, one will certainly observe different

		ML algorithms to accurately and quickly classify if a domain request is malicious or not..		results due to both the volume of orders and the variance in their characteristics. These tests will be essential to improve and make this solution more robust.
23	Malicious attack detection approach in cloud computing using machine learning techniques	We propose an accurate and complete approach for detecting and preventing assaults in cloud computing environment via the use of a machine learning techniques both supervised and un-supervised. The operational findings demonstrate that the suggested approach substantially increases attack detection, network security correctness, dependability, and accessibility in cloud computing environment, while drastically reducing false alarms.	Clustering and Classification were done using Support Vector Machine, Continuous k-Nearest Neighbours and ensemble method which were evaluated using accuracy, precision and recall.	Future work could be to expand classifiers' use for a more in-depth analysis of the request headers and body.
24	Ensemble Classifiers for Network Intrusion Detection Using a Novel Network Attack Dataset	The authors of this paper provide a comprehensive analysis of some existing ML classifiers for identifying intrusions in network traffic. It also produces a new reliable dataset called GTCS (Game Theory and Cyber Security) that matches real-world criteria and can be used to assess the performance of the ML classifiers in a detailed experimental evaluation. Finally, the authors propose an ensemble and adaptive classifier model composed of multiple classifiers with different learning paradigms to address the issue of the accuracy and false alarm rate in IDSs.	The classifiers included NB, logistic, multilayer perceptron (neural network), SMO (SVM), IBK (*k*–nearest neighbor), and J48 (decision tree).	Future work could be to expand classifiers' use for a more in-depth analysis of the request headers and body. Addition- ally, the authors may see how to use other mechanisms in conjunction with classifiers, such as reputation mechanisms.
25	Deep Learning for Encrypted Traffic Classification: An Overview	In this article, the authors introduce a general framework for deep-learning based traffic classification. The authors present commonly used deep learning methods and their	Models like CNN, CNN+LSTM, LSTM, RKHS+CNN were studied based on the computational	There are potentially many ways to define auxiliary tasks for traffic classification

		application in traffic classification tasks. Then they discuss open problems, challenges, and opportunities for traffic classification.	complexity and number of packets needed.	without the need for additional labeling. Many variations of multi-task learning were used successfully for natural language processing and computer vision. However, it has not been studied for network traffic classification.

IV. OVERALL ARCHITECTURE

1. ARCHITECTURE DIAGRAM

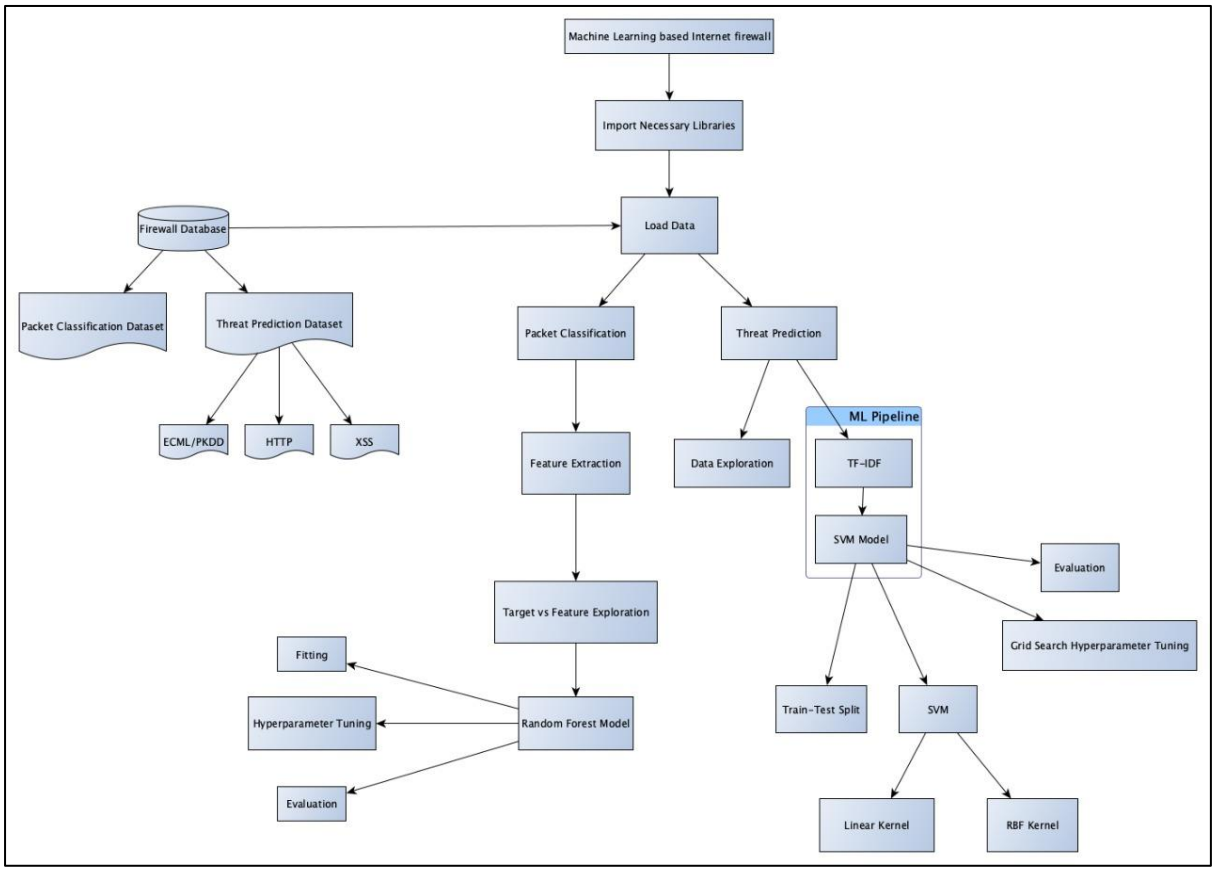


Fig 4.1: Proposed System Flowchart/Architecture

2. FLOW OF ARCHITECTURE

The database is divided into two parts as packet classification dataset (Internet Firewall dataset – UCI) and threat prediction dataset where the threat prediction dataset is formed by combining three datasets; ECML/PKDD, HTTP and XSS datasets.

We start by importing the necessary libraries and loading the dataset into memory for performing operations on it. The first part of our project deals with the classifications of packets from internet firewall data. The steps involved in the classification of internet firewall packets starts with feature extraction and target versus feature exploration. Various types of visualizations are provided for the same as part of the EDA (exploratory data analysis). The model chosen for this classification is random forest which involves independent classification results from multiple decision trees that come to consensus over the final classification outcome. The model is fit to our dataset then we tune the hyper parameters to obtain the best results which are evaluated using various parameters such as logloss, mse, rmse and accuracy.

The second part of our project deals with threat prediction using the internet firewall data. We explore the data and visualise features to get an overview of the dataset. Multiple different

attack datasets are combined to form the resulting dataset. TF-IDF vectorizer is used for analysing the attack string fields. The model chosen for internet firewall threat prediction is the Support Vector Machine (both linear and rbf kernels are used). To use the model, we first split the dataset into training and testing data and create a model pipeline with TF-IDF and subsequent SVM. The SVM pipelined model is fit to the data with different kernel functions for the model and a parameter field is defined on which Grid Search Cross Validation is performed. Linear kernel SVM and RBF kernel SVM are used and other hyperparameters are also used and the predictions for each are evaluated using accuracy, precision, recall and F1-score as evaluation parameters. Classification report and confusion matrix of the results as well as obtained best parameters are mentioned as well.

V. METHODOLOGY

Import Necessary Libraries

First we need to import the necessary python libraries for the application such as numpy, pandas, time, matplotlib, seaborn, plotly, h2O (for Random forest), sklearn (for SVM, Grid Search and other functionality) and xml and json reading packages for performing actions on the Threat Prediction module datasets.

Dataset

The dataset used for Packet Classification is the Internet Firewall Dataset from UCI repository where 65532 instances of internet traffic logs through Firat University network are available. The logs have 12 features in total out of which one feature corresponds to the action performed by the firewall. We will consider this feature as the class for our analysis where the action can be classified as; allow, deny, drop, and reset-both. The remaining 11 features will be fed to the model for classification. The features of the dataset are Source Port, Destination Port, NAT Source Port, NAT Destination Port, Bytes which are divided into Bytes Sent and Bytes Received, Packets, Elapsed Time in seconds.

The datasets used for Threat Prediction module are a combination of the ECML/PKDD 2007 dataset along with the HTTP parameters dataset and the XSS dataset. All these datasets are parsed together into one common dataset. They are then merged and cleaned into one large dataset for a big dataset. There are more than 90000 entries in the common dataset with attributes pattern and type. This dataset is fit into an SVM model to accurately predict the corresponding threat.

Packet Classification Module – Classification of actions of packet data

Feature Exploration

Feature Exploration is done for both numerical and categorical features. The numeric features are Bytes, Bytes Sent, Bytes Received, Packets, Elapsed Time, pkts_sent and pkts_received. Th logarithmic distribution of these features is plotted. For categorical features, source port, destination port, NAT Source and NAT Destination Port, the frequency distribution of entries is plotted, and a source/destination visualization is created to show the communication of packets. Same procedure is followed for the NAT Source and Destination Ports.

Target vs Feature Exploration

For the numeric features, we plot the distribution of the features with respect to the 4 labels of actions in our dataset namely allow, drop, deny and reset-both. This is done for all the numeric features in the dataset. For categorical features, we visualize a crosstable target and feature which gives us a visualization for the categorical feature with respect to the action.

Random Forest Model – Fitting, Hyperparameter Tuning and Evaluation

To classify the data, we use the Random Forest model which is an ensemble learning technique using multiple decision trees. The target is converted to categorical, and the dataset is split 70-30 train and test. The random forest model is the fit using appropriate hyperparameters tuned to obtain the best results. Hyperparameters that will be considered are number of cross validations, number of decision trees, max depth of the trees, minimum rows, number of folds (cross-validations), stopping metric, stopping rounds and stopping tolerance. Various evaluation metrics are also used such as accuracy, AUC, error, logloss, mse, r2, rmse and other cross validation metrics. The scoring history is also visualized with respect to logloss. Variable importance is also plotted, and a confusion matrix is also displayed to show the actual vs predicted results. It is important to understand the effectiveness of our model and we can obtain the following by looking at performance metrics of the implemented model. Evaluation of the performance of the model is an essential step for the functioning of the firewall in real world application.

Threat Prediction Module – Classification of whether given entries are valid or contain a threat

Preprocessing and Data Exploration

The data in the combined dataset of the ECML/PKDD, HTTP parameters and the XSS dataset are explored by visualizaing the frequency of each entry. We see the distribution of valid, path traversal, sql injection, xss and command injection entries in the dataset along with their frequencies in the combined dataset.

ML Pipeline (TF-IDF + SVM) and GridSearchCV

To prepare the dataset to fit into SVM model, we first split the dataset into train and test sets. A pieline will be created using which uses TF-IDF and SVM and this will serve as the model

for classification. The TF-IDF Vectorizer is used to assign an importance to each target and then the SVM model is used for classification of the extracted features.

GridSearchCV: Grid Search cross validation is performed for hyperparameter tuning purposes to obtain best fit. The parameter grid consists of three parameters namely the tfidf vectorizer ngram range, the SVC C penalty parameter of the error term and both linear and rbf kernels.

Evaluation

The created pipeline will be evaluated using grid score as well as classification metrics such as accuracy, precision, recall and f1-score. The classification report is also created. A confusion matrix will also be created for evaluation of the model.

STEP-WISE IMPLEMENTATION OF MODULES

A. PACKET CLASSIFICATION

1. IMPORT necessary libraries (numpy, h2o, matplotlib, seaborn, time, pandas)
2. READ packet classification dataset
3. VIEW dataset head, info and description (head(), info(), describe() functions)
4. PLOT frequency of target actions (allow, deny, drop, reset-both)
5. LOG TRANSFORM numerical features
6. PLOT transformed numerical features
7. VIEW top 20 entries (frequency wise) for each categorical feature (Source Port, Destination Port, NAT Source Port, NAT Destination Port)
8. PLOT top 20 entries for each categorical feature.
9. PLOT packet Source to Destination port and NAT Source to NAT Destination port scatter plot
10. PLOT action-wise distribution of numerical features
11. PLOT action-wise distribution of categorical features
12. INITIALIZE h2o (for creating Random Forest model)
13. SPLIT dataset into train and test datasets (75-25)
14. CREATE Random Forest model with 5 cross validations
15. SET number of trees=10 with max depth=30 and min rows=5 and minimizing the logloss error metric.
16. TRAIN created model
17. DISPLAY Cross Validation Metrics Summary
18. PLOT logloss for training and validation(testing) for the 5 cross validations
19. PLOT variable importance
20. PROVIDE Training Set performance
21. PLOT Training Set Confusion Matrix
22. PROVIDE Test Set performance
23. PLOT Test Set Confusion Matrix

B. THREAT PREDICTION

1. MERGE XSS, ECML/PKDD and HttpParams datasets
2. READ combined dataset
3. VIEW dataset details (info(), describe() and other methods)
4. CALCULATE frequency of each attack type
5. PLOT distribution of attack types as pie chart
6. Import necessary libraries (sklearn for TF-IDF, SVC, pipeline, GridSearchCV, train_test_split, classification report and confusion matix)
7. SPLIT dataset into training and test sets (75-25)
8. CREATE ML pipeline of TF-IDF followed by SVM
9. SET parameter grid for the pipeline hyperparamters
10. SET GridSearchCV for the created pipeline
11. FIT the created grid
12. PRINT grid score (model score)
13. PRINT Classification Report
14. PRINT Confusion Matrix
15. PRINT Best Parameters obtained from GridSearchCV

Advantages of our method

Using the random forest method reduces overfitting in decision trees and helps to improve the accuracy, Random Forest method is also very effective in both classification and regression problems, and it works well with both categorical and continuous values. Missing values are also automated and are hence not a problem in Random Forest method. The performance of the model in terms of accuracy or quality measures such as errors or logloss is also very good as it uses an ensemble of uncorrelated decision trees to make its decision hence making the prediction accurate and with less errors. . Hence it works extremely well with our dataset containing both numeric and categorical values.

TF-IDF is easy to calculate. It is an easy way to extract the most descriptive keywords in a document and measures the uniqueness and relevance of the content (in this case attack strings). SVM works relatively well when there is a clear margin of separation between classes. It is more effective in high dimensional spaces. SVM is effective in cases where the number of dimensions is greater than the number of samples and is relatively memory efficient.

Disadvantages of our method

Random Forest algorithm has a few drawbacks such as a high computational cost and requirement owing to the fact that it uses multiple decision trees. It also requires a lot of time to train as it needs to get the best decision from a committee of decision trees. The dataset that we are using is also specific and hence the model requires a broader range of data for more effective and wide-scale predictions. TF-IDF is only useful as a lexical level feature. Synonymities are neglected. It doesn't capture semantic features. SVM algorithm is not suitable for large data sets. SVM does not perform very well when the data set has more noise i.e. target classes are overlapping.

VI. RESULTS

1) PACKET CLASSIFICATION

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65532 entries, 0 to 65531
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Source Port           65532 non-null  int64
 1   Destination Port      65532 non-null  int64
 2   NAT Source Port       65532 non-null  int64
 3   NAT Destination Port  65532 non-null  int64
 4   Action                65532 non-null  object
 5   Bytes                 65532 non-null  int64
 6   Bytes Sent            65532 non-null  int64
 7   Bytes Received        65532 non-null  int64
 8   Packets               65532 non-null  int64
 9   Elapsed Time (sec)    65532 non-null  int64
 10  pkts_sent             65532 non-null  int64
 11  pkts_received         65532 non-null  int64
dtypes: int64(11), object(1)
memory usage: 6.0+ MB
```

Fig 6.1: Packet classification dataset info

	Source Port	Destination Port	NAT Source Port	NAT Destination Port	Bytes	Bytes Sent	Bytes Received	Packets	Elapsed Time (sec)	pkts_sent	pkts_re
count	65532.000000	65532.000000	65532.000000	65532.000000	6.553200e+04	6.553200e+04	6.553200e+04	6.553200e+04	65532.000000	65532.000000	65532.
mean	49391.969343	10577.385812	19282.972761	2671.049930	9.712395e+04	2.238580e+04	7.473815e+04	1.028660e+02	65.833577	41.399530	61.
std	15255.712537	18466.027039	21970.689669	9739.162278	5.618439e+06	3.828139e+06	2.463208e+06	5.133002e+03	302.461762	3218.871288	2223.
min	0.000000	0.000000	0.000000	0.000000	6.000000e+01	6.000000e+01	0.000000e+00	1.000000e+00	0.000000	1.000000	0.
25%	49183.000000	80.000000	0.000000	0.000000	6.600000e+01	6.600000e+01	0.000000e+00	1.000000e+00	0.000000	1.000000	0.
50%	53776.500000	445.000000	8820.500000	53.000000	1.680000e+02	9.000000e+01	7.900000e+01	2.000000e+00	15.000000	1.000000	1.
75%	58638.000000	15000.000000	38366.250000	443.000000	7.522500e+02	2.100000e+02	4.490000e+02	6.000000e+00	30.000000	3.000000	2.
max	65534.000000	65535.000000	65535.000000	65535.000000	1.269359e+09	9.484772e+08	3.208818e+08	1.036116e+06	10824.000000	747520.000000	327208.

Fig 6.2: Description of the dataset

In Fig 6.1, Fig 6.2 the Internet Firewall Dataset for Packet Classification has been read and the info() function is used in Fig 6.1 to display the different features and number of non-null entries for each feature (65532 entries). In Fig 6.2, the describe() function is used to find different values such as count, mean, std and other values for the features in the dataset. There are 11 columns in the dataset with the Action column being the target column.

```
Target Frequency
allow          37640
deny           14987
drop           12851
reset-both        54
Name: Action, dtype: int64
```

Fig 6.3: Target Frequency

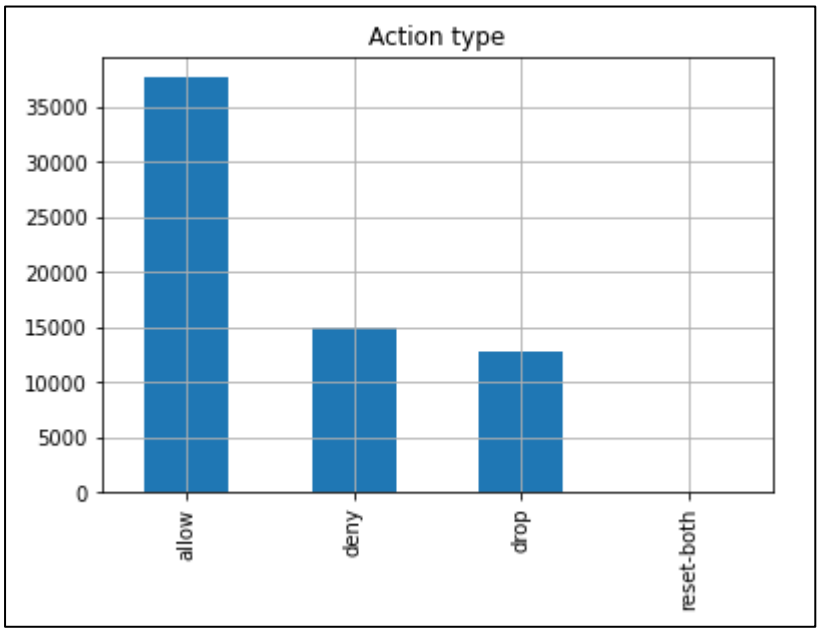


Fig 6.4: Target Frequency Bar Plot

In Fig 6.3, 6.4 the frequency of each of four possible actions namely allow, deny, drop and reset-both are both displayed (Fig 6.3) and plotted in the form of a bar chart (Fig 6.4)

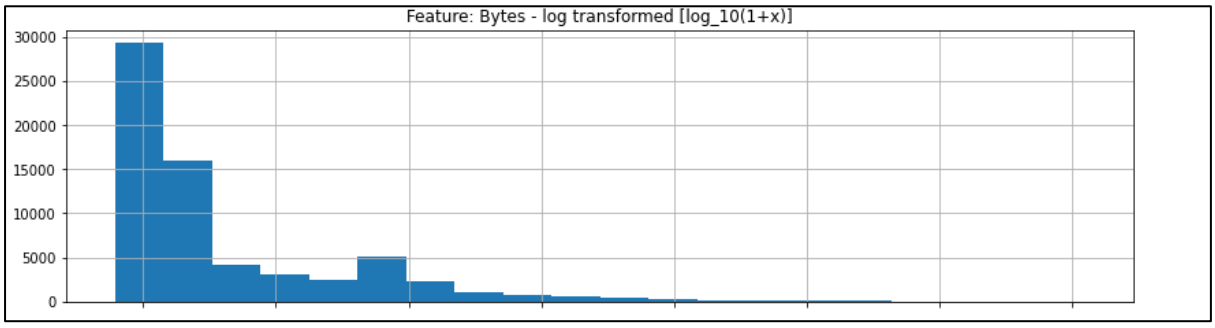


Fig 6.5: Log transformed features

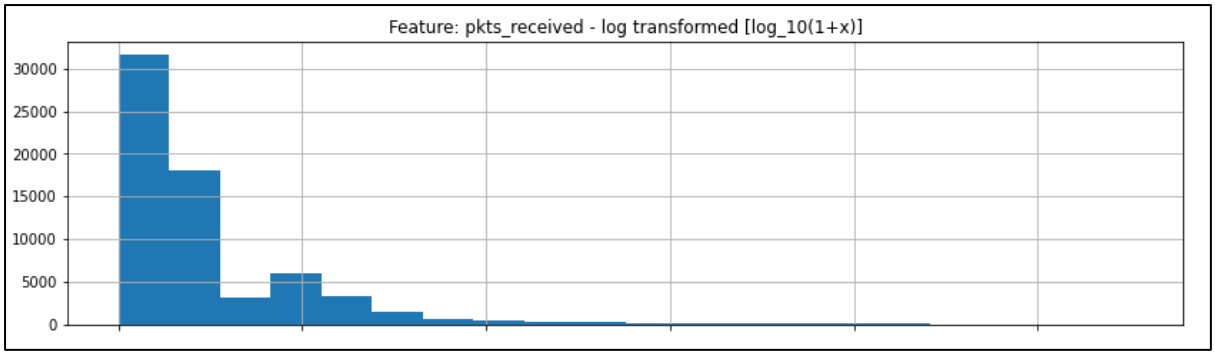


Fig 6.6: Log transformed features

In Fig 6.5, Fig 6.6 two numerical features (Bytes and pkts_received) have been log transformed [log_10(1+x) transformation] to create smaller values that can be easily fit into the Random Forest model.

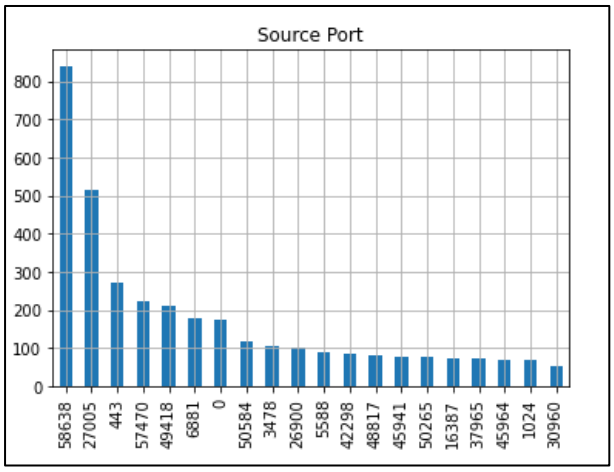


Fig 6.7: Plot of 20 most used Source Ports

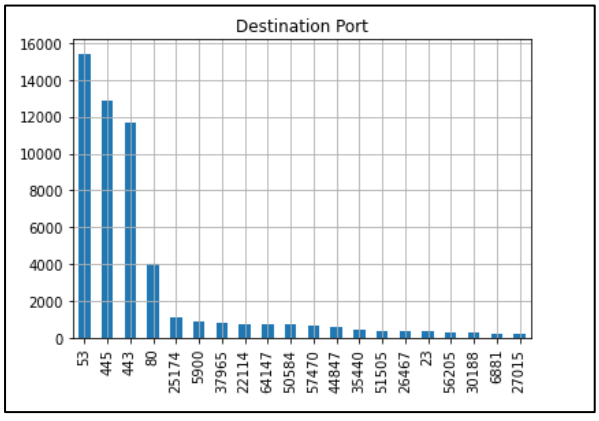


Fig 6.8: Plot of 20 most used Destination Ports

Fig 6.7, 6.8 show the frequency of the top 20 most used Source and Destination ports

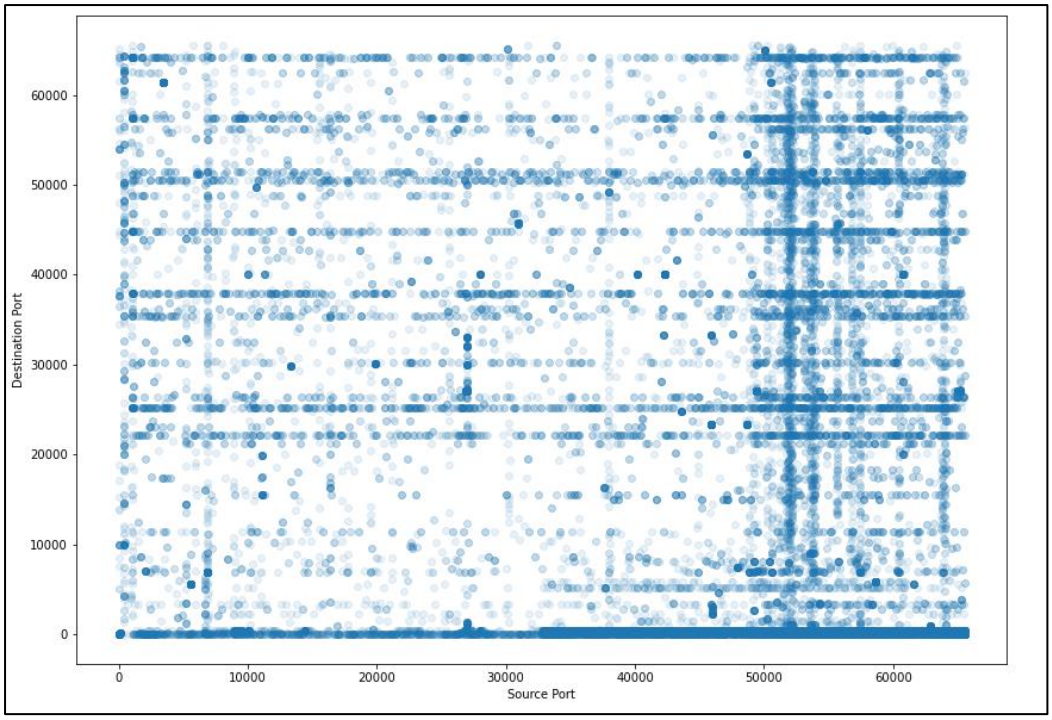


Fig 6.9: Scatter Plot of Source to Destination Packet Transfer

Fig 6.9 displays the scatter plot of the Source port - Destination port of each entry in the dataset. Packets move from Source Port to Destination Port and this has been visualized via this scatter plot.

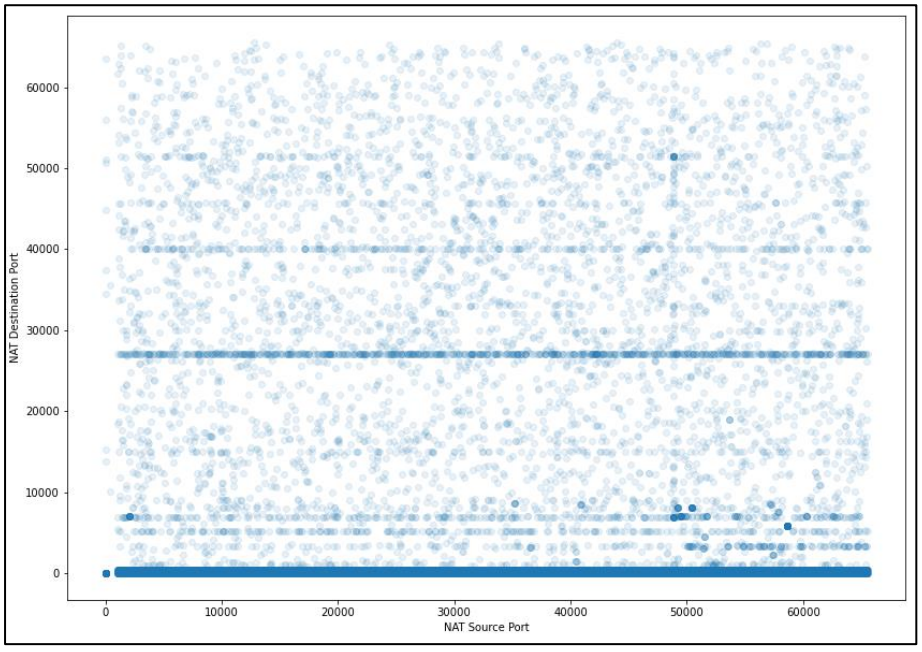


Fig 6.10: Scatter Plot of NAT Source Port to NAT Destination Port

Fig 6.10 displays the scatter plot of the NAT Source port – NAT Destination port of each entry in the dataset.

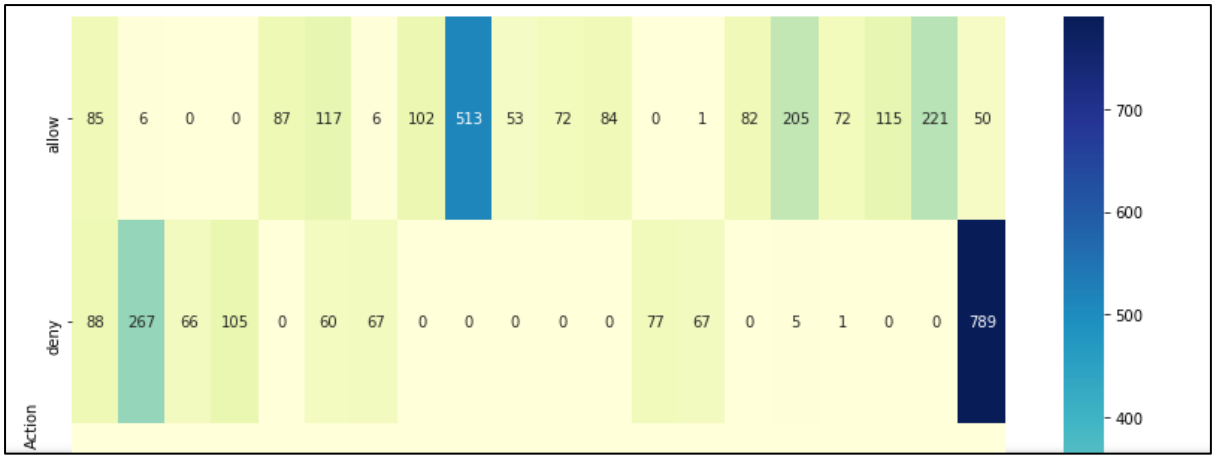


Fig 6.11: Source Port Top 20 Levels

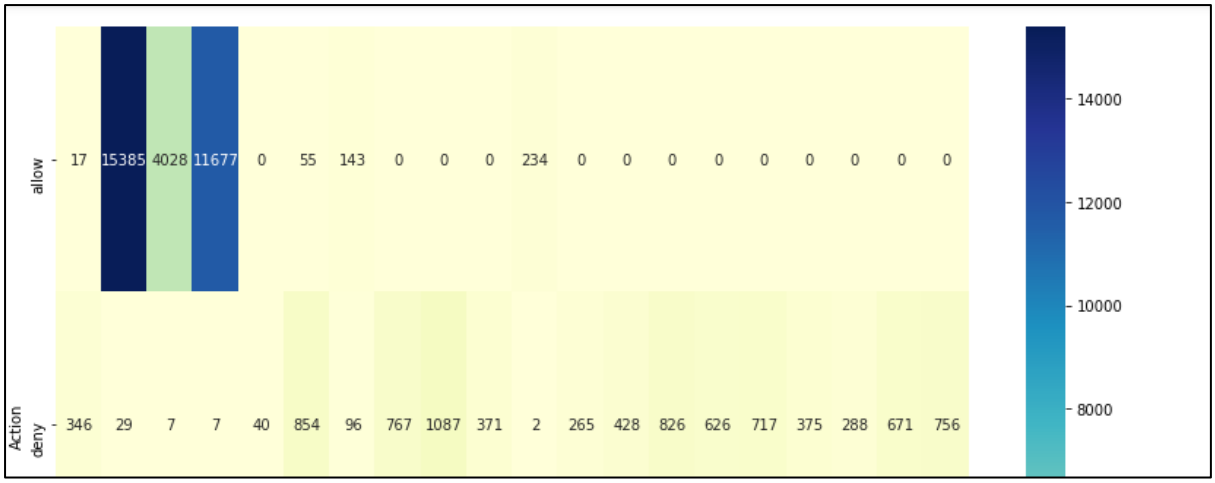


Fig 6.12: Destination Port Top 20 Levels

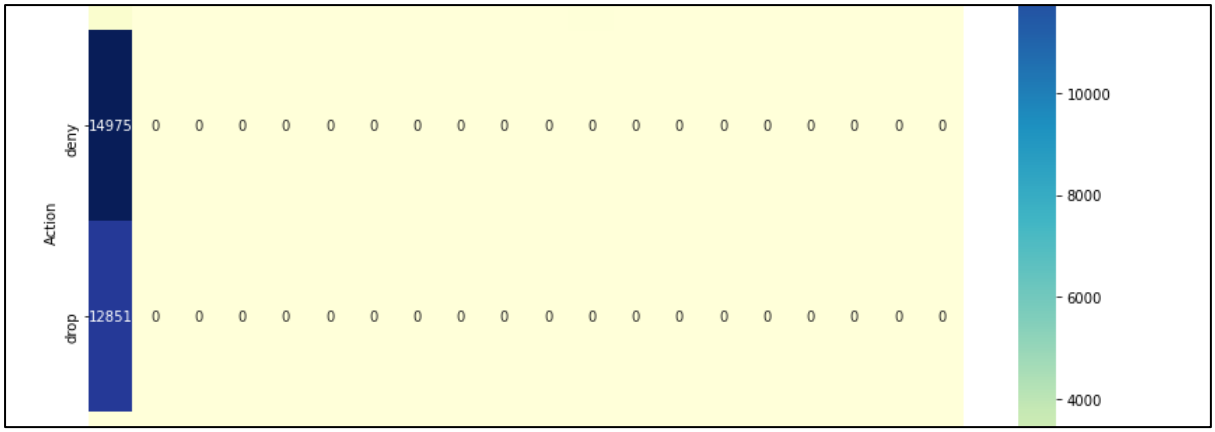


Fig 6.13: NAT Source Port Top 20 Levels

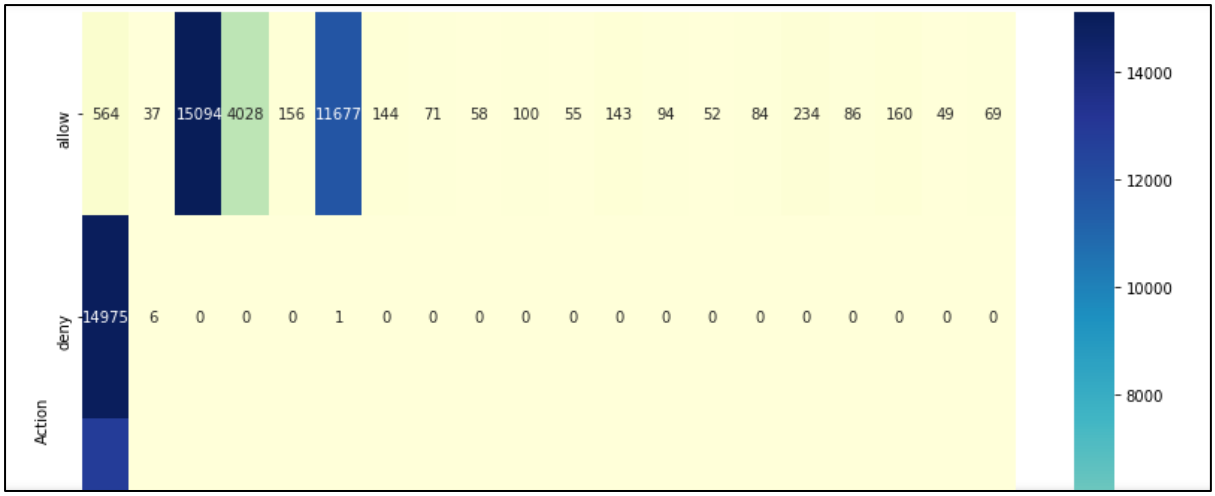


Fig 6.14: NAT Destination Port Top 20 levels

Fig 6.11 – Fig 6.14 shows the top 20 most used Source Ports, Destination Ports, NAT Source Port and NAT Destination ports segregated by the four actions (allow, deny, drop and reset-both).

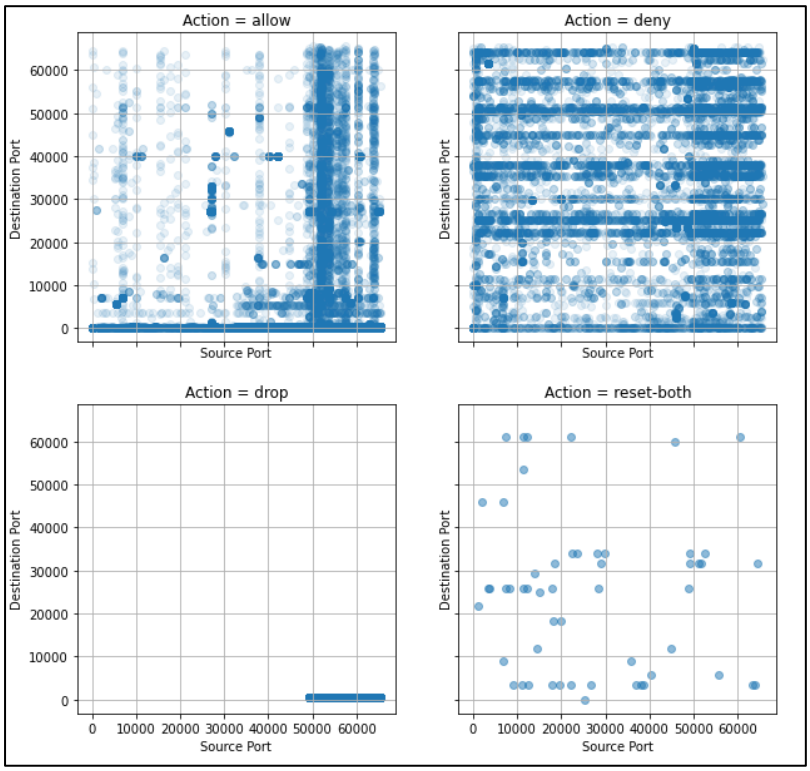


Fig 6.15: Source to Destination Port divided by Action

In Fig 6.15, the packet movement from Source Port to Destination Port has been segregated with respect to the 4 available actions and the subsequent scatter plot is provided.

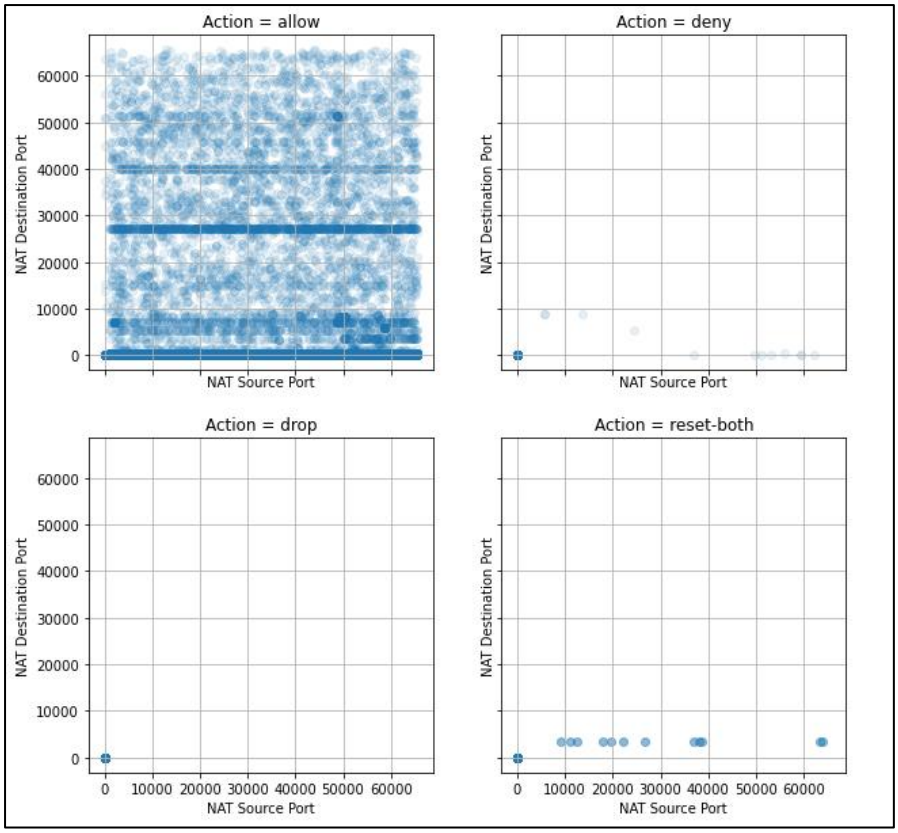


Fig 6.16: NAT Source to NAT Destination Port divided by Action

In Fig 6.16, the packet movement from NAT Source Port to NAT Destination Port has been segregated with respect to the 4 available actions and the subsequent scatter plot is provided.

Random Forest Model

```
In [22]: train, test=df_hex.split_frame(ratios=[0.75], seed=999)

In [23]: train[target].as_data_frame().value_counts()

Out[23]: Action
         allow         28186
         deny          11274
         drop           9645
         reset-both       38
         dtype: int64

In [24]: test[target].as_data_frame().value_counts()

Out[24]: Action
         allow         9454
         deny          3713
         drop          3206
         reset-both      16
         dtype: int64
```

Fig 6.17: Train-test split (75-25)

Cross-Validation Metrics Summary:

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	accuracy	0.998129	0.000448	0.998375	0.998674	0.997682	0.997649	0.998263
1	auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
2	err	0.001871	0.000448	0.001625	0.001326	0.002318	0.002351	0.001737
3	err_count	18.400000	4.449719	16.000000	13.000000	23.000000	23.000000	17.000000
4	logloss	0.044462	0.004302	0.046043	0.046080	0.040268	0.049999	0.039918
5	max_per_class_error	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000
6	mean_per_class_accuracy	0.748949	0.000330	0.749356	0.749211	0.748574	0.748707	0.748896
7	mean_per_class_error	0.251051	0.000330	0.250644	0.250789	0.251426	0.251293	0.251104
8	mse	0.006887	0.001044	0.007462	0.007576	0.005592	0.007872	0.005934
9	pr_auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
10	r2	0.989098	0.001637	0.988213	0.987742	0.991108	0.987793	0.990633
11	rmse	0.082792	0.006395	0.086384	0.087043	0.074780	0.088725	0.077031

Fig 6.18: Cross-Validation Metrics Summary

Fig 6.18 shows the Cross-Validation Metrics Summary which displays the mean values and the standard deviation as well as the values for each of the cross validations for metrics such as accuracy, logloss, mse, r^2, rmse and other such metrics. The Random Forest model was created to minimize the logloss error metric. From the above table we can see that the created model has a mean accracy of 99.81% with standard deviation of 0.000448 and mean logloss of 0.04444 with standard deviation of 0.004302. Logloss is a cross-entropy based metric that measures the quality of predictions as compared to accuracy. It is a gauge of additional error coming from the predictions as compared to the true values. Logloss value of 0 implies a perfect model. Other metrics such as Mean Squared Error (mse) have a mean value of 0.0068 and r^2 has a mean score of 0.989098 and rmse has mean score 0.08279.

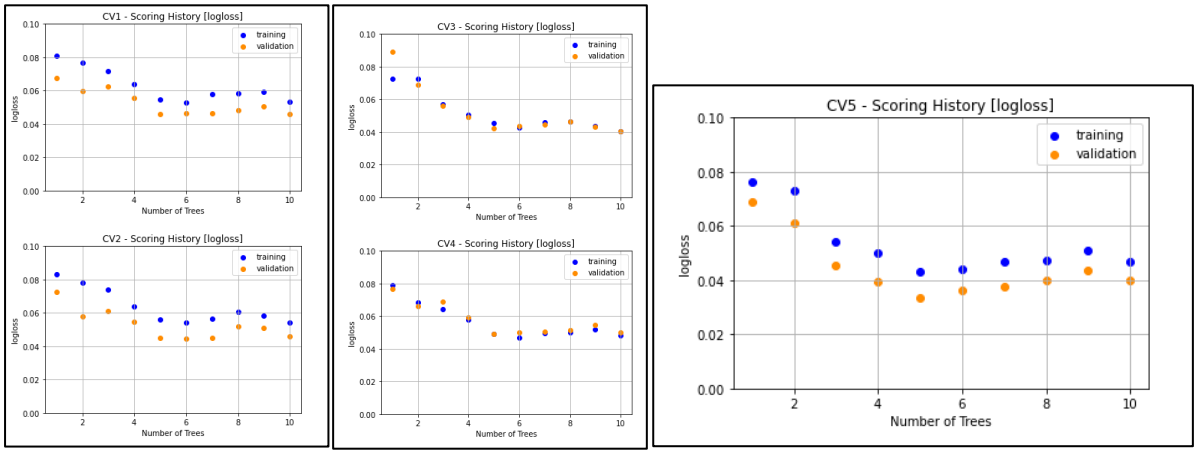


Fig 6.19: Logloss Scoring History per Cross Validation

Fig 6.19 shows the logloss values for the both training and testing datasets plotted against number of trees used in the random forest model (1 to 10). For a model to be effective it is important for the testing logloss to be lower than the training logloss (lower probability of error in predictions). It is observed that in general for most cross-validations, the logloss of testing set is lower than the logloss of training set.

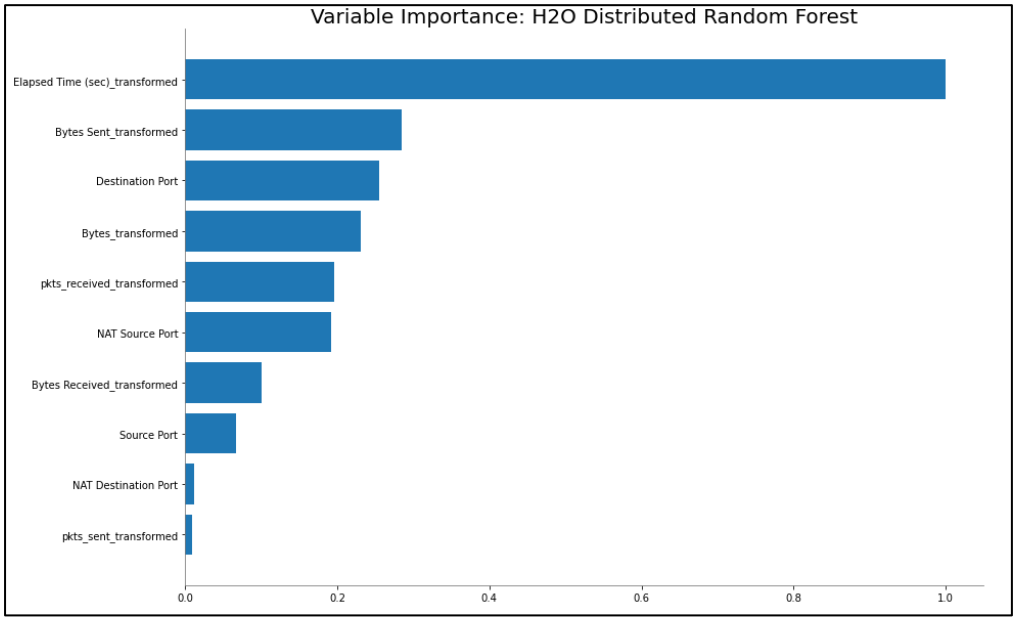


Fig 6.20: Variable Importance Plot

Fig 6.20 displays the Variable Importance Plot. It is observed that Elapsed Time (transformed) is the most important variable for the Random Forest Model followed by Bytes Sent (transformed) and then Destination Port followed by Bytes (transformed). The importance of the features are scored from 0 to 1 with 1 being the most important.

	predict	allow	deny	drop	reset-both	target
0	allow	0.999993	0.000007	0.0	0.0	allow
1	allow	0.999993	0.000007	0.0	0.0	allow
2	allow	0.999993	0.000007	0.0	0.0	allow
3	allow	0.999993	0.000007	0.0	0.0	allow
4	allow	0.999993	0.000007	0.0	0.0	allow

Fig 6.21: Training Set classification probability (per entry)

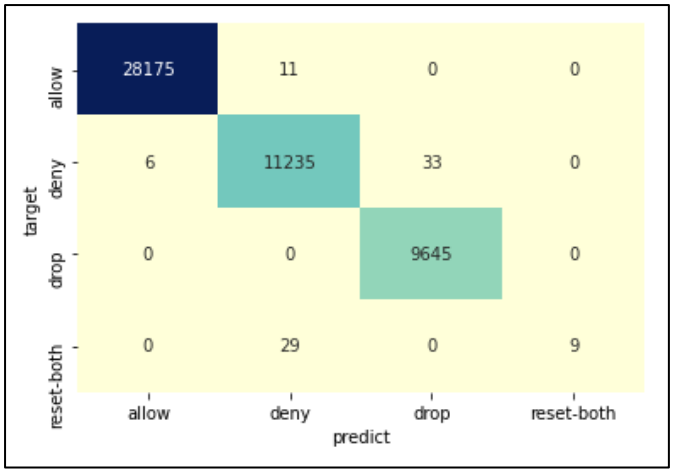


Fig 6.22: Training Set Performance – Confusion Matrix

Fig 6.21, 6.22 show the Training Set predictive probabilities and performance. The table (Fig 6.21) shows the probability of classification of each entry and its actual and predicted target labels. Fig 6.22 shows the confusion matrix for the training set data.

	predict	allow	deny	drop	reset-both	target
0	allow	0.999993	0.000007	0.0	0.0	allow
1	allow	0.999993	0.000007	0.0	0.0	allow
2	allow	0.999992	0.000008	0.0	0.0	allow
3	allow	0.999993	0.000007	0.0	0.0	allow
4	allow	0.999993	0.000007	0.0	0.0	allow

Fig 6.23: Training Set classification probability (per entry)

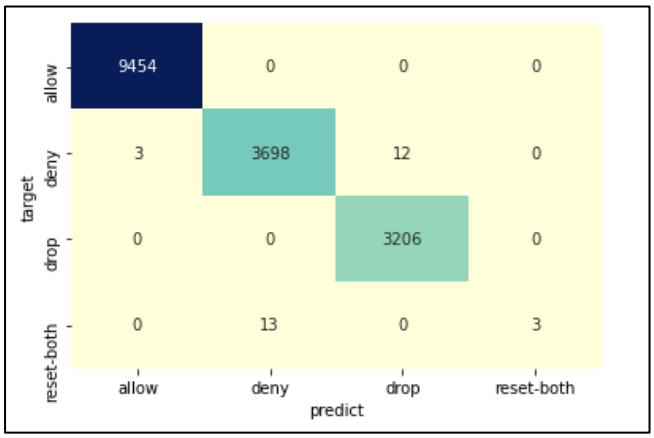


Fig 6.24: Testing Set Performance

Fig 6.23, 6.24 show the Testing Set predictive probabilities and performance. The table (Fig 6.23) shows the probability of classification of each entry and its actual and predicted target labels. Fig 6.24 shows the confusion matrix for the testing set data.

2. THREAT PREDICTION

	pattern	type
0	crteyreti=wsn&et3tf6shov=tdsviee y fum$oh3;ore&saib5hfavhepc=tcilb...	valid
1	*;q=0.7	valid
2	nhgiopie/7.8.4.7.0	valid
3	/dyylkl.xd9cpu/4ot0ta/ts6xnrp1/hssh/a2cuerht/se8j00jif@ubfx.77g/s0...	valid
4	/2m6vlb1r37jspc/cwvv/mbar/oqrd0/msc/etceebwgi/io/m2zxmv/r@i98tor3y...	valid
...	...	...
90248	<img onpointerenter=alert(1)>xss</img>	xss
90249	<source onbeforepaste="alert(1)" contenteditable>test</source>	xss
90250	<div draggable="true" contenteditable>drag me <head ondrop=al...	xss
90251	<cite id="citerefdomingos2015" class="citation book"><a href="...	valid
90252		valid

Fig 6.25: Combined Dataset

Fig 6.25 shows the complete merged and parsed dataset which is a combination of the XSS, HttpParameters and the ECML/PKDD dataset which contains the attack strings patterns of various types of attacks such as sqli, xss and other as well as valid strings.

	pattern	type
count	90253	90253
unique	81158	5
top	*	valid
freq	1285	60623

```
valid              60623
sqli               13153
xss                 9730
cmdi                3461
path-traversal      3286
Name: type, dtype: int64
```

Fig 6.26: Dataset description and Counts

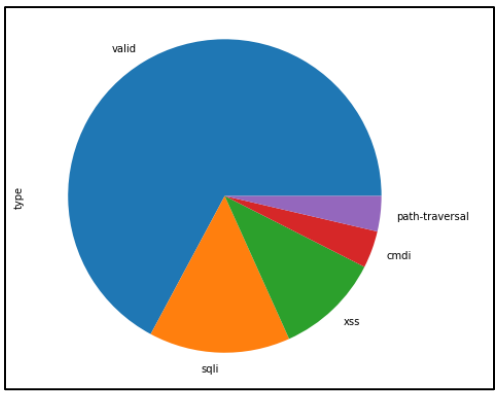


Fig 6.27: Pie Chart of Threat Distribution

Fig 6.26 shows the counts of each of the targets (valid, sqli, xss, cmdi and path-traversal). Fig 6.27 plots the pie-chart of the counts.

	precision	recall	f1-score	support
cmdi	0.99	0.98	0.98	865
path-traversal	1.00	0.97	0.98	822
sqli	1.00	0.99	0.99	3288
valid	1.00	1.00	1.00	15156
xss	1.00	1.00	1.00	2433
accuracy			1.00	22564
macro avg	0.99	0.99	0.99	22564
weighted avg	1.00	1.00	1.00	22564

Fig 6.28: Classification Report

Fig 6.28 is the classification report of the created ML pipeline which consist of TF-IDF vectorizer followed by SVM model (both linear and rbf kernels). The hyperparameters have been set in a parameter grid which is used in GridSearchCV to find the optimal hyperparameters for the created model. the classification report gives us the precision, recall and f1-score segregated by target. The accuracy of the created model is also 99.579% (from confusion matrix below) which means that the created model has perfect or almost perfect predictive power.

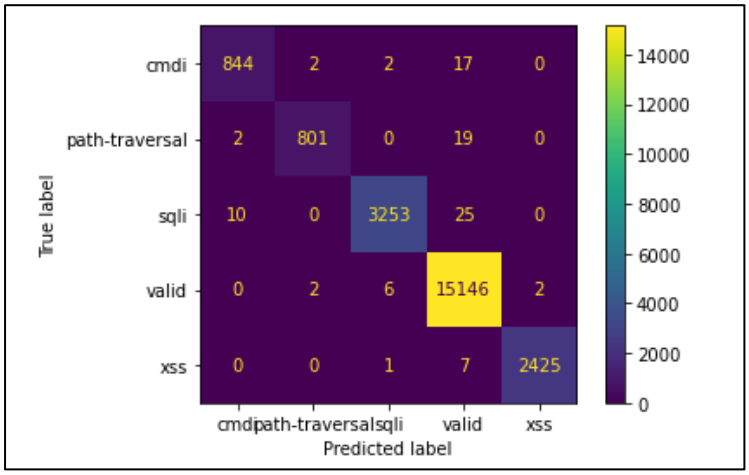


Fig 6.29: Confusion Matrix

Fig 6.29 displays the confusion matrix for the created model.

```
grid.best_params_
```

```
{'svc__C': 10, 'svc__kernel': 'rbf', 'tfidfvectorizer__ngram_range': (1, 2)}
```

Fig 6.30: Best Parameters from Grid Search

In Fig 6.30 we can see the most optimal hyperparameters from GridSearch Cross-Validation. The most optimal hyperparameters are a svc_C (penalty parameter) of 10 and the rbf svc kernel as well as ngram range of (1,2) for the TF-IDF vectorizer.

VII. ANALYSIS

1. Packet Classification

Cross-Validation Metrics Summary

Table 7.1: Cross-Validation Metrics Summary

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	accuracy	0.998129	0.000448	0.998375	0.998674	0.997682	0.997649	0.998263
1	auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
2	err	0.001871	0.000448	0.001625	0.001326	0.002318	0.002351	0.001737
3	err_count	18.400000	4.449719	16.000000	13.000000	23.000000	23.000000	17.000000
4	logloss	0.044462	0.004302	0.046043	0.046080	0.040268	0.049999	0.039918
5	max_per_class_error	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000
6	mean_per_class_accuracy	0.748949	0.000330	0.749356	0.749211	0.748574	0.748707	0.748896
7	mean_per_class_error	0.251051	0.000330	0.250644	0.250789	0.251426	0.251293	0.251104
8	mse	0.006887	0.001044	0.007462	0.007576	0.005592	0.007872	0.005934
9	pr_auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
10	r2	0.989098	0.001637	0.988213	0.987742	0.991108	0.987793	0.990633
11	rmse	0.082792	0.006395	0.086384	0.087043	0.074780	0.088725	0.077031

Cross-Validation Metrics Summary:

		mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
0	accuracy	0.998129	0.000448	0.998375	0.998674	0.997682	0.997649	0.998263
1	auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
2	err	0.001871	0.000448	0.001625	0.001326	0.002318	0.002351	0.001737
3	err_count	18.400000	4.449719	16.000000	13.000000	23.000000	23.000000	17.000000
4	logloss	0.044462	0.004302	0.046043	0.046080	0.040268	0.049999	0.039918
5	max_per_class_error	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000
6	mean_per_class_accuracy	0.748949	0.000330	0.749356	0.749211	0.748574	0.748707	0.748896
7	mean_per_class_error	0.251051	0.000330	0.250644	0.250789	0.251426	0.251293	0.251104
8	mse	0.006887	0.001044	0.007462	0.007576	0.005592	0.007872	0.005934
9	pr_auc	NaN	0.000000	NaN	NaN	NaN	NaN	NaN
10	r2	0.989098	0.001637	0.988213	0.987742	0.991108	0.987793	0.990633
11	rmse	0.082792	0.006395	0.086384	0.087043	0.074780	0.088725	0.077031

Fig 7.1: Cross-Validation Metrics Summary

Cross-Validation Metrics Summary displays the mean values and the standard deviation as well as the values for each of the cross validations for metrics such as accuracy, logloss, mse, r^2, rmse and other such metrics. The Random Forest model was created to minimize the logloss error metric. From the above table we can see that the created model has a mean accracy of 99.81% with standard deviation of 0.000448 and mean logloss of 0.04444 with standard deviation of 0.004302. Logloss is a cross-entropy based metric that measures the quality of predictions as compared to accuracy. It is a gauge of additional error coming from the predictions as compared to the true values. Logloss value of 0 implies a perfect model. Other metrics such as Mean Squared Error (mse) have a mean value of 0.0068 and r^2 has a mean score of 0.989098 and rmse has mean score 0.08279.

Logloss Scoring History

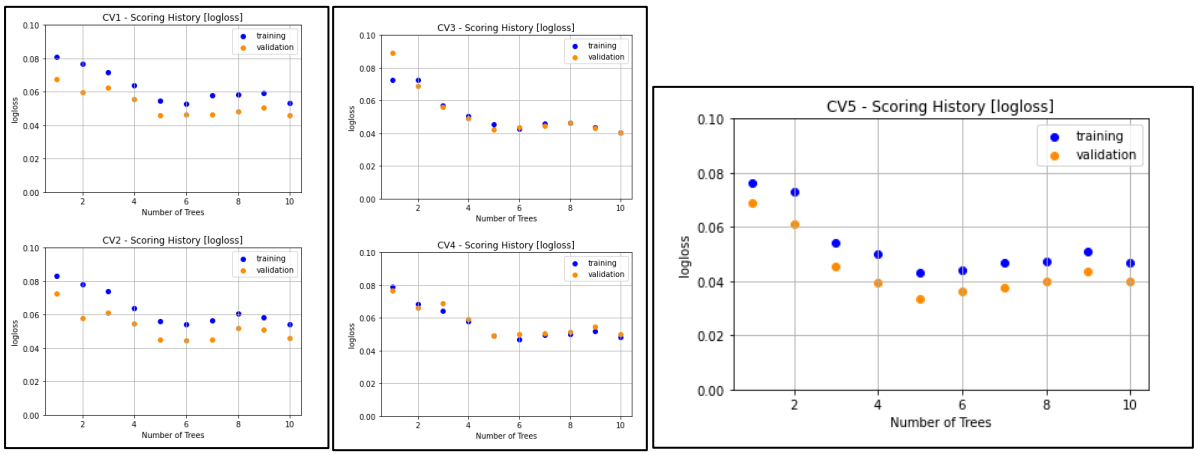


Fig 7.2: Logloss Scoring History per Cross Validation

Logloss values for the both training and testing datasets plotted against number of trees used in the random forest model (1 to 10). For a model to be effective it is important for the testing

logloss to be lower than the training logloss (lower probability of error in predictions). It is observed that in general for most cross-validations, the logloss of testing set is lower than the logloss of training set.

Variable Importance

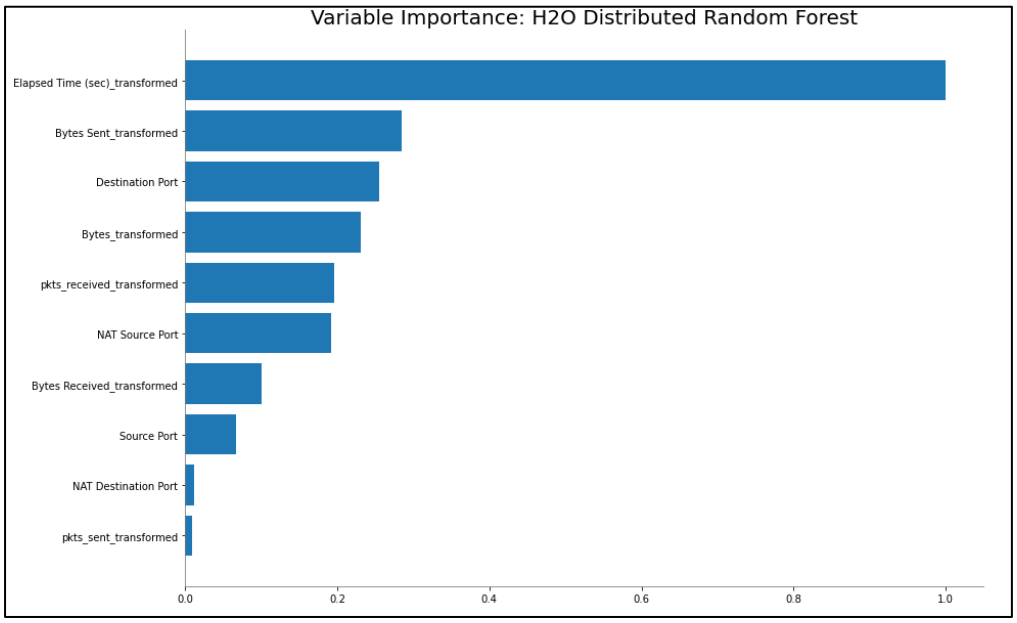


Fig 7.3: Variable Importance Plot

Variable Importance Plot scores the importance of the features for the model. It is observed that Elapsed Time (transformed) is the most important variable for the Random Forest Model followed by Bytes Sent (transformed) and then Destination Port followed by Bytes (transformed). The importance of the features are scored from 0 to 1 with 1 being the most important.

Training Set Performance

	predict	allow	deny	drop	reset-both	target
0	allow	0.999993	0.000007	0.0	0.0	allow
1	allow	0.999993	0.000007	0.0	0.0	allow
2	allow	0.999993	0.000007	0.0	0.0	allow
3	allow	0.999993	0.000007	0.0	0.0	allow
4	allow	0.999993	0.000007	0.0	0.0	allow

Fig 7.4: Training Set classification probability (per entry)

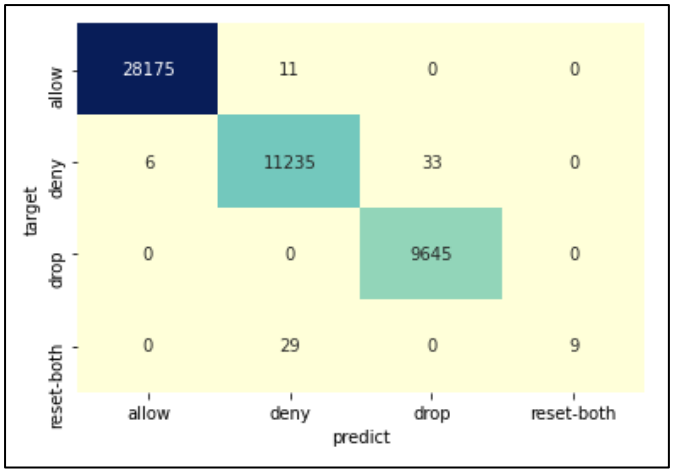


Fig 7.5: Training Set Performance – Confusion Matrix

Fig 7.4, 7.5 show the Training Set predictive probabilities and performance. The table (Fig 7.4) shows the probability of classification of each entry and its actual and predicted target labels. Fig 7.5 shows the confusion matrix for the training set data.

Testing Set Performance

	predict	allow	deny	drop	reset-both	target
0	allow	0.999993	0.000007	0.0	0.0	allow
1	allow	0.999993	0.000007	0.0	0.0	allow
2	allow	0.999992	0.000008	0.0	0.0	allow
3	allow	0.999993	0.000007	0.0	0.0	allow
4	allow	0.999993	0.000007	0.0	0.0	allow

Fig 7.6: Training Set classification probability (per entry)

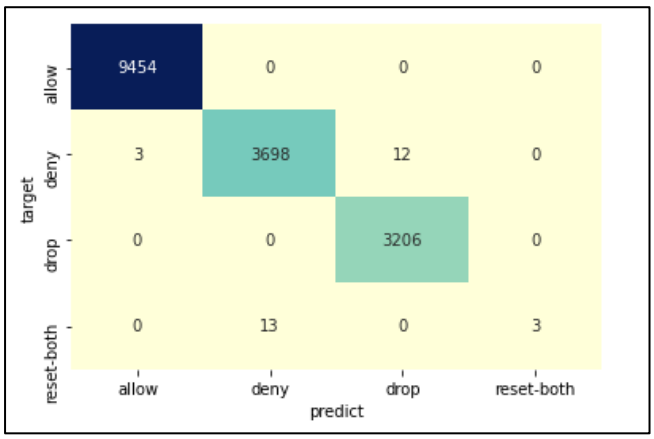


Fig 7.7: Testing Set Performance

Fig 7.6, 7.7 show the Testing Set predictive probabilities and performance. The table (Fig 7.6) shows the probability of classification of each entry and its actual and predicted target labels. Fig 7.7 shows the confusion matrix for the testing set data.

2. Threat Prediction

Classification Report

	precision	recall	f1-score	support
cmdi	0.99	0.98	0.98	865
path-traversal	1.00	0.97	0.98	822
sqli	1.00	0.99	0.99	3288
valid	1.00	1.00	1.00	15156
xss	1.00	1.00	1.00	2433
accuracy			1.00	22564
macro avg	0.99	0.99	0.99	22564
weighted avg	1.00	1.00	1.00	22564

Fig 7.8: Classification Report

Classification report of the created ML pipeline which consist of TF-IDF vectorizer followed by SVM model (both linear and rbf kernels). The hyperparameters have been set in a parameter grid which is used in GridSearchCV to find the optimal hyperparameters for the created model. the classification report gives us the precision, recall and f1-score segregated by target. The accuracy of the created model is also 1.00 which means that the created model has perfect or almost perfect predictive power.

Confusion Matrix

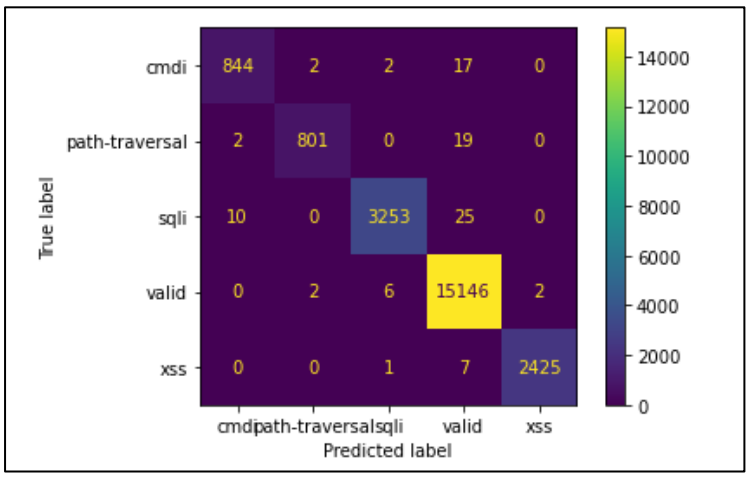


Fig 7.9: Confusion Matrix

Fig 7.9 displays the confusion matrix for the created model.

Optimal Hyperparameters

```
grid.best_params_
{'svc__C': 10, 'svc__kernel': 'rbf', 'tfidfvectorizer__ngram_range': (1, 2)}
```

Fig 7.10: Best Parameters from Grid Search

Sr No.	Hyperparameter	Optimal Value
1	svc_C	10
2	svc_kernel	rbf
3	tfidfvectorizer__ngram_range	(1, 2)

We can see the most optimal hyperparameters from GridSearch Cross-Validation. The most optimal hyperparameters are a svc_C (penalty parameter) of 10 and the rbf svc kernel as well as ngram range of (1,2) for the TF-IDF vectorizer.

VIII. CONCLUSION

Cybersecurity is an extremely important aspect of technology in today world. Securing data is of utmost importance for both users and big corporations alike. With the growing number and different types of attack that can occur nowadays, it is essential to have a system in place that is robust and can accurately predict whether any malicious activity is going on whether it be any bad packet transmission or any type of attack string or pattern (eg sql injection, xss etc). For this purpose, we have successfully developed a packet classification and threat detection system that is extremely accurate in predicting bad transmissions in packets and malicious activity in the form of attack pattern strings using a variety of machine learning techniques. For packet classification, Random Forest classifier has been implemented and a variety of performance metrics have been evaluated. The created Random Forest model has a mean accuracy of 99.81% and mean logloss of 0.04444 for 5 Cross Validations. Other metrics have also been evaluated such as Mean Squared Error (mse) with a mean value of 0.0068 and r^2 has a mean score of 0.989098 and rmse has mean score 0.08279. For threat detection, metrics such as accuracy, precision, recall and f1-score have been considered for the implemented ML pipeline consisting of TF-IDF and SVM. We have obtained an accuracy of 99.579% and the precision, recall and f1-score of the different targets has also been provided in the classification report.

For future work, we look towards trying different machine learning models to try to improve the training times required to fit the developed models. Another avenue of future work is to aim towards trying different natural language processing techniques for feature extraction form attack strings or pattern such as the bag of words technique.

IX. REFERENCES

[1] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, "Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection," *Big Data Min. Anal.*, vol. 4, no. 1, pp. 18–24, 2021, doi: 10.26599/BDMA.2020.9020019.

[2] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, "Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 1803–1816, 2021, doi: 10.1109/TNSM.2020.3014929.

[3] T. Wisanwanichthan and M. Thammawichai, "A Double-Layered Hybrid Approach for Network Intrusion Detection System Using Combined Naive Bayes and SVM," *IEEE Access*, vol. 9, pp. 138432–138450, 2021, doi: 10.1109/ACCESS.2021.3118573.

[4] G. Pu, L. Wang, J. Shen, and F. Dong, "A hybrid unsupervised clustering-based anomaly detection method," *Tsinghua Sci. Technol.*, vol. 26, no. 2, pp. 146–153, 2021, doi: 10.26599/TST.2019.9010051.

[5] L. Yan and J. Xiong, "Web-APT-Detect: A Framework For Web-Based Advanced Persistent Threat Detection Using Self-Translation Machine With Attention," *IEEE Lett. Comput. Soc.*, vol. 3, no. 2, pp. 66–69, 2020, doi: 10.1109/locs.2020.2998185.

[6] Z. Chkirbene, A. Erbad, R. Hamila, A. Mohamed, M. Guizani, and M. Hamdi, "TIDCS: A Dynamic Intrusion Detection and Classification System Based Feature Selection," *IEEE Access*, vol. 8, pp. 95864–95877, 2020, doi: 10.1109/ACCESS.2020.2994931.

[7] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep Learning Approach Combining Sparse Autoencoder with SVM for Network Intrusion Detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018, doi: 10.1109/ACCESS.2018.2869577.

[8] S. Naseer *et al.*, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48231–48246, 2018, doi: 10.1109/ACCESS.2018.2863036.

[9] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance Comparison of Support Vector Machine, Random Forest, and Extreme Learning Machine for Intrusion Detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018, doi: 10.1109/ACCESS.2018.2841987.

[10] D. Appelt, C. D. Nguyen, A. Panichella, and L. C. Briand, "A Machine-Learning-Driven Evolutionary Approach for Testing Web Application Firewalls," *IEEE Trans. Reliab.*, vol. 67, no. 3, pp. 733–757, 2018, doi: 10.1109/TR.2018.2805763.

[11] H. T. Nguyen, C. Torrano-Gimenez, G. Aalvarez, K. Franke, and S. Petrović, "Enhancing the effectiveness of web application firewalls by generic feature selection," *Log. J. IGPL*, vol. 21, no. 4, pp. 560–570, 2013, doi: 10.1093/jigpal/jzs033.

[12] K. Salah, K. Elbadawi, and R. Boutaba, "Performance modeling and analysis of network firewalls," *IEEE Trans. Netw. Serv. Manag.*, vol. 9, no. 1, pp. 12–21, 2012, doi: 10.1109/TNSM.2011.122011.110151.

[13] H. Hamed, A. El-Atawy, and E. Al-Shaer, "On dynamic optimization of packet matching in high-speed firewalls," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 10, pp. 1817–1830, 2006, doi: 10.1109/JSAC.2006.877140.

[14] Kumar, P., Kumar, & A. A., Sahayakingsly, & C., & Udayakumar, & A. (2083). *Analysis of intrusion detection in cyber attacks using DEEP learning neural networks*. https://doi.org/10.1007/s12083-020-00999-y/Published

[15] Kumar, S., Gupta, S., & Arora, S. (2021). Research Trends in Network-Based Intrusion Detection Systems: A Review. In *IEEE Access* (Vol. 9, pp. 157761–157779). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2021.3129775

[16] Thakkar, A., & Lohiya, R. (2022). A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review*, *55*(1), 453–563. https://doi.org/10.1007/s10462-021-10037-9

[17] Tian, Z., Luo, C., Qiu, J., Du, X., & Guizani, M. (2020). A Distributed Deep Learning System for Web Attack Detection on Edge Devices. *IEEE Transactions on Industrial Informatics*, *16*(3), 1963–1971. https://doi.org/10.1109/TII.2019.2938778

[18] Domingues Junior, M., & Ebecken, N. F. F. (2021). A new WAF architecture with machine learning for resource-efficient use. *Computers and Security*, *106*. https://doi.org/10.1016/j.cose.2021.102290

[19] Ucar, E., & Ozhan, E. (2017). The Analysis of Firewall Policy Through Machine Learning and Data Mining. *Wireless Personal Communications*, *96*(2), 2891–2909. https://doi.org/10.1007/s11277-017-4330-0

[20] Saba, T., Sadad, T., Rehman, A., Mehmood, Z., & Javaid, Q. (2021). Intrusion Detection System through Advance Machine Learning for the Internet of Things Networks. *IT Professional*, *23*(2), 58–64. https://doi.org/10.1109/MITP.2020.2992710

[21] Shekhawat, A. S., Troia, F. di, & Stamp, M. (2019). Feature analysis of encrypted malicious traffic. *Expert Systems with Applications*, *125*, 130–141. https://doi.org/10.1016/j.eswa.2019.01.064

[22] Marques, C., Malta, S., & Magalhães, J. (2021). DNS firewall based on machine learning. *Future Internet*, *13*(12). https://doi.org/10.3390/fi13120309

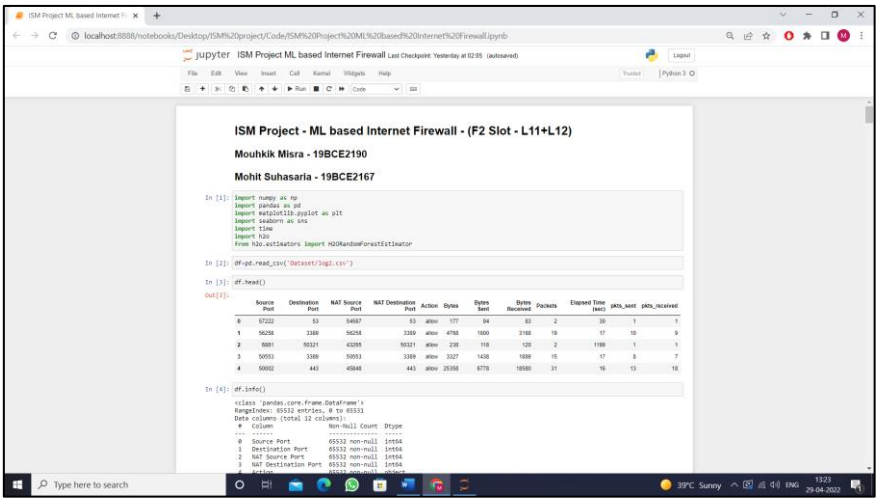
[23] Arunkumar, M., & Ashok Kumar, K. (2022). Malicious attack detection approach in cloud computing using machine learning techniques. *Soft Computing*. https://doi.org/10.1007/s00500-021-06679-0

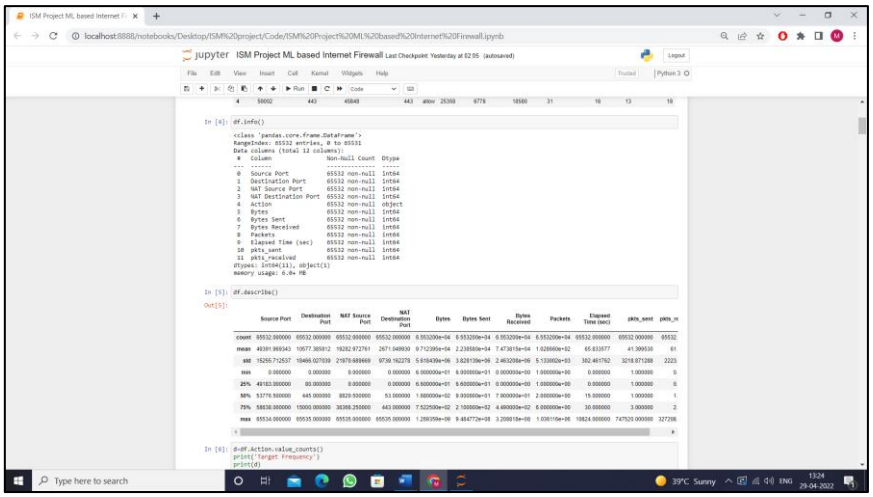
[24] Mahfouz, A., Abuhussein, A., Venugopal, D., & Shiva, S. (2020). Ensemble classifiers for network intrusion detection using a novel network attack dataset. *Future Internet*, *12*(11), 1–19. https://doi.org/10.3390/fi12110180

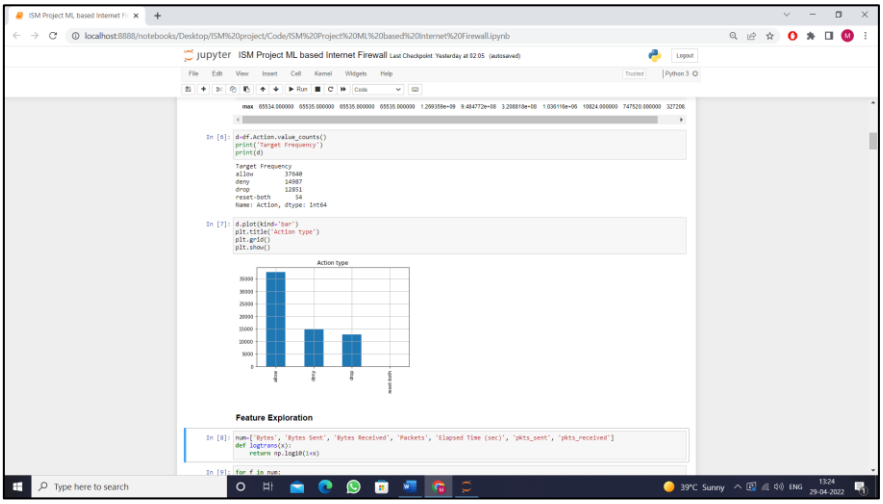
[25] Rezaei, S., & Liu, X. (2019). Deep Learning for Encrypted Traffic Classification: An Overview. In *IEEE Communications Magazine* (Vol. 57, Issue 5, pp. 76–81). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/MCOM.2019.1800819

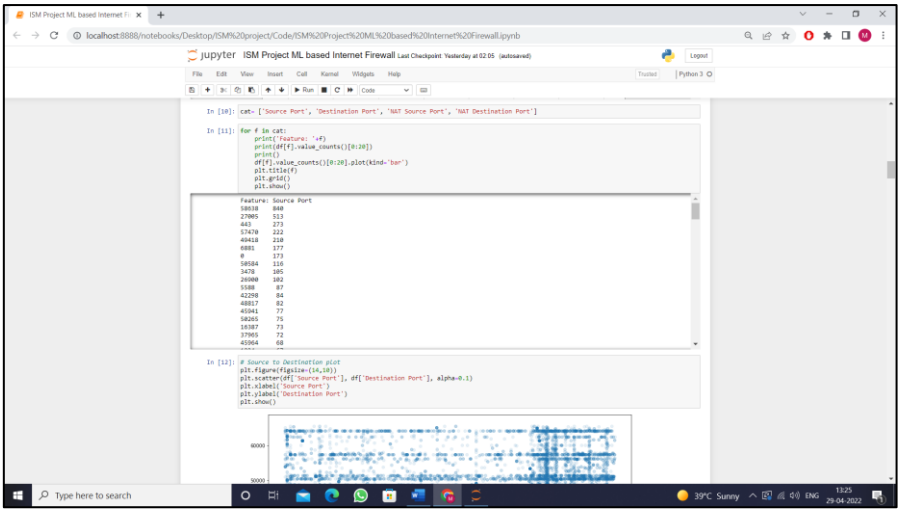
APPENDIX

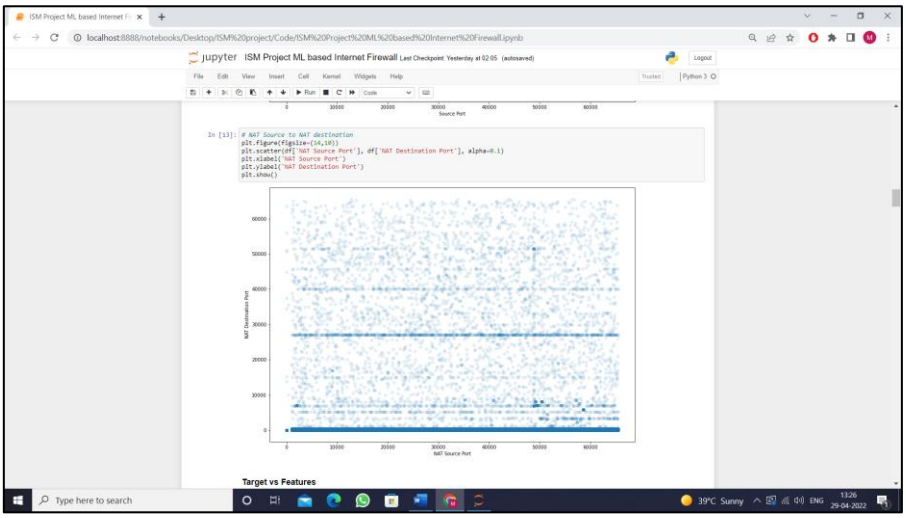
Code Screenshots

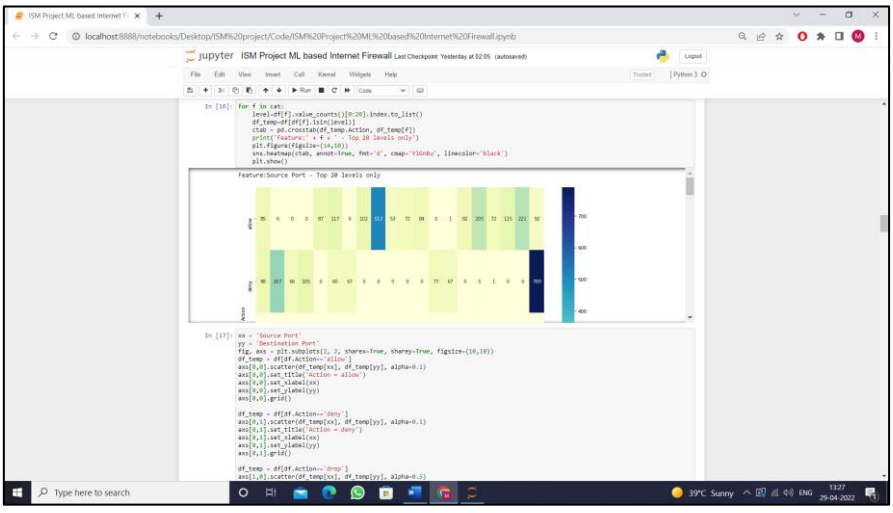


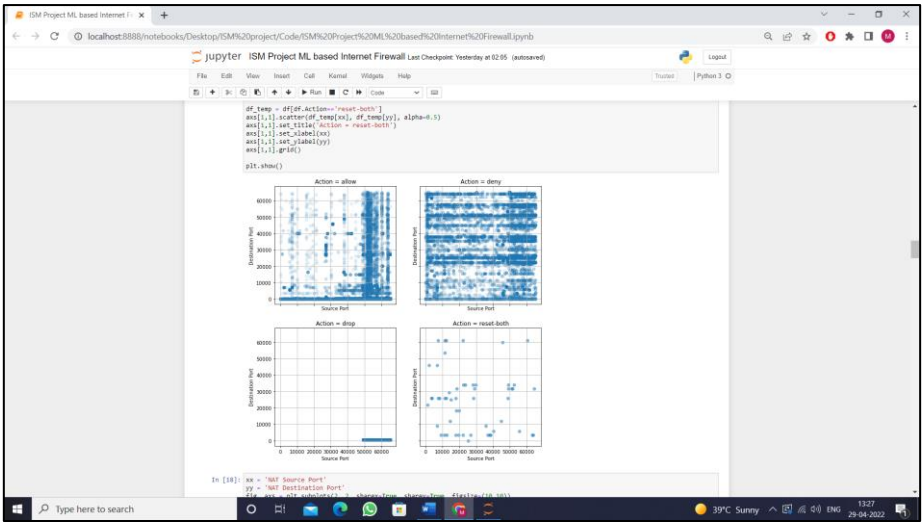


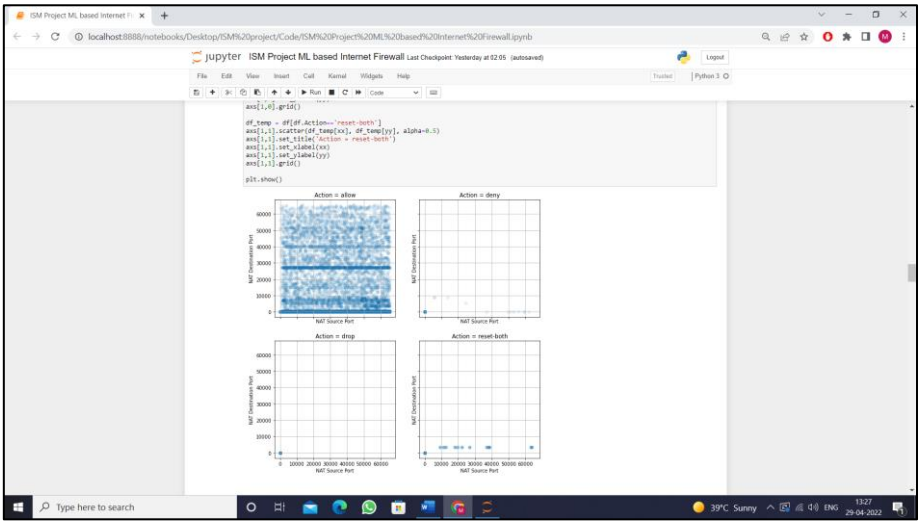


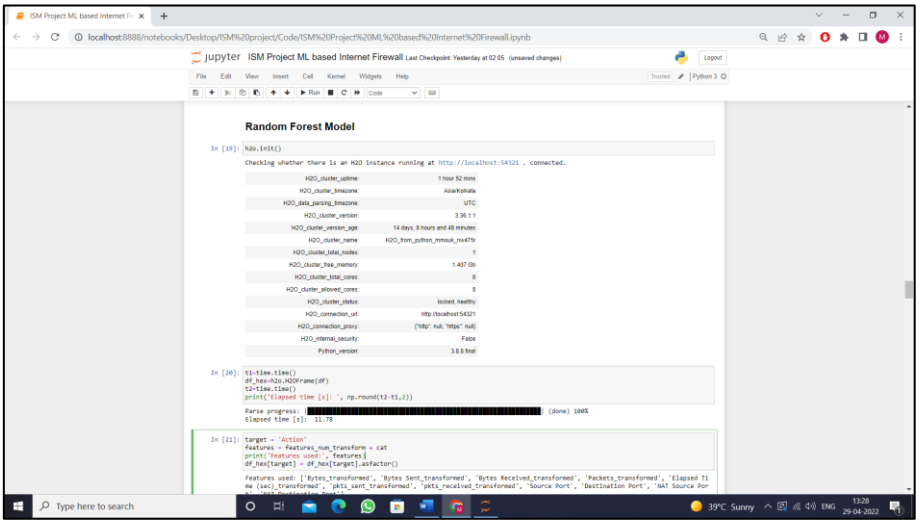

ISM Project ML based Internet Firewall
In [10]: cat= ['Source Port', 'Destination Port', 'NAT Source Port', 'NAT Destination Port']
In [11]: for f in cat:
Feature: Source Port
In [12]: # Source to Destination plot
plt.figure(figsize=(14,10))
plt.scatter(df['Source Port'], df['Destination Port'], alpha=0.1)
plt.xlabel('Source Port')
plt.ylabel('Destination Port')
plt.show()

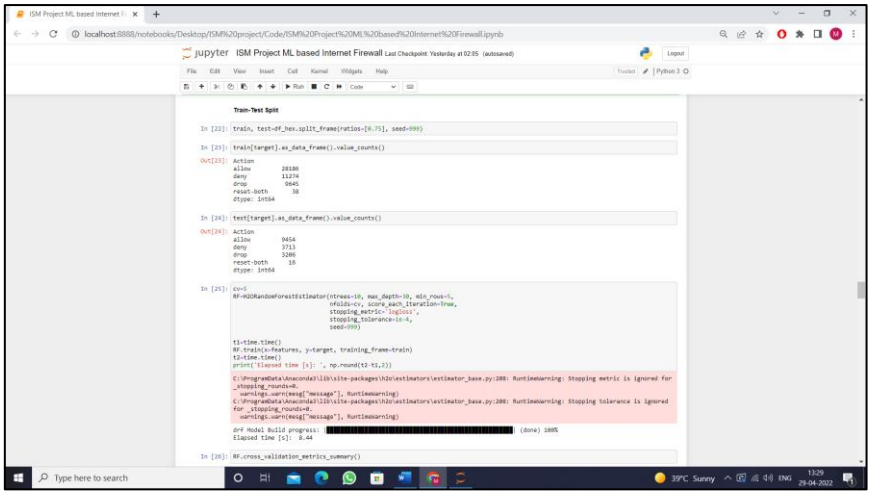

ISM Project ML based Internet Firewall
In [13]: # NAT Source to NAT destination
plt.figure(figsize=(14,10))
plt.scatter(df['NAT Source Port'], df['NAT Destination Port'], alpha=0.1)
plt.xlabel('NAT Source Port')
plt.ylabel('NAT Destination Port')
plt.show()
NAT Destination Port
NAT Source Port
Target vs Features

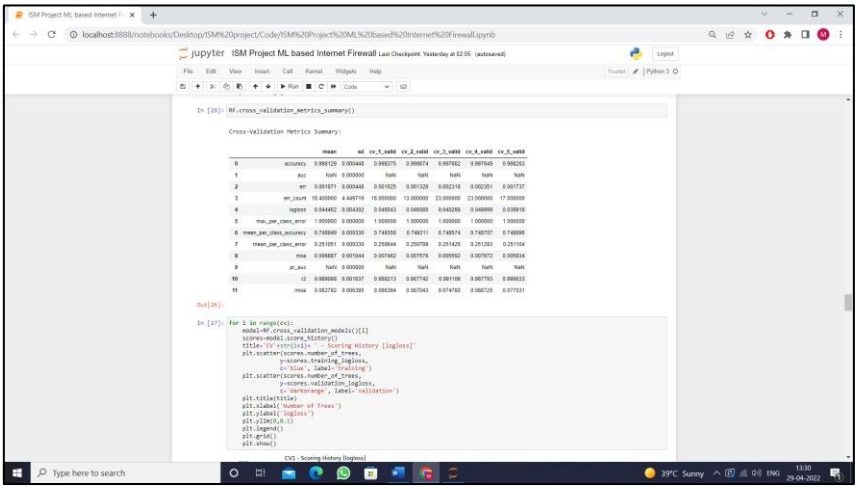

ISM Project ML based Internet Firewall
In [16]: for f in cat:
Feature:Source Port - Top 20 levels only
In [17]: xx = 'Source Port'
yy = 'Destination Port'
fig, axs = plt.subplots(2, 2, sharex=True, sharey=True, figsize=(10,10))
df_temp = df[df.Action=='allow']
axs[0,0].scatter(df_temp[xx], df_temp[yy], alpha=0.1)
axs[0,0].set_title('Action = allow')
axs[0,0].set_xlabel(xx)
axs[0,0].set_ylabel(yy)
axs[0,0].grid()
df_temp = df[df.Action=='deny']
axs[0,1].scatter(df_temp[xx], df_temp[yy], alpha=0.1)
axs[0,1].set_title('Action = deny')
axs[0,1].set_xlabel(xx)
axs[0,1].set_ylabel(yy)
axs[0,1].grid()
df_temp = df[df.Action=='drop']

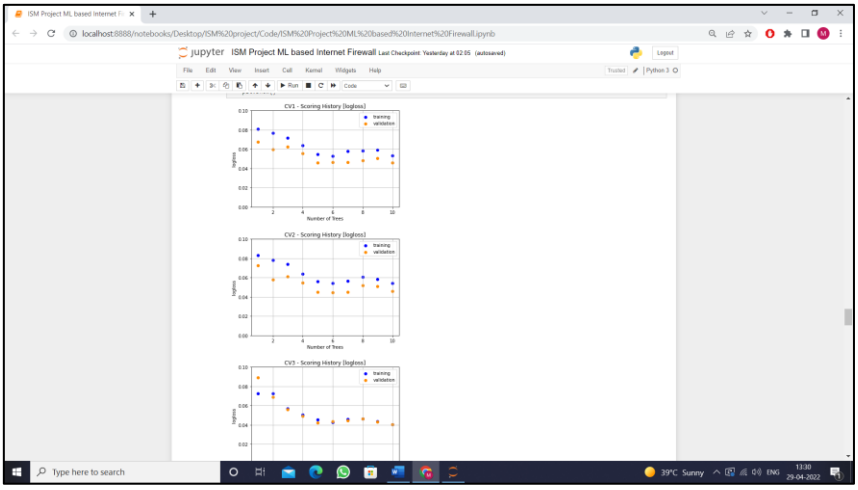

ISM Project ML based Internet Firewall
Action = allow
Action = deny
Action = drop
Action = reset-both
Source Port
Destination Port

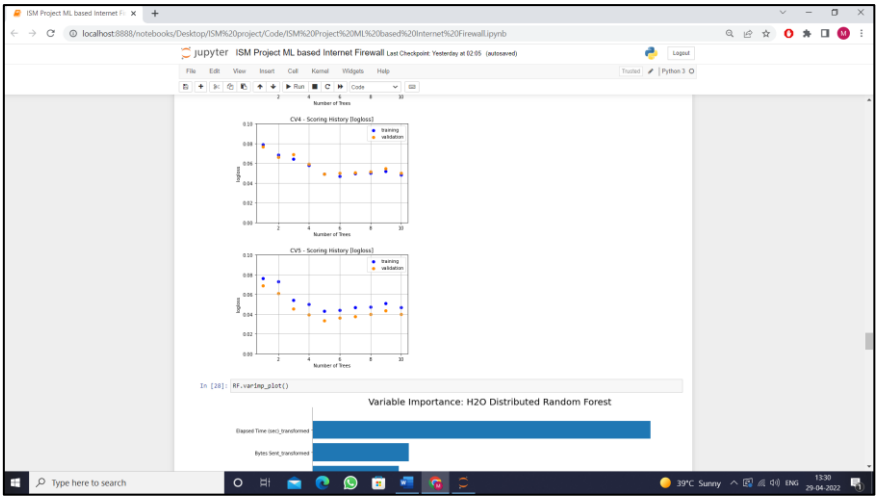

ISM Project ML based Internet Firewall
Action = allow
Action = deny
Action = drop
Action = reset-both
NAT Source Port
NAT Destination Port

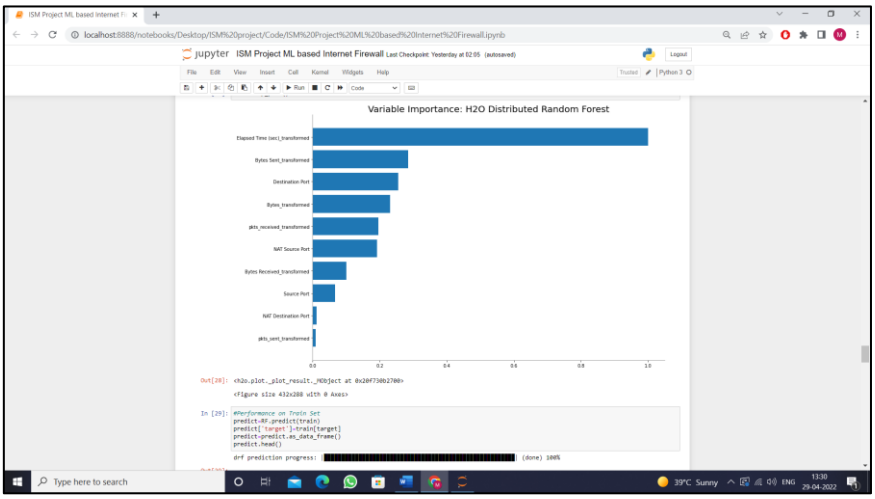

ISM Project ML based Internet Firewall
Random Forest Model
h2o.init()
Checking whether there is an H2O instance running at http://localhost:54321 . connected.
Parse progress: (done) 100%
Elapsed time [s]: 11.78

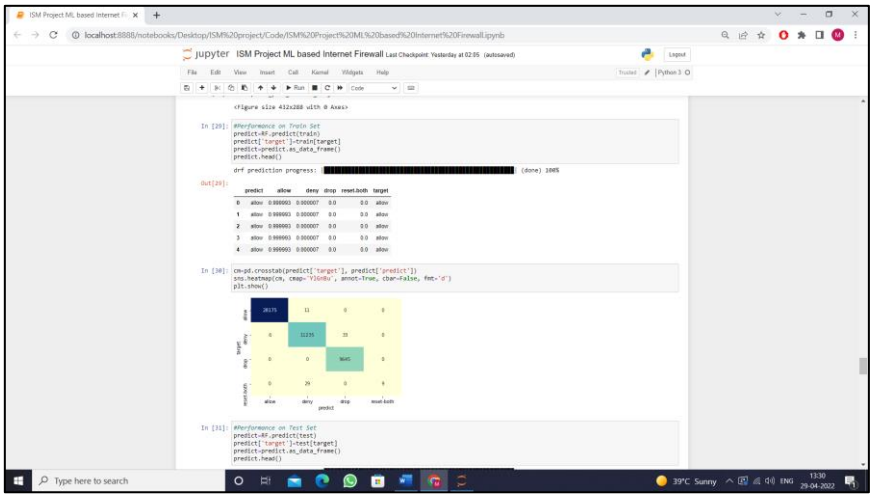

ISM Project ML based Internet Firewall
Train-Test Split
In [22]: train, test=df_hex.split_frame(ratios=[0.75], seed=999)
In [23]: train[target].as_data_frame().value_counts()
Out[23]: Action
allow 28186
deny 11274
drop 9045
reset-both 38
dtype: int64
In [24]: test[target].as_data_frame().value_counts()
Out[24]: Action
allow 9454
deny 3713
drop 3286
reset-both 16
dtype: int64
In [25]: cv=5
RF.train(x,y,train_frame=train)
Elapsed time [s]: 8.44
In [26]: RF.cross_validation_metrics_summary()

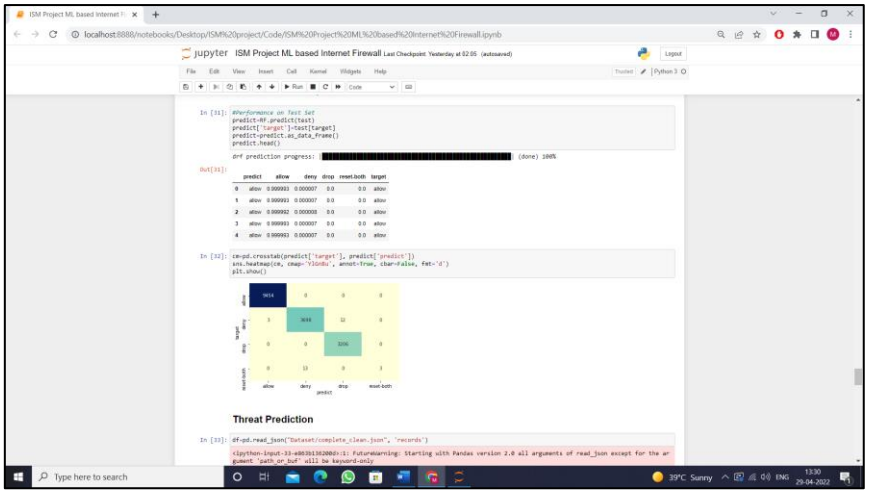

ISM Project ML based Internet Firewall
In [26]: RF.cross_validation_metrics_summary()
Cross-Validation Metrics Summary:
In [27]: for i in range(cv):
model=RF.cross_validation_models()[i]
scores=model.scoring_history()
title='CV '+str(i+1)+ ' - Scoring History [logloss]'
plt.scatter(scores.number_of_trees,
y=scores.training_logloss,
c='blue', label='training')
plt.scatter(scores.number_of_trees,
y=scores.validation_logloss,
c='darkorange', label='validation')
plt.title(title)
plt.xlabel('Number of Trees')
plt.ylabel('logloss')
plt.ylim(0,0.1)
plt.legend()
plt.grid()
plt.show()

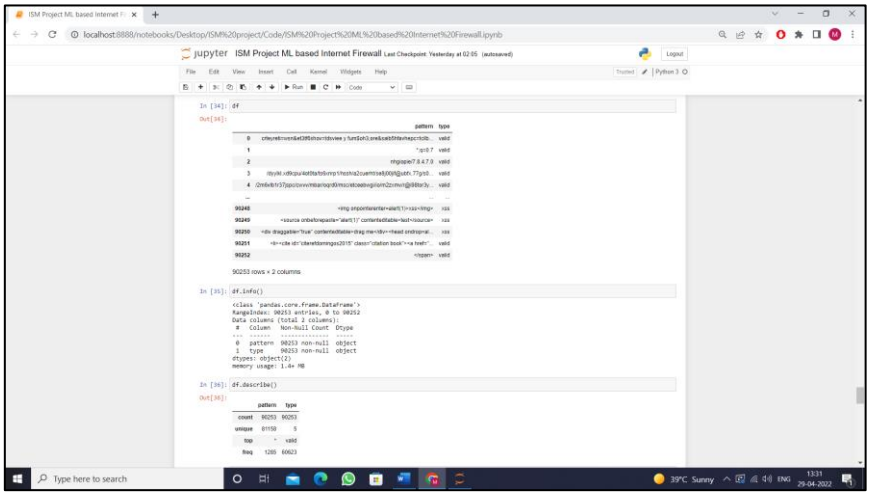

ISM Project ML based Internet Firewall
CV1 - Scoring History [logloss]
CV2 - Scoring History [logloss]
CV3 - Scoring History [logloss]
training
validation
logloss
Number of Trees

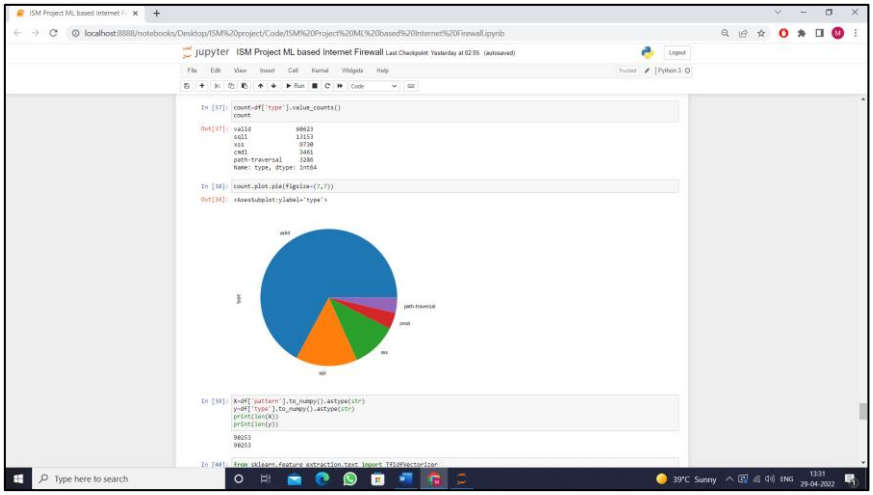

ISM Project ML based Internet Firewall
CV4 - Scoring History (logloss)
CV5 - Scoring History (logloss)
Variable Importance: H2O Distributed Random Forest

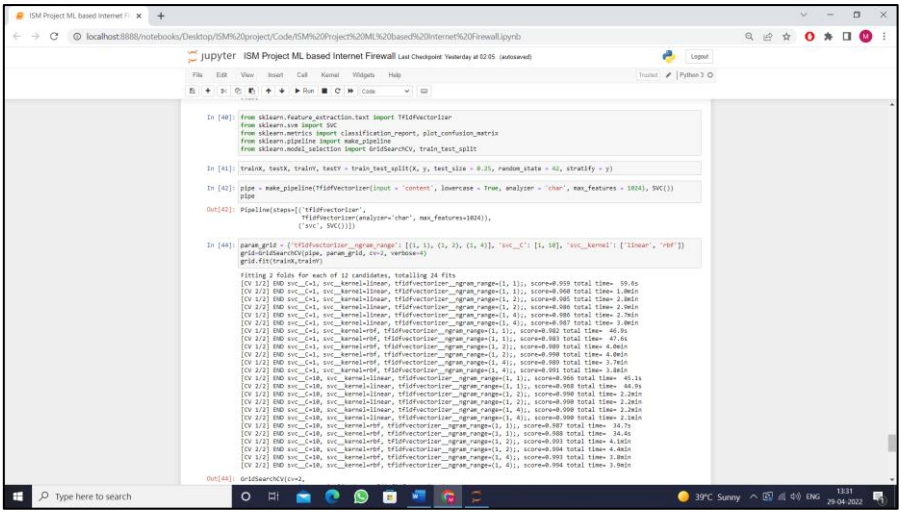

ISM Project ML based Internet Firewall
Variable Importance: H2O Distributed Random Forest

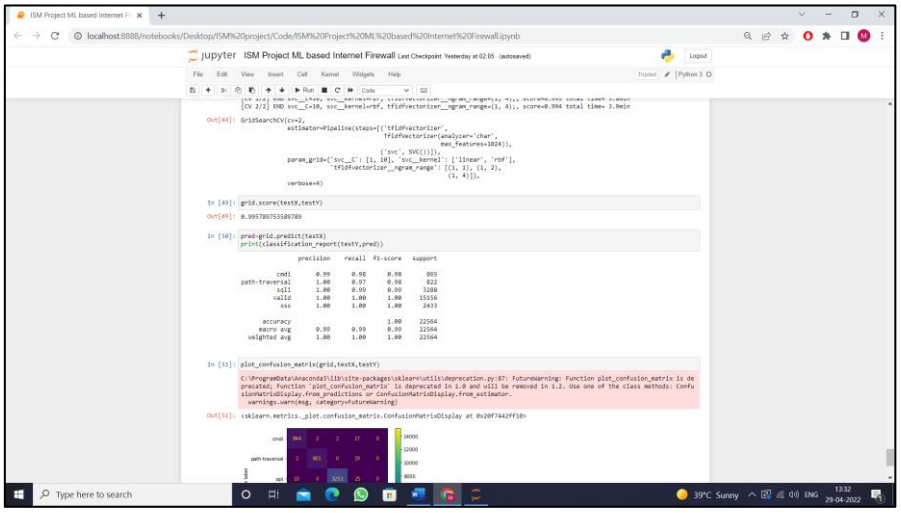

ISM Project ML based Internet Firewall

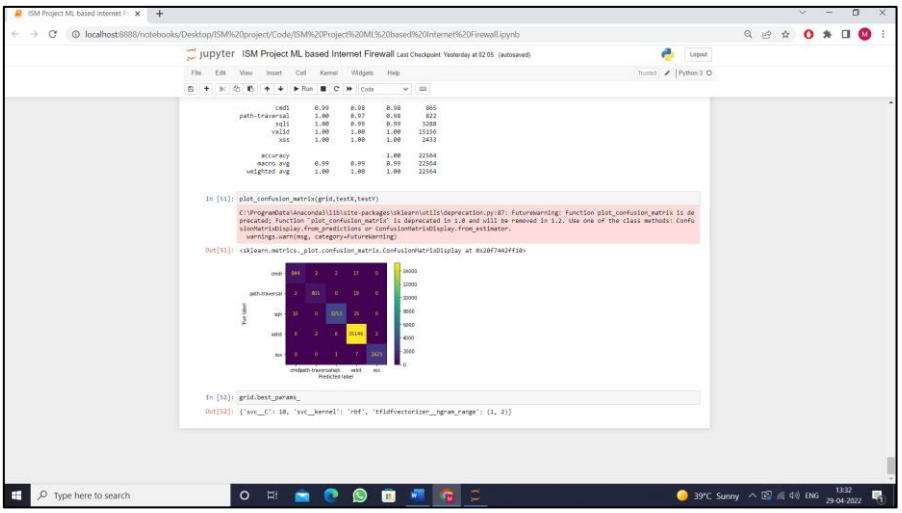

ISM Project ML based Internet Firewall
Threat Prediction


ISM Project ML based Internet Firewall


ISM Project ML based Internet Firewall


ISM Project ML based Internet Firewall
localhost:8888/notebooks/Desktop/ISM%20project/Code/ISM%20Project%20ML%20based%20Internet%20Firewall.ipynb
jupyter ISM Project ML based Internet Firewall
In [40]:
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import classification_report, plot_confusion_matrix
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV, train_test_split
In [41]:
trainX, testX, trainY, testY = train_test_split(X, y, test_size = 0.25, random_state = 42, stratify = y)
In [42]:
pipe = make_pipeline(TfidfVectorizer(input = 'content', lowercase = True, analyzer = 'char', max_features = 1024), SVC())
pipe
Out[42]:
Pipeline(steps=[('tfidfvectorizer',
TfidfVectorizer(analyzer='char', max_features=1024)),
('svc', SVC())])
In [44]:
param_grid = {'tfidfvectorizer__ngram_range': [(1, 1), (1, 2), (1, 4)], 'svc__C': [1, 10], 'svc__kernel': ['linear', 'rbf']}
grid=GridSearchCV(pipe, param_grid, cv=2, verbose=4)
grid.fit(trainX,trainY)
Fitting 2 folds for each of 12 candidates, totalling 24 fits
[CV 1/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 1);, score=0.959 total time= 59.6s
[CV 2/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 1);, score=0.960 total time= 1.0min
[CV 1/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 2);, score=0.985 total time= 2.0min
[CV 2/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 2);, score=0.986 total time= 2.0min
[CV 1/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 4);, score=0.986 total time= 2.7min
[CV 2/2] END svc__C=1, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 4);, score=0.987 total time= 3.0min
[CV 1/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 1);, score=0.982 total time= 46.0s
[CV 2/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 1);, score=0.983 total time= 47.6s
[CV 1/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 2);, score=0.989 total time= 4.0min
[CV 2/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 2);, score=0.990 total time= 4.0min
[CV 1/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 4);, score=0.989 total time= 3.7min
[CV 2/2] END svc__C=1, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 4);, score=0.991 total time= 3.8min
[CV 1/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 1);, score=0.966 total time= 45.1s
[CV 2/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 1);, score=0.968 total time= 44.9s
[CV 1/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 2);, score=0.990 total time= 2.2min
[CV 2/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 2);, score=0.990 total time= 2.2min
[CV 1/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 4);, score=0.990 total time= 2.2min
[CV 2/2] END svc__C=10, svc__kernel=linear, tfidfvectorizer__ngram_range=(1, 4);, score=0.990 total time= 2.1min
[CV 1/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 1);, score=0.987 total time= 34.7s
[CV 2/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 1);, score=0.988 total time= 34.6s
[CV 1/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 2);, score=0.993 total time= 4.1min
[CV 2/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 2);, score=0.994 total time= 4.4min
[CV 1/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 4);, score=0.993 total time= 3.8min
[CV 2/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 4);, score=0.994 total time= 3.9min
Out[44]:
GridSearchCV(cv=2,
Type here to search
39°C Sunny
ENG
13:31
29-04-2022


ISM Project ML based Internet Firewall
localhost:8888/notebooks/Desktop/ISM%20project/Code/ISM%20Project%20ML%20based%20Internet%20Firewall.ipynb
jupyter ISM Project ML based Internet Firewall
[CV 2/2] END svc__C=10, svc__kernel=rbf, tfidfvectorizer__ngram_range=(1, 4);, score=0.994 total time= 3.9min
Out[44]:
GridSearchCV(cv=2,
estimator=Pipeline(steps=[('tfidfvectorizer',
TfidfVectorizer(analyzer='char',
max_features=1024)),
('svc', SVC())]),
param_grid={'svc__C': [1, 10], 'svc__kernel': ['linear', 'rbf'],
'tfidfvectorizer__ngram_range': [(1, 1), (1, 2),
(1, 4)]},
verbose=4)
In [49]:
grid.score(testX,testY)
Out[49]:
0.9957897555d9789
In [50]:
pred=grid.predict(testX)
print(classification_report(testY,pred))
precision recall f1-score support
cmdi 0.99 0.98 0.98 865
path-traversal 1.00 0.97 0.98 822
sqli 1.00 0.99 0.99 3288
valid 1.00 1.00 1.00 15156
xss 1.00 1.00 1.00 2433
accuracy 1.00 22564
macro avg 0.99 0.99 0.99 22564
weighted avg 1.00 1.00 1.00 22564
In [51]:
plot_confusion_matrix(grid,testX,testY)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
warnings.warn(msg, category=FutureWarning)
Out[51]:
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20f7442ff10>
cmdi
path-traversal
sqli
valid
xss
True label
16000
12000
10000
8000
Type here to search
39°C Sunny
ENG
13:32
29-04-2022


ISM Project ML based Internet Firewall
localhost:8888/notebooks/Desktop/ISM%20project/Code/ISM%20Project%20ML%20based%20Internet%20Firewall.ipynb
jupyter ISM Project ML based Internet Firewall
cmdi 0.99 0.98 0.98 865
path-traversal 1.00 0.97 0.98 822
sqli 1.00 0.99 0.99 3288
valid 1.00 1.00 1.00 15156
xss 1.00 1.00 1.00 2433
accuracy 1.00 22564
macro avg 0.99 0.99 0.99 22564
weighted avg 1.00 1.00 1.00 22564
In [51]:
plot_confusion_matrix(grid,testX,testY)
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecated in 1.0 and will be removed in 1.2. Use one of the class methods: ConfusionMatrixDisplay.from_predictions or ConfusionMatrixDisplay.from_estimator.
warnings.warn(msg, category=FutureWarning)
Out[51]:
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x20f7442ff10>
cmdi
path-traversal
sqli
valid
xss
True label
cmdi
path-traversal
sqli
valid
xss
Predicted label
16000
12000
10000
8000
6000
4000
2000
In [52]:
grid.best_params_
Out[52]:
{'svc__C': 10, 'svc__kernel': 'rbf', 'tfidfvectorizer__ngram_range': (1, 2)}
Type here to search
39°C Sunny
ENG
13:32
29-04-2022