**DATA-DRIVEN DEFENSE: MACHINE LEARNING PARADIGM FOR NETWORK INTRUSION DETECTION**

**MAJOR PROJECT REPORT**

Submitted to the Department of Computer Applications, Bharathiar University in partial fulfilment of the requirements for the award of the degree of

**MASTER OF SCIENCE IN DATA ANALYTICS**

Submitted by

**P R YASHWANTH RAJAN (22CSEG34)**

Under the guidance of

**Mr. K. MOORTHY, MCA.,**

Department of Computer Applications

|  |
| --- |
|  |
|  |  |



**DEPARTMENT OF COMPUTER APPLICATIONS**

**BHARATHIAR UNIVERSITY**

**COIMBATORE – 641 046**

**APRIL-2024**

**DECLARATION**

I hereby declare that this project, titled **“DATA-DRIVEN DEFENSE: MACHINE LEARNING PARADIGM FOR NETWORK INTRUSION DETECTION”** submitted to the Department of Computer Applications, Bharathiar University, Coimbatore is a record of original project work done by **P R YASHWANTH RAJAN (22CSEG34)** under the supervision and guidance of **Mr. K. MOORTHY, MCA.,** Department of Computer Applications, Bharathiar University, Coimbatore and that this project work has not previously formed the basis of the award of the Degree / Diploma / Associate Ship / Fellowship or similar title to any candidate of any university.

**Place:** Coimbatore Signature of Candidate

**Date: (P R YASHWANTH RAJAN)**

Countersigned by

**Mr. K. MOORTHY, MCA.,**

Department of Computer Applications

**CERTIFICATE**

This is to certify that the project work title **“DATA-DRIVEN DEFENSE: MACHINE LEARNING PARADIGM FOR CYBER ATTACK DETECTION”** submitted to the Department of Computer Applications, Bharathiar University in partial fulfilment of the requirement for the award of a Degree in **Master of Science in Data Analytics,** is a record of the original work done by **P R YASHWANTH RAJAN (22CSEG34)** under my supervision and guidance and this project work has not formed the basis of the award of any Degree / Diploma / Associate Ship / Fellowship or similar title to any candidate of any university.

**Place:** Coimbatore

**Date:**

**Project Guide Head of the Department**

Submitted for the University Viva-Voce Examination held on ……………

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

I express my respectful thanks to our Professor & Head of the Department **Dr. M. PUNITHAVALLI ,M.Sc., M.Phil., Ph.D**, Department of Computer Applications, Bharathiar University, for permitting me to carry out my major project report work in “**DATA-DRIVEN DEFENSE: MACHINE LEARNING PARADIGM FOR CYBER ATTACK DETECTION**”

I really deem it a special privilege to convey my prodigious and everlasting thanks to my guide **Mr. K. MOORTHY, MCA.,** Department of Computer Applications, Bharathiar University, for his valuable guidance and personal interest in this major project report work. Last but not least, I also acknowledge the help done by my parents and acknowledge the encouraging support of my friends who were involved in this major project, in one way or the other. I thank Almighty for showering the divine grace on me and offer prayers to the lord for everything that was given to me.

|  |  |  |
| --- | --- | --- |
| **S. NO** | **CONTENT** | **PAGE NO** |
|  | **ABSTRACT** |  |
| **1** | **INTRODUCTION** |  |
| **2** | **ABOUT THE DATASET**  2.1 DATASET CREATION  2.2 FEATURES OF DATASET  2.3 ATTACKS TO BE CLASSIFIED |  |
| **3** | **DATA PREPROCESSING AND EDA**  3.1 EXPLORATORY DATA ANALYSIS  3.2 ANALYSING NUMERICAL VARIABLES  3.2.1 PLOTTING DISCREATE VARIABLES  3.2.2 PLOTTING CONTINUOUS VARIABLES  3.3 ANALYSING CATEGORICAL VARIABLES  3.3.1 PLOTTING THE TARGET VARIABLES |  |
| **4** | **CORRELATION**  4.1 FEATURES CORRELATED WITH LABEL |  |
| **5** | **FEATURE IMPORTANCE RANKING** |  |
| **6** | **DATA TRANSFORMATION**  6.1 ENCODING  6.2 SPLITTING THE DATASET FOR TRAINING AND TESTING  6.3 STANDARDIZATION |  |
| **7** | **MODEL BUILDING AND SAVING**  7.1 MODEL BUILDING & SAVING FOR BINARY CLASSIFICATION("LABEL")  7.2 MODEL BUILDING & SAVING FOR MULTICLASS CLASSIFICATION("ATTACK CATEGORY") |  |
| **8** | **FUTURE WORK**  8.1 IMPROVE THE Y2 MODEL  8.2 DEPLOY A WEB APPLICATION  8.3 3.ENHANCING EXISTING APPLICATIONS WITH NETWORK SECURITY: A SEAMLESS INTEGRATION |  |
| **9** | **CONCLUSION** |  |
| **10** | **BIBILOGRAPHY** |  |

**TABLE OF THE CONTENT**

**ABSTRACT**

In a world where cyber threats increasingly endanger digital networks and systems, the significance of robust cyber security measures cannot be overstated. This project focuses on enhancing cyber security through the implementation of machine learning algorithms for the detection of cyber attacks. The methodology involves the analysis of network data to identify potential threats by establishing correlations among various variables. By leveraging machine learning, the accuracy and efficiency of cyber attack detection are significantly improved, thereby fortifying the security of digital networks and systems. The cyber attack data utilized in this study is sourced from the University of New South Wales (Australia). The dataset obtained from the Cyber Range Lab of UNSW Canberra, comprises two sets of training and testing data. It encompasses records of nine distinct types of cyber attacks, including Fuzzers, Analysis, Backdoors, DoS (Denial of Service), Exploits, Generic, Reconnaissance, Shellcode, and Worms. Each attack type represents a unique threat vector, ranging from exploiting vulnerabilities to disrupting system functionality. The proposed approach empowers organizations to proactively identify cyber threats by employing advanced machine learning techniques. Through the analysis of diverse attack scenarios, the model can adapt and evolve, ensuring robust detection against a broad spectrum of cyber threats. This research contributes to the ongoing efforts to strengthen cyber security practices and serves as a valuable resource for organizations seeking to safeguard their digital infrastructure.

1. **INTRODUCTION**

In the current era of rapid growth in computer networks and applications, cybersecurity research faces numerous challenges. Intrusions or attacks can be defined as events capable of compromising fundamental principles of computer systems, including availability, authority, confidentiality, and integrity. Traditional firewall systems struggle to detect modern attack environments and lack the ability to analyse network packets in-depth. To address these limitations, Intrusion Detection Systems (IDSs) are designed to provide robust protection for our cybersecurity infrastructure.

A Network Intrusion Detection System (NIDS**)** monitors the flow of network traffic to identify potential attacks. NIDSs fall into two main categories: misuse/signature-based andanomaly-based. Signature-based systems match incoming data against known attack patterns to detect intrusions. In contrast, anomaly-based systems create a normal profile based on typical network behaviour, flagging any deviations as potential attacks.

The effectiveness of NIDSs is evaluated based on their ability to accurately identify attacks. This evaluation requires comprehensive datasets containing both normal and abnormal behaviours. Notably, older benchmark datasets like KDDCUP 99and NSLKDD have been widely adopted for assessing NIDS performance.

In this project, we will analyse the UNSW**-**NB15dataset, preprocess the data, perform detailed exploratory data analysis (EDA), and develop machine learning algorithms to identify specific attacks included in the dataset.

1. **ABOUT DATASET**

The total number of records is two million and 540,044 which are stored in the four CSV files, namely, UNSW-NB15\_1.csv, UNSW-NB15\_2.csv, UNSW-NB15\_3.csv and UNSW-NB15\_4.csv.

The ground truth table is named UNSW-NB15\_GT.csv and the list of event file is called UNSW-NB15\_LIST\_EVENTS.csv.

A partition from this dataset was configured as a training set and testing set, namely, UNSW\_NB15\_training-set.csv and UNSW\_NB15\_testing-set.csv respectively. The number of records in the training set is 175,341 records and the testing set is 82,332 records from the different types, attack and normal.

There are 49 Features in this dataset , These features are described in UNSW-NB15\_features.csv file.

This Dataset is Collected from:

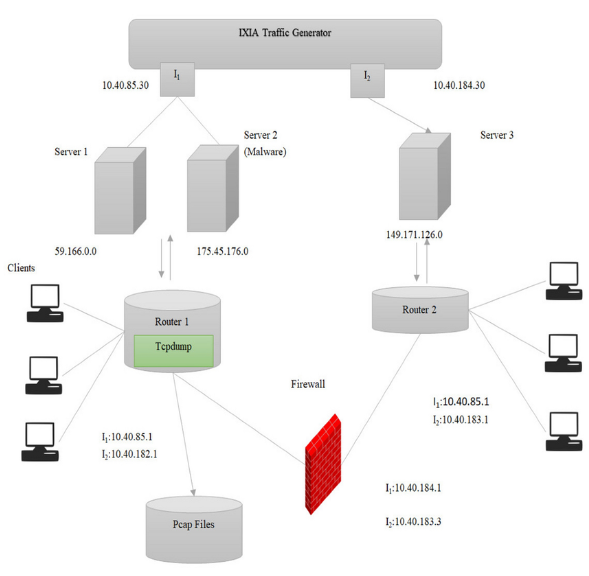
REPOSITORY - IEEE Data port

LINK **-** [https://ieee-dataport.org/documents/unswnb15-dataset#files](https://ieee-dataport.org/documents/unswnb15-dataset%23files)

**2.1 DATASET CREATION**

One of the major challenges in this field is the unavailability of a comprehensive network based data set which can reflect modern network traffic scenarios, vast varieties of low footprint intrusions and depth structured information about the network traffic. Evaluating network intrusion detection systems research efforts, KDD98, KDDCUP99 and NSLKDD benchmark data sets were generated a decade ago. However, numerous current studies showed that for the current network threat environment, these data sets do not inclusively reflect network traffic and modern low footprint attacks . This data set has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. Existing and novel methods are utilised to generate the features of the UNSWNB15 data set

The raw network packets of the UNSW-NB 15 dataset was created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) for generating a hybrid of real modern normal activities and synthetic contemporary attack behaviours. The Tcpdump tool is utilised to capture 100 GB of the raw traffic (e.g., Pcap files). This dataset has nine types of attacks, namely **Fuzzers , Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms**. The Argus, Bro-IDS tools are used and twelve algorithms are developed to generate totally 49 features with the class label. The number of records in the training set is 175,341 records and the testing set is 82,332 records from the different types, attack and normal.



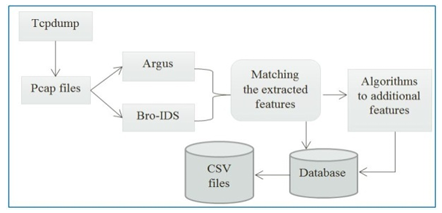


Fig 2.1 Framework Architecture for Generating UNSW-NB15 dataset

According to Fig. 2.1, the IXIA traffic generator is configured with the three virtual servers. The servers 1 and 3 are configured for normal spread of the traffic while server 2 formed the abnormal/malicious activities in the network traffic. Establishing the intercommunication between the servers, acquiring public and private network traffic, there are two virtual interfaces having IP addresses, 10.40.85.30 and 10.40.184.30. The servers are connected to hosts via two routers. The router 1 has 10.40.85.1 and 10.40.182.1 IP addresses, whereas router 2 is configured with 10.40.184.1 and 10.40.183.1 IP addresses. These routers are connected to the firewall device that is configured to pass all the traffic either normal or abnormal. The tcpdump tool is installed on the router 1 to capture the Pcap files of the simulation uptime. Moreover, the central intent of this whole testbed was to capture the normal or abnormal traffic, which was originated from the IXIA tool and dispersed among network nodes (e.g., servers and clients). Importantly, the IXIA tool is utilised as an attack traffic generator along with as normal traffic, the attack behaviour is nourished from the CVE site for the purpose of a real representation of a modern threat environment.

The whole architecture which is involved in generating the final shape of the UNSW-NB15 from pcap files to CSV files with 49 features (attributes in any CSV file) is presented in Fig. 3. All the 49 features of the UNSW-NB15 data set are elaborated from Tables II-VII along with the generation sequence explanation for understanding convenience.

**2.2 FEATURES OF DATASET**

The dataset comprises 49 features, and the accompanying table provides the feature names, their data types, and descriptions :

|  |  |  |  |
| --- | --- | --- | --- |
| S.no | Name | Type | Description |
| 1 | srcip | nominal | Source IP address |
| 2 | sport | integer | Source port number |
| 3 | dstip | nominal | Destination IP address |
| 4 | dsport | integer | Destination port number |
| 5 | proto | nominal | Transaction protocol |
| 6 | state | nominal | Indicates to the state and its dependent protocol, e.g. ACC, CLO, CON, ECO, ECR, FIN, INT, MAS, PAR, REQ, RST, TST, TXD, URH, URN, and (-) (if not used state) |
| 7 | dur | Float | Record total duration |
| 8 | sbytes | Integer | Source to destination transaction bytes |
| 9 | dbytes | Integer | Destination to source transaction bytes |
| 10 | sttl | Integer | Source to destination time to live value |
| 11 | dttl | Integer | Destination to source time to live value |
| 12 | sloss | Integer | Source packets retransmitted or dropped |
| 13 | dloss | Integer | Destination packets retransmitted or dropped |
| 14 | Service | nominal | http, ftp, smtp, ssh, dns, ftp-data ,irc and (-) if not much used service |
| 15 | Sload | Float | Source bits per second |
| 16 | Dload | Float | Destination bits per second |
| 17 | Spkts | integer | Source to destination packet count |
| 18 | Dpkts | integer | Destination to source packet count |
| 19 | swin | integer | Source TCP window advertisement value |
| 20 | dwin | integer | Destination TCP window advertisement value |
| 21 | stcpb | integer | Source TCP base sequence number |
| 22 | dtcpb | integer | Destination TCP base sequence number |
| 23 | smeansz | integer | Mean of the how packet size transmitted by the src |
| 24 | dmeansz | integer | Mean of the how packet size transmitted by the dst |
| 25 | trans\_depth | integer | Represents the pipelined depth into the connection of http request/response transaction |
| 26 | res\_bdy\_len | integer | Actual uncompressed content size of the data transferred from the server’s http service. |
| 27 | Sjit | Float | Source jitter (mSec) |
| 28 | Djit | Float | Destination jitter (mSec) |
| 29 | Stime | Timestamp | record start time |
| 30 | Ltime | Timestamp | record last time |
| 31 | Sintpkt | Float | Source interpacket arrival time (mSec) |
| 32 | Dintpkt | Float | Destination interpacket arrival time (mSec) |
| 33 | tcprtt | Float | TCP connection setup round-trip time, the sum of ’synack’ and ’ackdat’. |
| 34 | synack | Float | TCP connection setup time, the time between the SYN and the SYN\_ACK packets. |
| 35 | ackdat | Float | TCP connection setup time, the time between the SYN\_ACK and the ACK packets. |
| 36 | is\_sm\_ips\_ports | Binary | If source (1) and destination (3)IP addresses equal and port numbers (2)(4) equal then, this variable takes value 1 else 0 |
| 37 | ct\_state\_ttl | Integer | No. for each state (6) according to specific range of values for source/destination time to live (10) (11). |
| 38 | ct\_flw\_http\_mthd | Integer | No. of flows that has methods such as Get and Post in http service. |
| 39 | is\_ftp\_login | Binary | If the ftp session is accessed by user and password then 1 else 0. |
| 40 | ct\_ftp\_cmd | Integer | No of flows that has a command in ftp session. |
| 41 | ct\_srv\_src | Integer | No. of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26). |
| 42 | ct\_srv\_dst | Integer | No. of connections that contain the same service (14) and destination address (3) in 100 connections according to the last time (26). |
| 43 | ct\_dst\_ltm | Integer | No. of connections of the same destination address (3) in 100 connections according to the last time (26). |
| 44 | ct\_src\_ ltm | Integer | No. of connections of the same source address (1) in 100 connections according to the last time (26). |
| 45 | ct\_src\_dport\_ltm | Integer | No of connections of the same source address (1) and the destination port (4) in 100 connections according to the last time (26). |
| 46 | ct\_dst\_sport\_ltm | Integer | No of connections of the same destination address (3) and the source port (2) in 100 connections according to the last time (26). |
| 47 | ct\_dst\_src\_ltm | Integer | No of connections of the same source (1) and the destination (3) address in in 100 connections according to the last time (26). |
| 48 | attack\_cat | Nominal | The name of each attack category. In this data set , nine categories e.g. Fuzzers, Analysis, Backdoors, DoS Exploits, Generic, Reconnaissance, Shellcode and Worms |
| 49 | Label | binary | 1. for normal and 1 for attack records |

**2.3 ATTACKS TO BE CLASSIFIED**

The table below provides a description of the attacks to be classified by the algorithm, along with the corresponding number of records used in both the training and testing datasets.

|  |  |  |
| --- | --- | --- |
| Type | No of Records | Description |
| Normal | 2,218,761 | Natural transaction data. |
| Fuzzers | 24,246 | Attempting to cause a program or network suspended by feeding it the randomly generated data. |
| Analysis | 2,677 | It contains different attacks of port scan, spam and html files penetrations. |
| Backdoors | 2,329 | A technique in which a system security mechanism is bypassed stealthily to access a computer or its data. |
| Dos | 16,353 | A malicious attempt to make a server or a network resource unavailable to users, usually by temporarily interrupting or suspending the services of a host connected to the Internet. |
| Exploits | 44,525 | The attacker knows of a security problem within an operating system or a piece of software and leverages that knowledge by exploiting the vulnerability. |
| Generic | 215,481 | A technique works against all block ciphers (with a given block and key size), without consideration about the structure of the block-cipher. |
| Reconnaissance | 13,987 | Contains all Strikes that can simulate attacks that gather information. |
| Shellcode | 1,511 | A small piece of code used as the payload in the exploitation of software vulnerability. |
| Worms | 174 | Attacker replicates itself in order to spread to other computers. Often, it uses a computer network to spread itself, relying on security failures on the target |

**3. DATA PREPROCESSING AND EDA**

Data preprocessing and EDA are pivotal steps in constructing a robust and accurate predictive model for network intrusion detection. Here are the key steps involved in this process:

**Exploratory Data Analysis (EDA):**

* EDA entails visualizing and analyzing the data to gain insights into the relationships between variables.
* By examining patterns, trends, and outliers, EDA informs subsequent feature engineering and selection.

**Data Cleaning:**

* This step addresses missing values, outliers, and discrepancies in the dataset.
* Missing values can be imputed, and outliers detected using statistical methods or domain knowledge.
* Discrepancies are resolved by cross-checking with other sources or removing inconsistent data.

**Feature Engineering:**

* Feature engineering involves selecting relevant features that influence the target variable.
* Transforming categorical variables into numerical format (e.g., one-hot encoding) ensures usability.
* Feature scaling or normalization ensures equal importance across variables in the model.

**Feature Selection:**

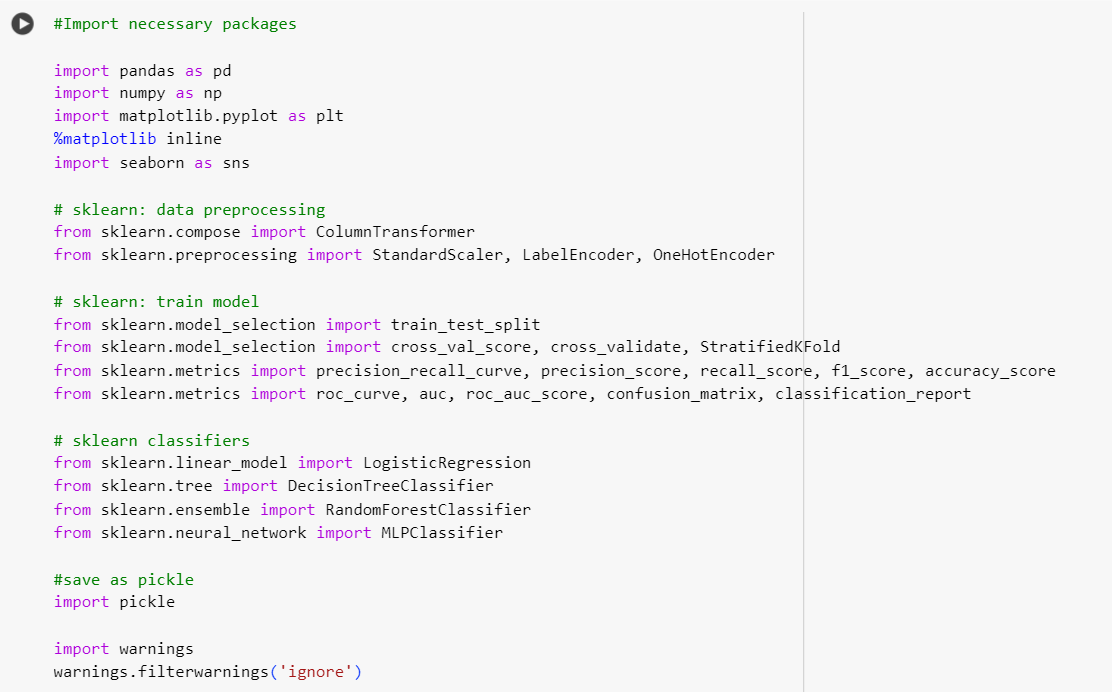
* This step involves selecting a subset of the features that are most relevant to the target variable.
* Feature selection can help to improve model performance and reduce overfitting. Feature importance can be determined using statistical methods, such as correlation or mutual information, or by using machine learning algorithms, such as random forest or gradient boosting.

The data preprocessing and EDA plays an important role in the Machine learning Model Development . This process involves the above mentioned steps ,These steps can help to ensure that the data is clean, relevant, and informative for building an accurate and reliable predictive model.

The dataset has already divided into train and test sets that files are used to develop the model, initially the dataset contains 49 Features but the train & test files has only 44 features the rest are omitted according to high Correlation and removed based on the Domain knowledge.

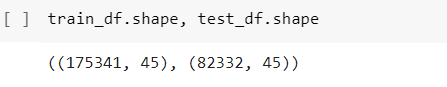
**3.1 EXPLORATORY DATA ANALYSIS**

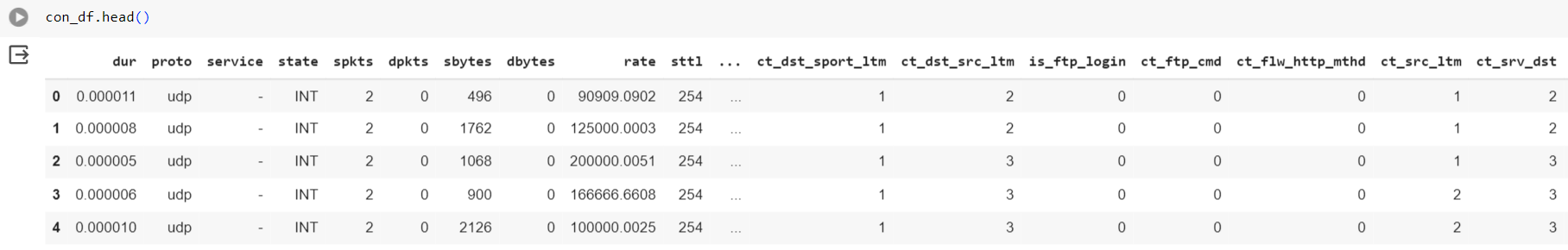
**Importing Necessary packages & Viewing Shape and Head of the Data**

****

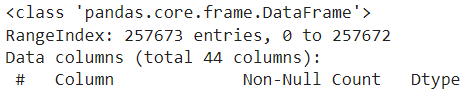
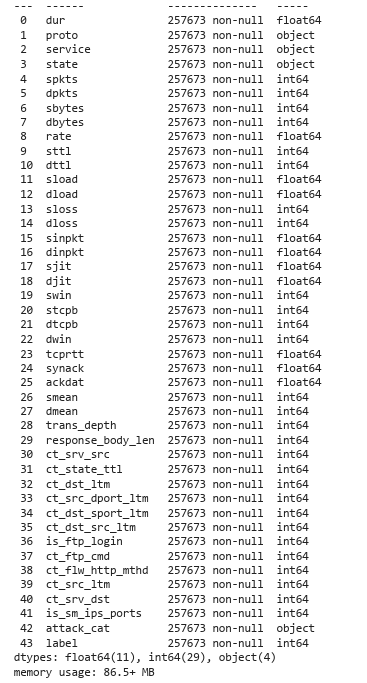
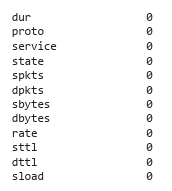
The necessary packages for Dataframe manipulation , visualization ,splitting training and testing data, and to perform Machine learning models where imported .

**Shape of the Training and Testing data is Viewed**

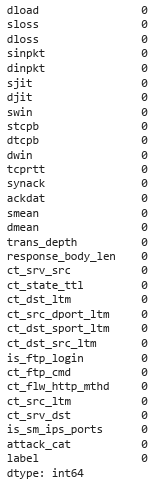




**Checking Null values and Data types of the features**

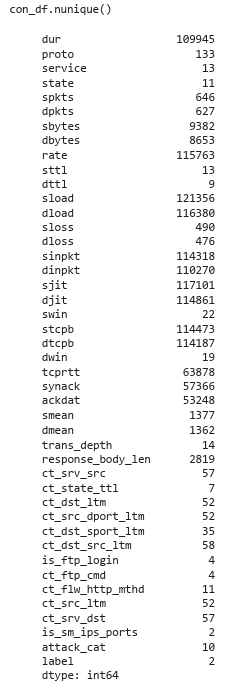






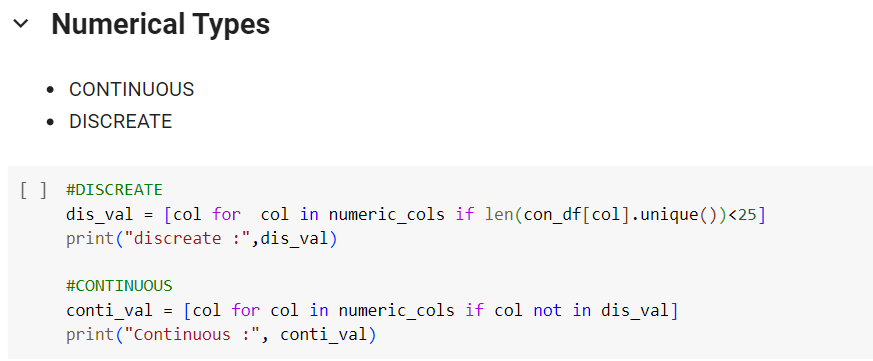
* The dataset contains – int64 , float64 & object data types , all the features have equal number of records there is no missing values.
* The dataframe contains totally – 44 columns and 2,57,673 Rows

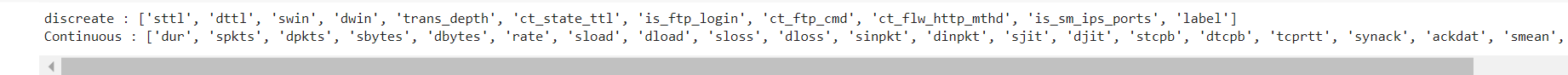
**Checking Number of unique values for each features**



Here, All the features have different number of unique features range (2 – 109945), the features are divided into Numerical and categorical variables .

**3.2 ANALYSISNG NUMERICAL VARIABLES**

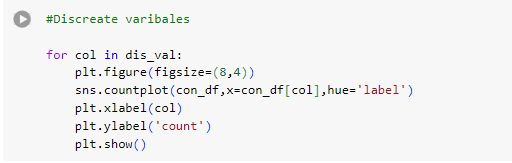




The numerical variables are divided into two categories numerical and categorical according to the unique values .

**3.2.1 PLOTTING DISCREATE VARIABLES .**

The discreate variables are stores in “dis\_val” , which is looped and the plots are plotted from that few plots where shown below



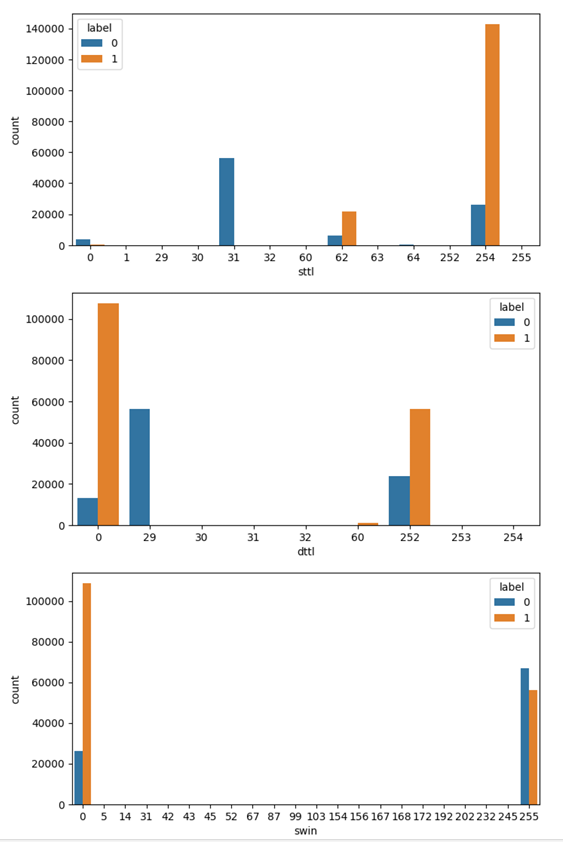
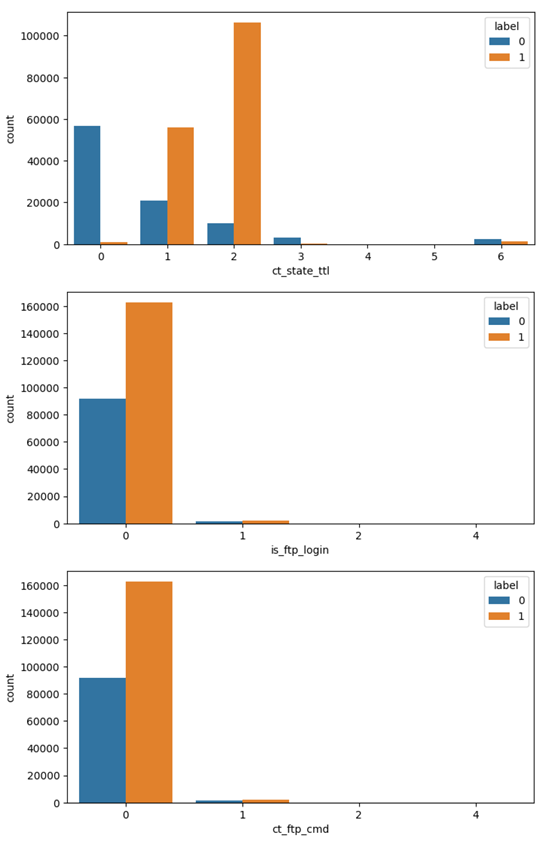
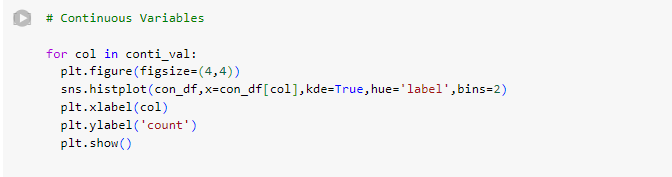


Fig 3.1 Plotting discreate variables

This discreate values where classified by labels using “Hue” parameter , where the 0 indicates normal and the “1” indicates the attacked Data .

**3.2.2 PLOTTING CONTINUOUS VARIABLES.**



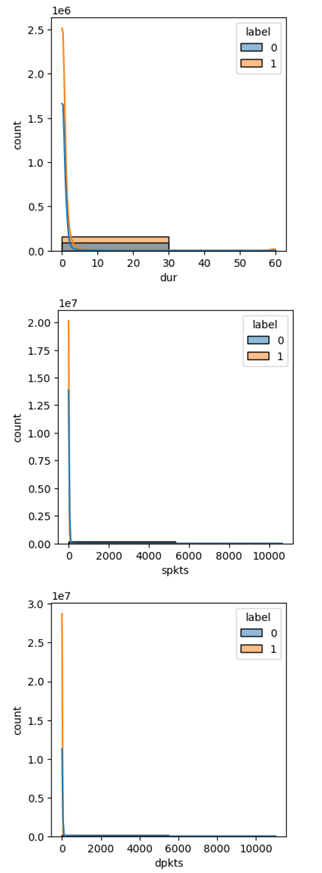
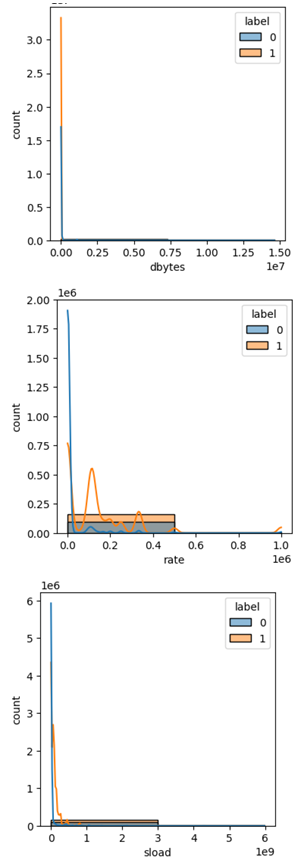
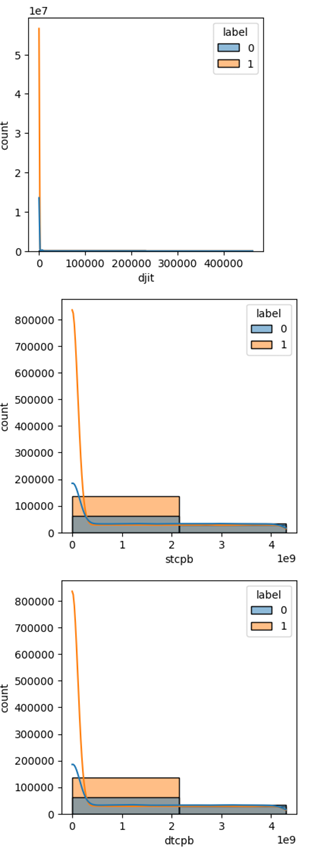
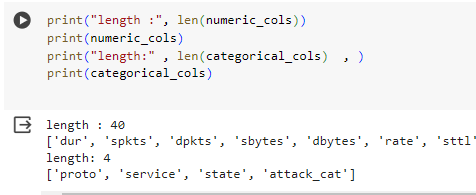


Fig 3.2 Plotting Continuous Variables.

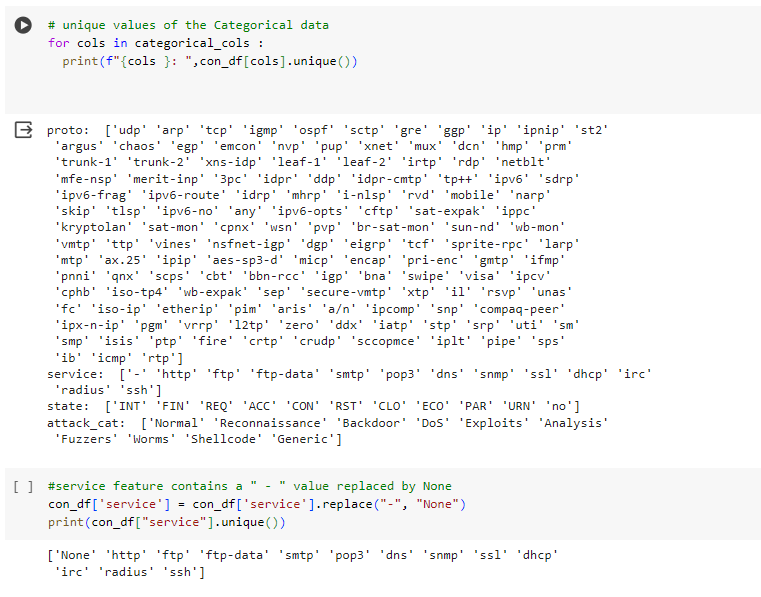
The continuous variables are stored in “conti\_val” looped and plotted with the hue of “Label”, only few of the plots where shown above from the Analysis this are used to view the range and distribution of the particular Feature.

**3.3 ANALYSING CATEGORICAL VARIABLES**

There are 4 Categorical Variables which are stored in a variable called categorical\_cols:



* The unique values present in the categorical columns are listed below, in which the “proto” has more number of sub- categories.
* In “service” column we could able to see a value “-” , it is not a good practice to keep the value as a symbol in a sub-category so it is changed as “None ” which indicates no service request .



**3.3.1 PLOTTING THE TARGET VARIABLES – LABELS & ATTACK CATEGORY**

The dataset includes two target variables: ‘Label’ indicates whether the network is under attack or not, while ‘Attack category’ specifies the type of attack encountered. Consequently, we construct two separate models: one classifies network status (attack or not), and the other identifies the specific attack initiated.

**Labels – Classifies whether the network is under attack or in Normal state.**

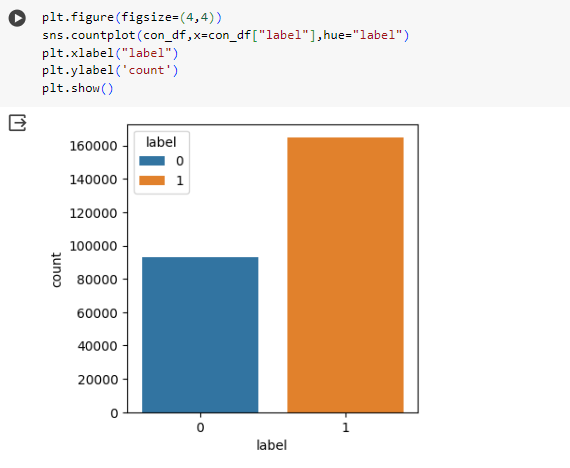
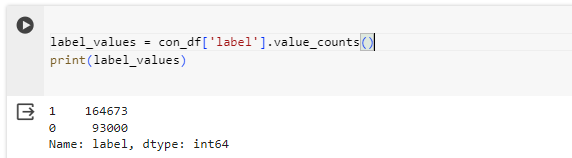


Fig 3.3 Plotting Label – target variable count

 0 – Indicates Normal Values, 1 – Indicates Attack Values

According to the count present in the dataset Attack values are more compared to the normal values.

**Attack Category – Classifies the types of attack encountered.**

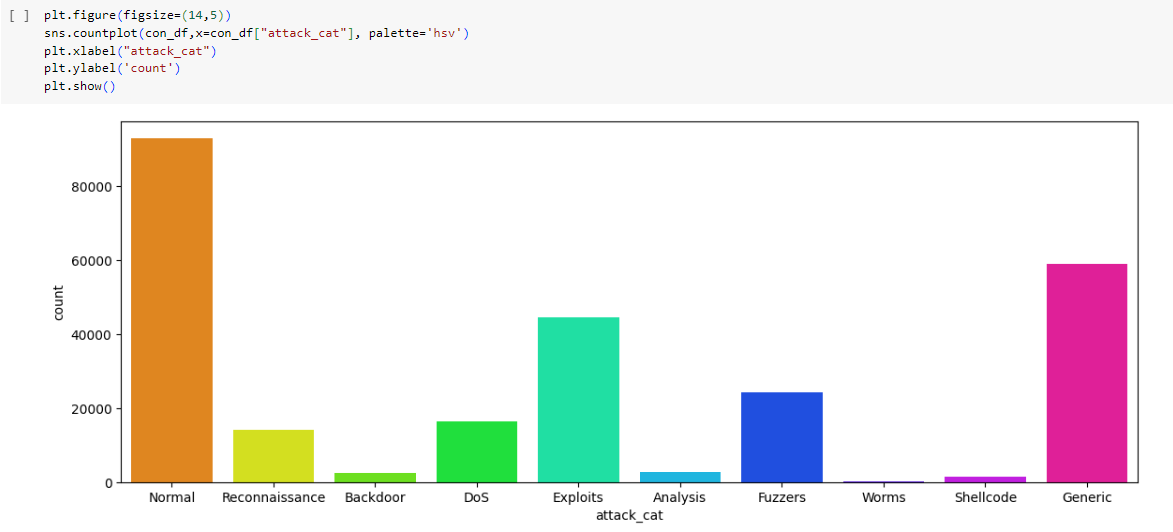
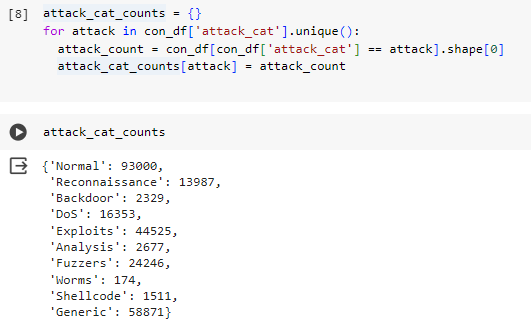


Fig 3.4 Plotting Attack category - target variable

From the above plot we could see the dataset is imbalance according to the attack category , the normal data has more records compared to the distinct attacks in the dataset.



The exact values of the attack categories are listed above from that the “normal” has more records comparatively and the “worms” and “Backdoor” has comparatively low records this may lead to bias and the model will unable to predict the respective attacks accurately.

**4. CORRELATION**

To improve our understanding of the variables involved in cyber attack detection, we need to analyze the network data. Correlation diagrams can be helpful in visualizing how different variables are associated with each other and with cyber attacks. Additionally, random forest models can help identify the importance of different features in predicting the target variable (cyber attacks). We can compare the feature rankings from the random forest with the results of the correlation analysis to gain a better understanding of the key features to focus on for effective cyber attack detection.

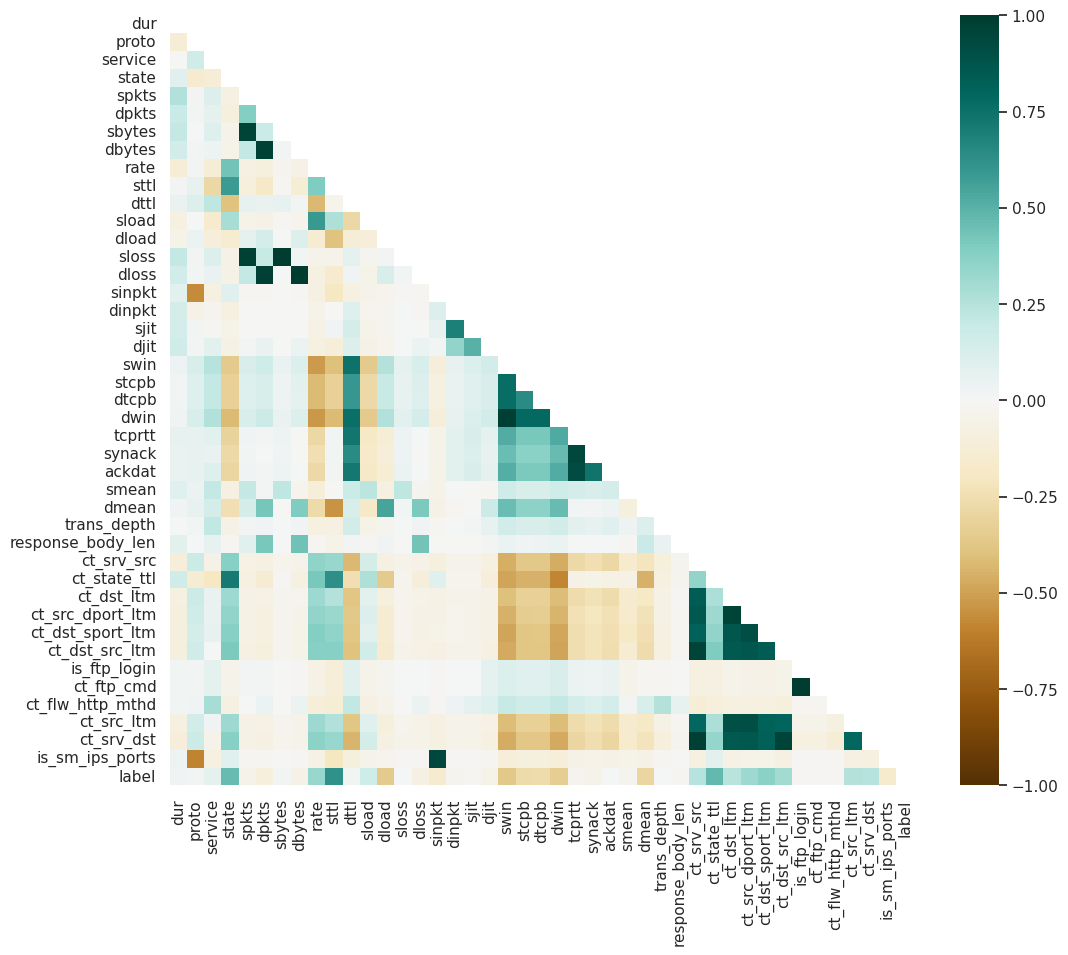
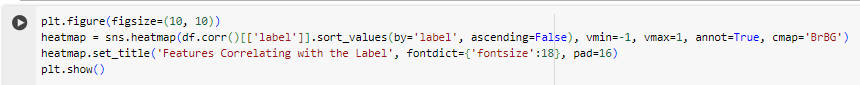


Fig 4.1 Plotting correlation among the features

**4.1 FEATURES CORRELATED WITH LABEL**

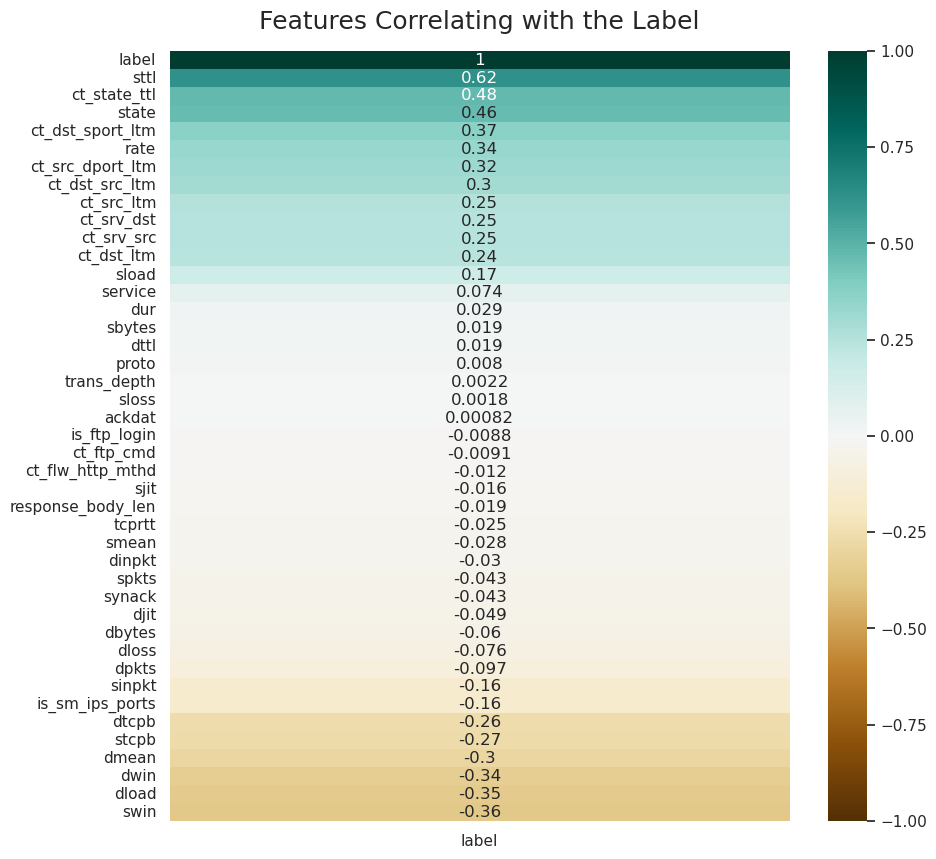


Fig 4.2 Plotting the feature correlating with the labels

The following variables are positively correlated with cyber attacks:

* sttl: Source to destination time to live value.
* ct\_state\_ttl and state: These features reflect various stages of TCP connections and may be related to port scanning, SYN flood, or DDoS attacks.
* ct\_dst\_sport\_ltm: This feature measures the number of connections from the same source IP to the same destination port in a short time period.
* rate: This feature may represent various types of traffic rates or frequencies.

The following variables are negatively correlated with cyber attacks:

* swin
* dload

**5. FEATURE IMPORTANCE RANKING**

1. **Identifying Key Predictors:**

* The top-ranked features are likely the most informative for predicting the target variable.
* These features play a crucial role in the model’s decision-making process.

1. **Feature Selection:**

* Based on the importance scores, you can decide whether to include all features or focus on a subset of the most relevant ones.
* Removing less important features can simplify the model and improve computational efficiency.

1. **Domain Understanding:**

* Analyzing the top features provides insights into the underlying data and domain.
* For example, if certain network protocol features (e.g., TCP, UDP) are highly ranked, it suggests their significance in detecting attacks.

1. **Model Interpretability:**

* Feature importance helps explain the model’s predictions to stakeholders.
* You can communicate which features contribute most to the model’s decisions.

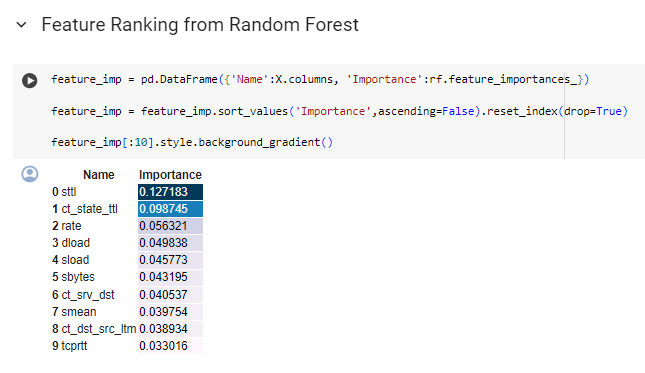


Fig 5.1 Plotting feature importance among the features

**STEPS INVOLVED IN FEATURE RANKING FROM THE RANDOM FOREST :**

1. **Creating a DataFrame for Feature Importance:**
   * A Random Forest (RF) model has been trained, and its feature importances are calculated
   * The feature\_imp DataFrame is created with two columns:

i. ‘Name’: Represents the names of the input features (columns) in the dataset.

ii. ‘Importance’: Indicates the importance score assigned to each feature by the RF model.

1. **Sorting Features by Importance:**

* The feature\_imp DataFrame is sorted in descending order based on the importance scores.
* This step ensures that the most important features appear at the top of the DataFrame.

1. **Top Features:**
   * The code selects the top 10 features (those with the highest importance scores) using feature\_imp[:10].
   * These features are likely the most influential in predicting the target variable.

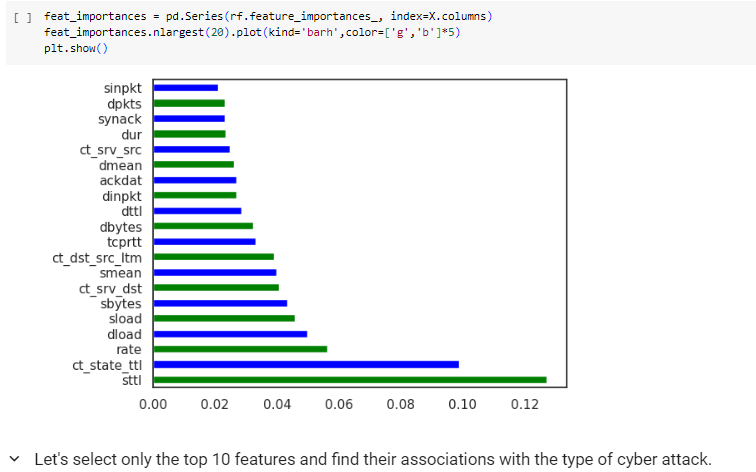


Fig 5.2 Plotting the feature importance in sorted form

**6. DATA TRANSFORMATION**

* 1. **ENCODING**

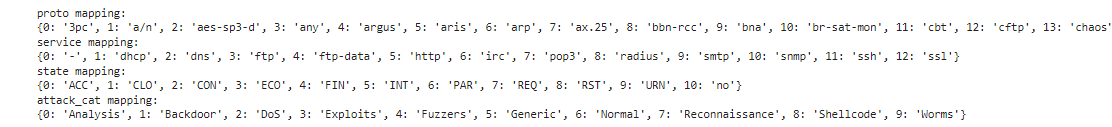
Most real-life datasets we encounter during our data science project development have columns of mixed data type. These datasets consist of both categorical as well as numerical columns. However, various Machine Learning models do not work with categorical data and to fit this data into the machine learning model it needs to be converted into numerical data. One approach to solve this problem can be label encoding where we will assign a numerical value to these labels for example Male and Female mapped to 0 and 1. But this can add bias in our model as it will start giving higher preference to the Female parameter as 1>0 but ideally, both labels are equally important in the dataset. To deal with this issue we will use the One Hot Encoding technique.

Disadvantage Of Using One-Hot Encoding To This Dataset

* It can lead to increased dimensionality, as a separate column is created for each category in the variable. This can make the model more complex and slow to train.
* It can lead to sparse data, as most observations will have a value of 0 in most of the one-hot encoded columns.
* It can lead to overfitting, especially if there are many categories in the variable and the sample size is relatively small.
* One-hot-encoding is a powerful technique to treat categorical data, but it can lead to increased dimensionality, sparsity, and overfitting. It is important to use it cautiously and consider other methods such as ordinal encoding or binary encoding.

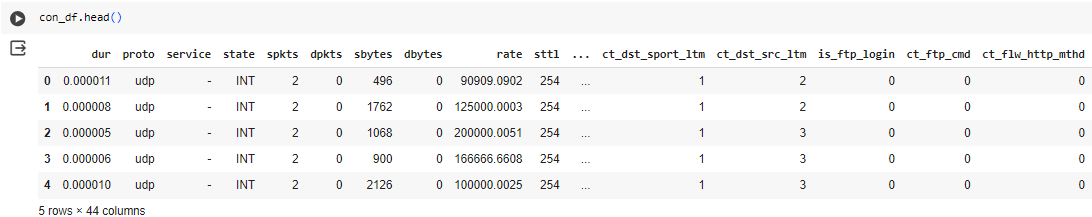
So here we tried “cat.codes.astype(int)” – inbuilt function to change the categorical values to numerical and it performs well in the binary classification.



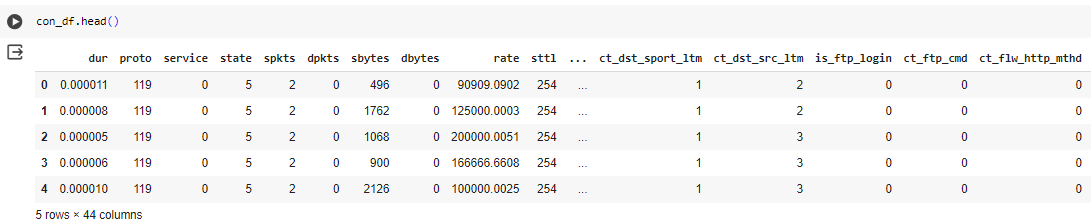


The above result shows how the categorical values are mapped to the numerical values of every categorical features , then it is converted into “int” as mentioned below



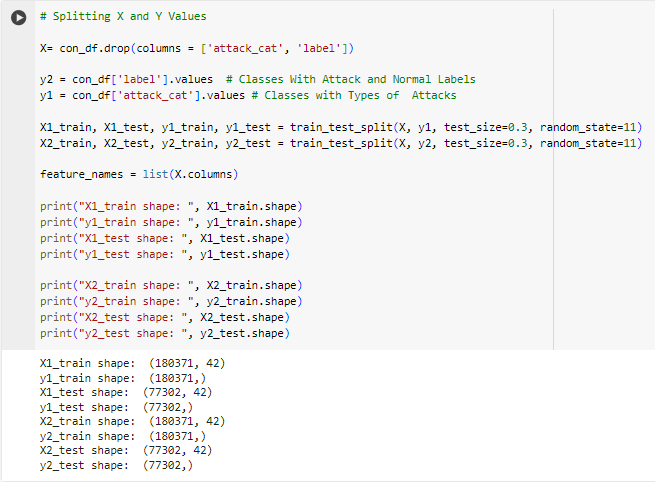
**Before Encoding :**

**After Encoding :**



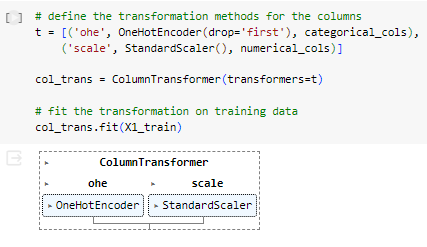
* 1. **SPLITTING THE DATASET FOR TRAINING AND TESTING**

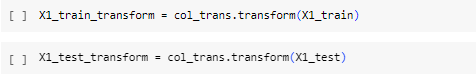
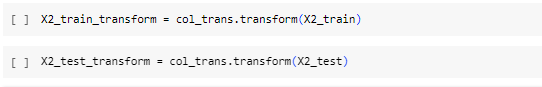
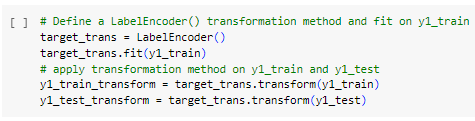
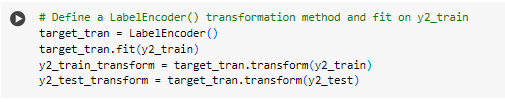
The dataset consists of two target variables which are “label” and “Attack category” so the dataset is splitted into two sets where the X1 – test, train and y1 – test , train are to train and test the Attack category based model . The X2 – train, test and y2 – test , train are to train and test the Label based models .



* 1. **STANDARDIZATION**

Standardization is a preprocessing technique commonly used in machine learning. Its purpose is to standardize the scale of input features. By removing the mean and scaling to unit variance, StandardScaler ensures that all features contribute equally to the model. Essentially, it transforms the data to follow a standard normal distribution, with a mean of 0 and a standard deviation of 1. This process is crucial for algorithms sensitive to differences in feature scales, enhancing model performance and facilitating meaningful comparisons across features .





Train and test data for both y1 & y2 target variables where standardized using “StandardScaler()”.

**7. MODEL BUILDING AND SAVING**

We have used multiple Machie learning models to verify which is performing well in this dataset for that we have used Stratified 5-Fold Cross-Validation, which is mainly used to compare different models and is particularly useful when dealing with imbalanced datasets**.**

**Cross-Validation**:

* Cross-validation is a technique used to evaluate the performance of a machine learning model on unseen data.
* It involves dividing the available data into multiple folds or subsets.
* One of these folds serves as a validation set, while the remaining folds are used for training the model.
* The process is repeated multiple times, each time using a different fold as the validation set.
* Finally, the results from each validation step are averaged to produce a more robust estimate of the model’s performance.
* Cross-validation helps prevent overfitting and provides a realistic estimate of how well the model will generalize to new, unseen data.

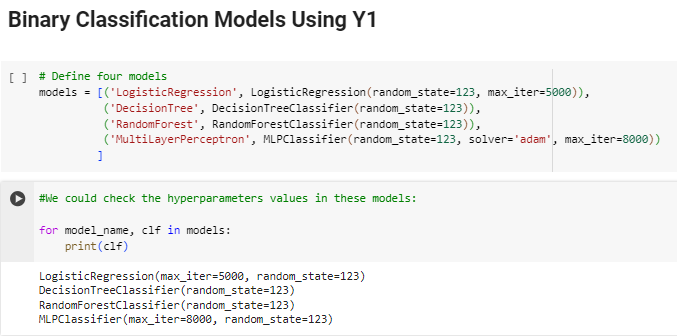
**Stratified 5-Fold Cross-Validation:**

* Stratified K-Fold cross-validation is a variation of K-Fold cross-validation.
* It ensures that the class distribution within each fold remains similar to the overall class distribution in the dataset.
* Specifically, it preserves the percentage of samples for each class.
* Stratified K-Fold is particularly useful when dealing with **imbalanced datasets**.
* In the case of 5-fold cross-validation, the data is split into five folds, and each fold serves as the validation set once.

**Why we have Used Cross-Validation?:**

* **Preventing Overfitting:** Cross-validation helps prevent overfitting by providing a more robust estimate of the model’s performance on unseen data.
* **Model Selection:** It allows us to compare different models and select the one that performs best on average.
* **Realistic Performance Estimate:** By evaluating the model on multiple validation sets, cross-validation reflects how well the model will generalize to new, unseen data.

**7.1. MODEL BUILDING FOR THE “LABEL”- TARGET VARIABLE (BINARY CLASSIFICATION)**



**Model Names :**

This list contains the names of the machine learning models used in the experiment (Logistic Regression, Decision Tree, Random forest , MLPClassifier).

**Model Definitions:**

Four machine learning models are defined:

**Logistic Regression**: A linear model for binary classification.

**Decision Tree:** A tree-based model for classification.

**Random Forest:** An ensemble of decision trees.

**Multi-Layer Perceptron (MLP):** A neural network with multiple hidden layers.

Each model is initialized with specific parameters (e.g., random\_state, max\_iter).

**Lists and DataFrame Initialization:**

* Several lists and a DataFrame are created to store cross-validation (CV) results and evaluation metrics on testing data.
* These lists will hold information such as model names, mean fit times, accuracy, precision, recall, F1 score and ROC AUC.

**Cross-Validation (CV) Loop:**

For each model:

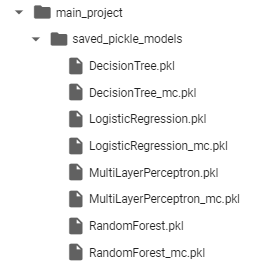
* A Stratified 5-fold cross-validator (cv) is defined.
* The specified scoring metrics (accuracy, precision, recall, F1, ROC AUC) are used for evaluation.
* The model is trained using the training data (X1\_train\_transform, y2\_train\_transform) and evaluated on validation sets.
* The mean values of the scores are calculated.
* CV results (fit time, accuracy, precision, etc.) are stored in the corresponding lists.

**Test Evaluation Metrics:**

* The model’s performance on the testing data will also be evaluated.
* Metrics such as accuracy, precision, recall, F1 score, and ROC AUC will be computed.

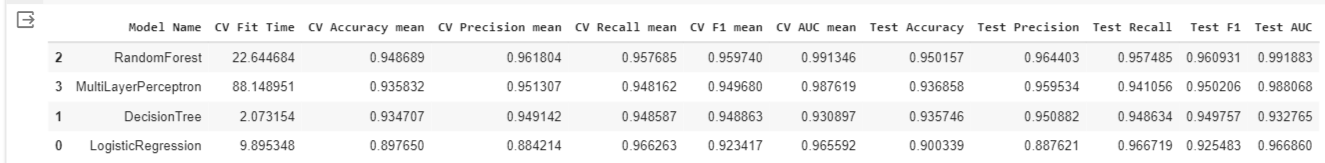
**Model is saved as Pickle file :**

Location – drive => MyDrive => main\_project => saved\_pickle\_models

****

the models are saved in pickle file for further development and Predictions .





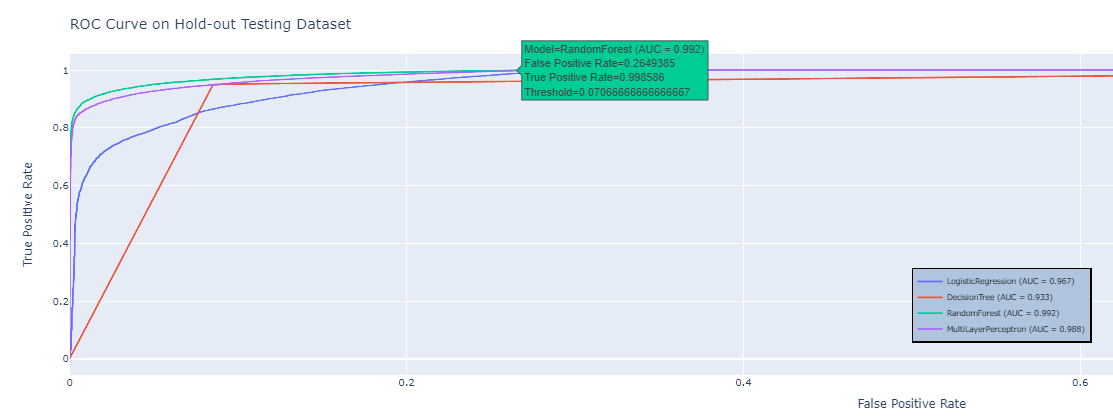
**ROC CURVE – TESTING DATA**

Fig 7.1 ROC CURVE – Test data

**CROSS-VALIDATION METRICS:**

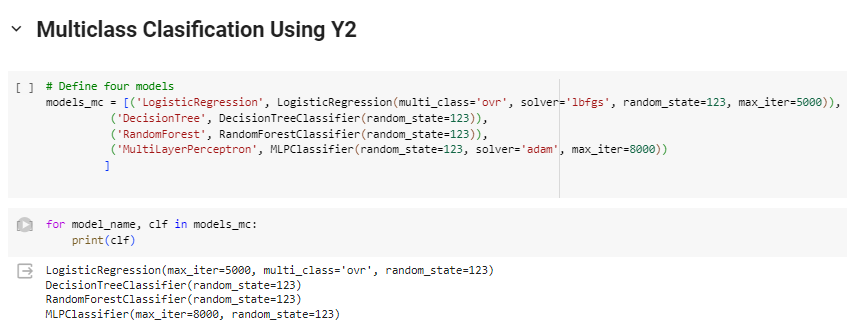
The following metrics are computed during cross-validation (CV) for each model:

* **CV Fit Time ('CV Fit Time'):** The average time taken by the model to fit (train) on each CV fold.
* **CV Accuracy Mean ('CV Accuracy mean'):** The average accuracy across CV folds.
* **CV Precision Mean ('CV Precision mean'):** The average precision (positive predictive value) across CV folds.
* **CV Recall Mean ('CV Recall mean'):** The average recall (true positive rate) across CV folds.
* **CV F1 Mean ('CV F1 mean'):** The average F1 score (harmonic mean of precision and recall) across CV folds.
* **CV AUC Mean ('CV AUC mean'):** The average area under the Receiver Operating Characteristic (ROC) curve across CV folds.

These metrics provide insights into how well the model performs during cross-validation. By this we can see that **Random forest** algorithm performs well in this dataset and gives the Accuracy rate of **95% .**

* 1. **MODEL BUILDING FOR THE “ATTACK CATEGORY”- TARGET VARIABLE**

**(MULTICLASS CLASSIFICATION).**



**Model Names :**

This list contains the names of the machine learning models used in the experiment (Logistic Regression, Decision Tree, Random forest , MultilayerPerceptron).

**Model Names :**

This list contains the names of the machine learning models used in the experiment (Logistic Regression, Decision Tree, Random forest , MLPClassifier).

**Model Definitions:**

Four machine learning models are defined:

**Logistic Regression**: A linear model for binary classification.

**Decision Tree:** A tree-based model for classification.

**Random Forest:** An ensemble of decision trees.

**Multi-Layer Perceptron (MLP):** A neural network with multiple hidden layers.

Each model is initialized with specific parameters (e.g., random\_state, max\_iter).

**Lists and DataFrame Initialization:**

* Several lists and a DataFrame are created to store cross-validation (CV) results and evaluation metrics on testing data.
* These lists will hold information such as model names, mean fit times, accuracy, precision, recall, F1 score and ROC AUC.

**Cross-Validation (CV) Loop:**

For each model:

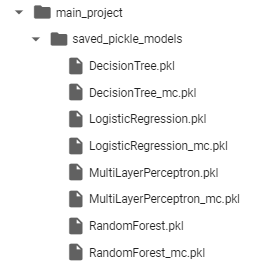
* A Stratified 5-fold cross-validator (cv) is defined.
* The specified scoring metrics (accuracy, precision, recall, F1, ROC AUC) are used for evaluation.
* The model is trained using the training data (X1\_train\_transform, y2\_train\_transform) and evaluated on validation sets.
* The mean values of the scores are calculated.
* CV results (fit time, accuracy, precision, etc.) are stored in the corresponding lists.

**Test Evaluation Metrics:**

* The model’s performance on the testing data will also be evaluated.
* Metrics such as accuracy, precision, recall, F1 score, and ROC AUC will be computed.

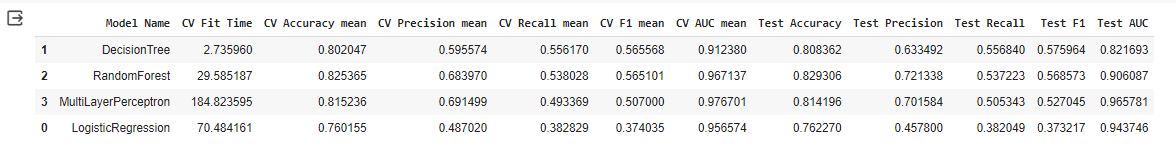
**Model is saved as Pickle file :**

Location – drive => MyDrive => main\_project => saved\_pickle\_models

****

the models are saved in pickle file for further development and Predictions .





**Cross-Validation Metrics:**

The following metrics are computed during cross-validation (CV) for each model:

* **CV Fit Time ('CV Fit Time'):** The average time taken by the model to fit (train) on each CV fold.
* **CV Accuracy Mean ('CV Accuracy mean'):** The average accuracy across CV folds.
* **CV Precision Mean ('CV Precision mean'):** The average precision (positive predictive value) across CV folds.
* **CV Recall Mean ('CV Recall mean'):** The average recall (true positive rate) across CV folds.
* **CV F1 Mean ('CV F1 mean'):** The average F1 score (harmonic mean of precision and recall) across CV folds.
* **CV AUC Mean ('CV AUC mean'):** The average area under the Receiver Operating Characteristic (ROC) curve across CV folds.

These metrics provide insights into how well the model performs during cross-validation. By this we can see that **Random forest** algorithm performs well in this dataset and gives the Accuracy rate of **82% .**

**8. FUTURE WORK**

**8.1 IMPROVE THE Y2 MODEL (MULTICLASS CLASSIFICATION)**

The second model, denoted as y2, targets the ‘Attack category.’ It is a multiclass classification model designed to classify instances into one of nine attack categories. Despite achieving an accuracy of 82%, the model faces challenges due to class imbalance. While it predicts individual attacks with reasonable probability, its overall global predictions lack precision and recall.

**Model Purpose:**

* The y2 model is specifically built to predict the type of attack (from nine possible categories) based on input features.
* Each category corresponds to a specific type of network intrusion or security threat.

**Multiclass Classification:**

* Multiclass classification involves assigning instances to one of several classes (in this case, attack categories).
* The model learns patterns from the dataset to make these predictions.

**Class Imbalance:**

* The dataset contains an unequal distribution of attack categories.
* Some categories may have significantly more instances than others.
* This imbalance affects the model’s ability to learn and generalize well.

**Accuracy vs. Precision and Recall:**

* The model achieves an overall accuracy of 82%.
* However, accuracy alone is not sufficient for evaluating performance, especially in imbalanced datasets.

**Precision**: The proportion of correctly predicted positive instances among all predicted positive instances.

**Recall**: The proportion of correctly predicted positive instances among all actual positive instances.

**SOLUTION:**

**Feature Engineering:**

* Review and preprocess the features: Ensure that the input features are relevant, properly scaled, and free from noise.
* Create new features: Extract meaningful information from existing features or engineer new ones that capture important patterns.

**Hyperparameter Tuning:**

* **Grid search or random search:** Experiment with different hyperparameters (e.g., learning rate, regularization strength) to find optimal values.
* **Cross-validation:** Use cross-validation to evaluate different hyperparameter settings and select the best combination.

**Model Selection:**

* **Try different algorithms**: Consider using other classification algorithms (e.g., Gradient Boosting, Support Vector Machines) to see if they perform better.
* **Ensemble methods:** Combine multiple models (e.g., bagging, boosting) to improve overall performance.

**Address Class Imbalance:**

The dataset has imbalanced classes, consider techniques such as oversampling, under sampling, or using synthetic data (SMOTE) to balance the class distribution.

**Feature Importance and Selection:**

* Analyze feature importance scores (if available) to focus on the most influential features.
* Remove irrelevant or redundant features.

**8.2 DEPLOY A WEB APPLICATION**

This Model can be deployed as a web Application using frontend Frameworks or Libraries the UI (user interface) can be created according to the requirements which intakes the necessary inputs which is used in our model .

**My Suggestion :**

We can develop a web application that accepts input both manually and via Excel files. Users can fill in the necessary values corresponding to the input data required by our model. Upon submission, the data will be sent to the model in the backend. The model will determine whether the network is under attack and classify the attack type if it aligns with the nine attack categories it has been trained on.

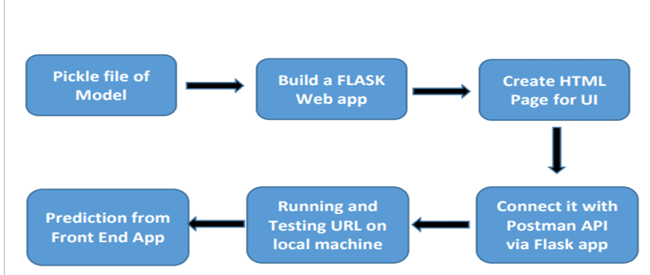


Fig 8.1 Workflow of the Web Application

* 1. **ENHANCING EXISTING APPLICATIONS WITH NETWORK SECURITY:**

**A SEAMLESS INTEGRATION**

Our model can seamlessly integrate into existing applications or cybersecurity detection components. By doing so, we enhance these applications with an additional layer of security. Here’s how it works:

**Integration:**

* We embed our trained model within the existing infrastructure.
* Whether it’s a web application, network monitoring tool, or security system, our model becomes an integral part of the solution.

**Real-Time Monitoring:**

* As the application runs, our model continuously monitors network traffic.
* It analyzes patterns, behaviors, and communication to detect any anomalies or suspicious activities.

**Early Detection and Alerts:**

* If an attack or unusual behavior occurs, our model promptly identifies it.
* Alerts are triggered, notifying administrators or users about potential threats.

**Swift Resolution:**

* Armed with insights from our model, security teams can take immediate action.
* Whether it’s blocking an IP address, isolating a compromised device, or investigating further, the response is swift.

**Holistic Security:**

* Our model complements existing security measures, providing an additional layer of defense.
* It works alongside firewalls, intrusion detection systems, and other security components.

**9. CONCLUSION**

In conclusion , my project focused on network intrusion detection using the UNSW-NB15 dataset, which contains network packet data. I aimed to create a machine learning model capable of predicting whether a network is under attack and classifying the specific type of attack. To achieve this, I conducted a thorough exploratory data analysis (EDA), performed feature engineering and selection, and worked with two target variables: one indicating network attack presence and the other specifying the type of attack encountered.

Through cross-validation, I evaluated multiple models, ultimately finding that the Random Forest algorithm performed well with high accuracy. This model effectively classifies attacks across various networks. Additionally, I outlined potential avenues for further model development.

Finally, the trained model is saved as a pickle file, which can be used for attack prediction and future enhancements. It’s essential to recognize that model accuracy relies on both the quality and quantity of training data. Therefore, utilizing a large and diverse dataset is recommended to enhance model performance.

**10. BIBILOGRAPHY**

**1.** Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)."Military Communications and Information Systems Conference (MilCIS), 2015. IEEE, 2015.

**2.**  Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set." Information Security Journal: A Global Perspective (2016): 1-14.

**3.** Moustafa, Nour, et al. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." IEEE Transactions on Big Data (2017).

**4.** Moustafa, Nour, et al. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." Data Analytics and Decision Support for Cybersecurity. Springer, Cham, 2017. 127-156.

**5.** Sarhan, Mohanad, Siamak Layeghy, Nour Moustafa, and Marius Portmann. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. In Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings (p. 117). Springer Nature.

**6.** Shyu, M.-L., Sarinnapakorn, K., Kuruppu-Appuhamilage, I.,Chen, S.-C., Chang, L., & Goldring, T. (2005). Handlingnominal features in anomaly intrusion detection problems.15th International Workshop on Research Issues in DataEngineering: Stream Data Mining and Applications, 2005.RIDE-SDMA 2005. IEEE.

**7.** So-In, C., Mongkonchai, N., Aimtongkham, P., Wijitsopon,K., & Rujirakul, K. (2014). An evaluation of data miningclassification models for network intrusion detection. 2014Fourth International Conference on Digital Informationand Communication Technology and its Applications(DICTAP), IEEE.

**8.** Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006). Beyondaccuracy, F-score and ROC: A family of discriminantmeasures for performance evaluation. In AI 2006:Advances in artificial intelligence (Vol. 4304, pp.1015–1021). Lecture Notes in Computer Science. Berlin,Germany: Springer.SPSS tool. (2014). Retrieved from <http://www-01.ibm.com/software/analytics/spss/>

**9.** Tavallaee, M., (2009). A detailed analysis of the KDD CUP 99data set. In Proceedings of the Second IEEE Symposium onComputational Intelligence for Security and DefenceApplications 2009 (pp. 53–58). Piscataway, NJ: IEEE.tcpdump tool.

R.Heady, G.Luger, A.Maccabe, M.Servilla. “The architecture of a

network level intrusion detection system”. Tech. rep., Computer Science

Department, University of New Mexico, New Mexico ,1990.

[2] M.Aydın, M. Ali, A. Halim Zaim, and K. Gökhan Ceylan. “A hybrid

intrusion detection system design for computer network

security”, Computers & Electrical Engineering, 2009, p 517-526.

[3] Axelsson, Stefan. “Intrusion detection systems: A survey and

taxonomy”, Technical report, 2000, Vol. 99.

R.Heady, G.Luger, A.Maccabe, M.Servilla. “The architecture of a

network level intrusion detection system”. Tech. rep., Computer Science

Department, University of New Mexico, New Mexico ,1990.

[2] M.Aydın, M. Ali, A. Halim Zaim, and K. Gökhan Ceylan. “A hybrid

intrusion detection system design for computer network

security”, Computers & Electrical Engineering, 2009, p 517-526.

[3] Axelsson, Stefan. “Intrusion detection systems: A survey and

taxonomy”, Technical report, 2000, Vol. 99.

R.Heady, G.Luger, A.Maccabe, M.Servilla. “The architecture of a

network level intrusion detection system”. Tech. rep., Computer Science

Department, University of New Mexico, New Mexico ,1990.

[2] M.Aydın, M. Ali, A. Halim Zaim, and K. Gökhan Ceylan. “A hybrid

intrusion detection system design for computer network

security”, Computers & Electrical Engineering, 2009, p 517-526.

[3] Axelsson, Stefan. “Intrusion detection systems: A survey and

taxonomy”, Technical report, 2000, Vol. 99.

**10.** R.Heady, G.Luger, A.Maccabe, M.Servilla. “The architecture of a network level intrusion detection system”. Tech. rep., Computer Science Department, University of New Mexico, New Mexico ,1990.

**11.** M.Aydın, M. Ali, A. Halim Zaim, and K. Gökhan Ceylan. “A hybrid intrusion detection system design for computer network security”, Computers & Electrical Engineering, 2009, p 517-526.

**12.** Axelsson, Stefan. “Intrusion detection systems: A survey and taxonomy”, Technical report, 2000, Vol. 99.

