Intelligent Traffic Classification Feature Engineering Technique (ITCFFE) For SDN Networks Based On Neural Networks

By

A.Arul Selvan Gnanamonickam
Research Scholar, Bharathiar University. Coimbatore, Tamilnadu, India
Email : aarulselvan1974@gmail.com

Dr.B.Paramasivan M.E.,Ph.D
Professor, Dept. of Information Technology, National Engg. College (An Autonomous Institution), Kovilpatti Thoothukudi, Tamilnadu, India
Email : bparamasivam@yahoo.co.in

Abstract

In recent years, the rise of Internet traffic has expanded explosively as a result of the quick increase in the number of Internet users. As a result of the exponential rise in Internet applications and their high computing costs, both port-based strategies and DPIs (deep packet inspections) are less effective. An integral component of the networking domain that transforms traditional networking into an automated network will be software defined networking. SDNs (Software Defined Networks) are used to centralize network architectures. The control planes and data planes have been separated as a result of SDNs. This has also resulted in the creation of a centralised network controller with comprehensive views of complete networks. Therefore, there is only one control plane (SDN controller) for all the switches in SDNs, in contrast to traditional networks where the two levels of control and data are linked together. Data planes are in charge of straightforward data packet forwarding while control planes do traffic routing. One key issue of this new networking architecture is security of data and identification of malicious packets. This paper uses the traffic information dataset of SDNs in an attempt to select most important features required for classifications of network packets into normal and malicious classes. The proposed scheme called ITCFFE (Intelligent Traffic Classification Feature Engineering Technique) is based on correlations between features and BFEs (Backward feature Eliminations) for dropping unwanted features while retaining the most important features for its final outcomes. The proposed ITCFFE schema is evaluated using multiple classifiers for its efficiency where classification accuracy of more than 95% is achieved.

Introduction

With more networks being utilised for different application demands including texts, photos, audios, and videos, network data traffic has grown. Additionally, as network component speeds have increased, internet traffic has also exploded. SDNs [1], which have the capacity to operate networks dynamically, have grown to be well-liked technology for handling this type of data. SDNs are far more adaptable than conventional networking since their control planes are built on software. They enable administrators to administer networks, change settings, and increase network capabilities using single interfaces without requiring the addition of additional hardware. Despite their flexibility, SDNs are vulnerable to security vulnerabilities. Since SDNs cover a wide range of network topologies, information about network traffic of packets going via SDNs hosts/switches is encountered by their controllers. This traffic data may be utilised to create sets of SDNs characteristics that can be examined or used to keep an eye out for malicious packets. Identification of network traffic patterns and assessments of the contents of the traffic data become crucial components for evaluating the security and safety of packet transfers within SDNs. The two most common methods for classifying traffic are deep packet inspections and port-based classifications. These methods are getting old as more communication is encrypted and as more apps use dynamic ports and ports for other well-known applications. [2]. MLTs (Machine Learning Techniques) are other methods for classifying traffic. In order to overcome fundamental problems with packet/port-based inspections for classifying encrypted flows, they can be utilised to investigate statistical features of network traffic flows. In actuality, accurate traffic classifications allow for efficient resource allocation and management across the network [3]. The SDNs architecture is shown in Figure 1.

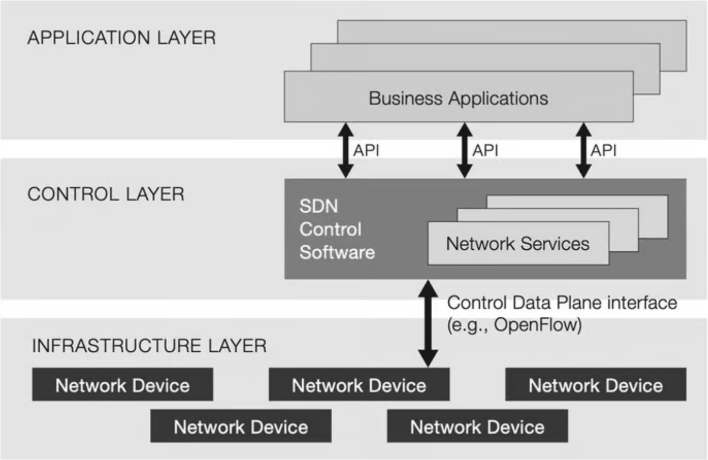


Fig. 1 – *Overview of SDNs*

Both before and after the development of MLTs, other strategies, including intelligence calculations, were used. Monitoring and traffic analysis are crucial for improving a network's security of service. SDNs resources are essential and must be applied to the targeted goals. Additionally, it is quite challenging to watch over and forecast data flows in SDNs for a number of different reasons. A variety of human and artificial intelligence-based procedures and techniques have been used in previous research to forecast data. Using DMTs (Data Mining Techniques), a clever university programme forecasted daily internet traffic [4]. Internet data flow was predicted by the study [5] using MLTs, whereas the study [6] employed double exponential predictors based on ANNs (artificial neural networks). The projected network traffic was also found in [7]. Consequently, even with the use of high level abstraction languages based on reactive programming, defining security policies that take into account various scenarios and applications running on the network can be a daunting task. SDNs hold the key to creating networks that can adapt effectively and efficiently to ever-changing conditions. MLTs can be used to identify and defend against attacks on SDNs. Therefore, the primary goal of this work is to extract and identify basic SDNs traffic properties that classifiers can utilise to quickly detect malicious packets. The following are the portions of this essay: The suggested approach for classifying internet traffic in SDNs is presented in Section 3. The execution and performance assessment of the suggested technique are provided in Section 3. Finally, discuss and analyze the results in Section 4

Review of Literature

This section provides information about SDN literature and feature choices. In order to discern application protocols in runtime, Hanigan et al. [8] employed traffic classification techniques based on DPIs to analyse flows via SDNs. Their objective is to make it possible for controllers to recognise and separate various application flows while controlling and programming flows to provide QoS (Quality of Services) for delay-sensitive applications. When the network is busy, a significant portion of the controllers' processing power must go toward the DPIs' tools. As a result, the network's overall performance is impacted. A methodology to identify the application type of current flows in a wireless network, which consists of several mobile devices linked to OFSs (Open Flow switches) was proposed by Arsalan et al. [9]. Trainer based MLTs gets the data in control planes. The OFS switch, on the other hand, collects the characteristics of various flows and transmits them to the control layer so that a model for application layer identification may be developed. When a host enters the network after the model has been created, the OFSs communicate device’s flow attributes to traffic categorization models based on MLTs. Jang et al. [10] suggested using a dataset of flows attributes as input to the K-means algorithm during the learning phase of clustering. In order to create a traffic categorization model, these clusters are used. Based on the information gleaned from the packet content, the clusters with related traits are combined. Despite having an 89% accuracy rate, this technology eliminates the necessity to accurately diagnose encrypted packets and investigate packet contents. Most studies base their traffic classification on a collection of statistics from offline, stored flows. The categorization of internet traffic faces two challenges: a high temporal complexity and processing overhead. Additionally, current methods place a tremendous burden on the system. This study aims to categorise traffic over SDNs using statistics in the controller and data from packet headers received from OF switches. A framework for online traffic categorization based on application layer protocol is proposed by taking into account protocol capabilities on collecting flows data and neural network variants such as Feed Forwards, MLPs (Multi-Layer Perceptrons), and NBs (Nave Bayes). The accuracy of the suggested approach is 97.6%, compared to the preceding methods' best accuracy of 94% [11]. Low runtime execution, minimal network overhead, and low processor overhead are benefits of this approach over existing ones. The concept of flow-based anomaly detection has recently gained interest. We will examine similar work that has already been completed in this part. SDNs architecture and MLTs have been the subject of several research in the past. Based on MLPs, a flow-based anomaly detection architecture was put forth [12], and a gravitational search method was researched. In order to distinguish between regular and pathological network traffic, a model was created. The Support Vector Machines (SVMs) used by NIDS (Network Intrusion Detection Systems) after this were more accurate and had a lower false alarm rate [13On the NOX controller and Open Flow switches, a model was created [14]. TRWs (Threshold Random Walks), maximum entropies detectors, rate limiting, and other four novel techniques for anomaly identification were introduced. All of these were effective in finding network irregularities. Attack models based on flow characteristics were first described in DDoS (Distributed Denial of Services) [15]. This approach made use of the idea of a self-organizing map to find anomalies. An SVMs classifier based on DDoS assaults was presented, and the results showed very low false positive alert rates [16]. A classifier based on an improved protection method for SVMs was developed [17]. Six characteristics were used to model DNNs (Deep Neural Networks) based on anomaly detection systems in [18]. This model successfully detected abnormalities with a high degree of accuracy. A new model was put out [19] to improve detection accuracy and to achieve high levels of performance. To evaluate the performance of the suggested model, it was contrasted with alternative models. [20] also offered a risk assessment system to determine the effects of multi-stage assaults beforehand. A two-layer and three-layer intrusion detection scheme that was mostly compatible with wireless sensor networks was developed in [21]. A model was developed in [22] to understand the effects of application DDoS assaults and to examine different server characteristics. This model was created primarily to understand attack severity and server performance under various assaults. To stop unwanted access to data in [23], a method for database intrusion detection was created. Different ranker algorithms were created. Information Gains, Relief-F

3. Proposed Methodology

One of the most critical aspects of network administrations are traffic categorizations which are executed using one of the three techniques namely port based classifications, DPIs and techniques based on AIs. Ports based systems, are simple and fast but may be easily manipulated and are less trustworthy. DPIs produce good results; however they can only be utilized for unencrypted traffics and fail while encountering real-time encrypted data/traffics. The primary goal of this work is the application of AIs to detect malicious packets early in traffic flows. The proposed ITCFFE schema is built on DLTs to discover the optimal characteristics of datasets which can be used by classifiers to identify malicious packets. ITCFFE's methodology follows the stages of data preparations, feature preparations/bifurcations. The selected features sets are classified for evaluations using RFs (random Forests), GNBs (Gaussian Naïve Bayes) and DTs (decision Trees). The proposed ITCFFE's selected feature sets were used by the aforesaid classifiers and evaluated in terms of training and testing accuracies.

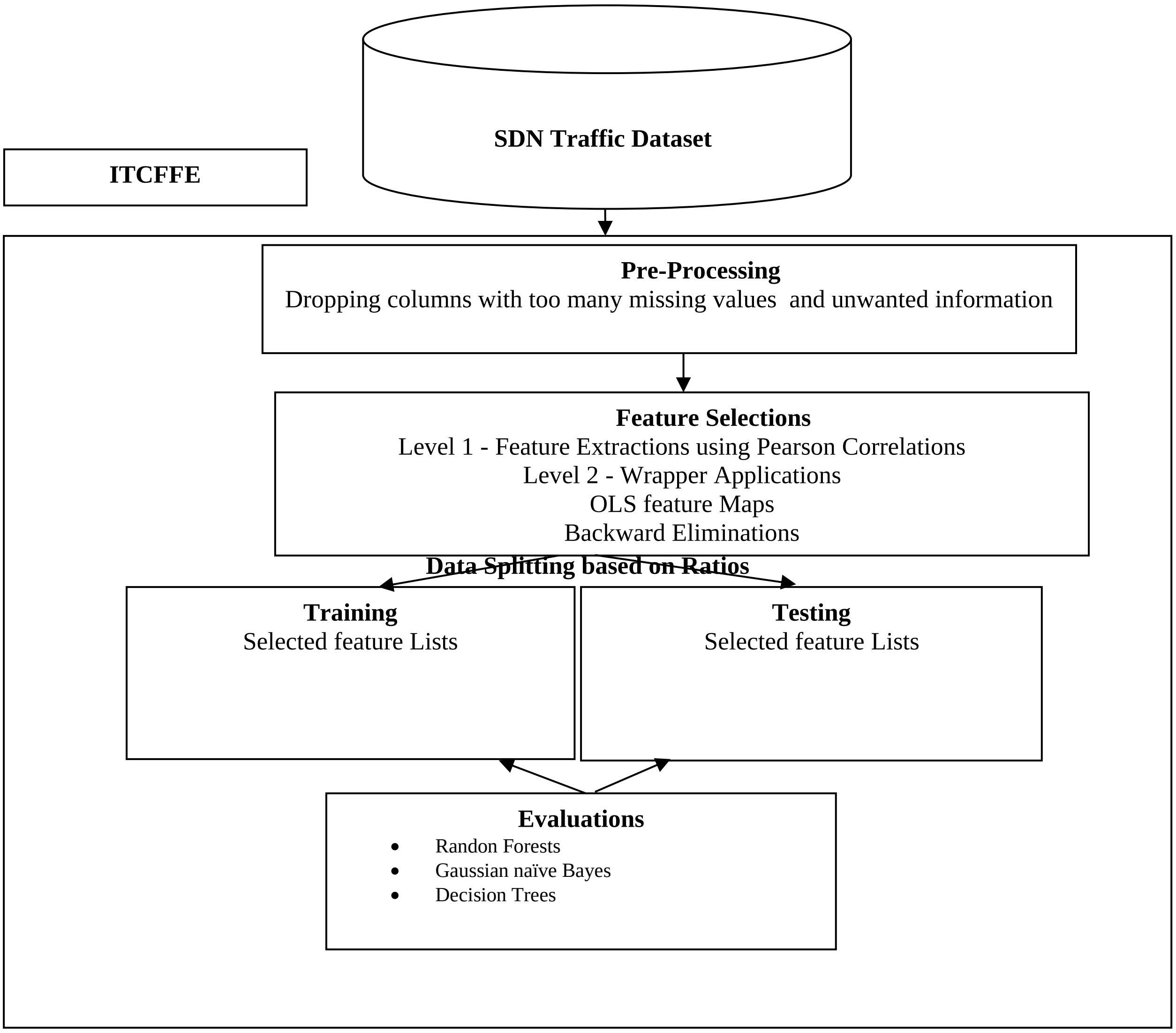


Fig. 2 – *Architecture of ITCFFE*

ITCFFE – Data preparation

This is a preliminary step for DMTs (data mining techniques) that transform raw data into a more understandable, useful and efficient formats. After completing these fundamental stages, the concept of data will be clearer and more understood. Real-world data are typically incomplete because they include unnecessary numbers, missing values, or aggregate data. Errors in data are also possible. In data pre-processing, they are normalised by filling missing values, smoothing or eliminating noisy data and outliers, and resolving discrepancies.

Handling Missing values

There are several approaches to dealing with missing data, including disregarding data rows. This strategy is recommended for records in which the majority of the data is absent, leaving records worthless. When only a few attribute values are missing, this procedure is typically avoided. Poor performances fall below level when missing values are not disregarded or eliminated. These are time-consuming strategies when done manually and thus unsuitable for practically all cases.

Remove Unwanted Data

Unwanted data is data that is duplicated or useless. Scraping data from many sources and then merging it may result in some redundant data if not done properly. This duplicate data should be eliminated because it is useless and will just add to the quantity of data and time required to train the model.

ITCFFE Feature Selections

ITCFFE's feature selections are executed in two levels. In Level 1, Feature Extractions are executed using Pearson Correlation while at level 2 Wrapper Applications are used for identifying minimal features sets from datasets that represent complete data and enhance classification accuracy. Level 2 uses OLSs (Ordinary Least Squares) and BEs (Backward Eliminations).

ITCFFE feature Selections Level 1

Voluminous data needs to be dealt very consciously for proper outcomes to achieve approaches. MLTs or DLTs provide single outputs from huge amounts of data be it structured or unstructured. At varied coefficients and degrees, these components may contribute to the needed results. They must be filtered out in various ways based on their importance in deciding outputs, as well as taking into account redundancy in these components. There are output variables for every n input variables in supervised learning. Correlations are statistical metrics that show how much two or more variables fluctuate in tandem. In layman's words, it informs us how much one variable varies when another variable changes little. It can have positive, negative, or zero values depending on the direction of the shift. Pearson correlations can be used to find correlations between any two variables x,y may be used to determine degree of associations between linearly related variables and given as Equation (1).

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\,\sqrt{n\sum y_i^2 - (\sum y_i)^2}} \quad \text{Equation (1)}$$

Where n stands for counts of observations while i denotes ith observation

This work uses Dummy variables which are qualitative or discrete variables that reflect category data and can have values of 0 or 1 to indicate the lack or existence of a given property. Indicator variables, design variables, and binary basis variables are other names for dummy variables. They are commonly employed in time-series analysis, seasonal component analysis, linear regression models, and a variety of other applications where qualitative data is prevalent. In general, any regression analysis' explanatory or independent variables are expected to be quantitative in character. Variables such as temperature, distance, age, and so on are quantitative in the sense that they are measured on a well-defined scale.

ITCFFE feature Selections Level 2

Current datasets have extremely high dimensions and in order process them using MLTs, feature selections are pertinent. Variables may be irrelevant or less meaningful to dependent variables, which increases complexities, making it difficult for models to comprehend during training ed and resulting in erroneous or less dependable predictions. Feature selection processes in wrapper methods are based on certain MLTs which attempt to fit themselves to given datasets. they employ greedy search strategies, weighing all potential feature combinations against evaluation criteria which are performance measures. For e.g. the assessment criterion for regressions can be p, R^2, Adjusted R^2 while for classifications it could be accuracies, precisions, recalls and f1-scores. Finally, feature combinations that produce best results are selected by MLTs. Forward selections, BEs, and bi-directional eliminations are the most widely utilized wrapper methods (Stepwise Selection). BEs and wrappers are used in this work.

ITCFFE – Feature Bifurcations

MLTs are a burgeoning technology that allows computers/machines to transform massive amounts of data into predictions. These predictions, however, are very dependent on the quality of the data, and if we do not use the correct data for our model, it will not provide the predicted outcome. In most machine learning projects, the original dataset is divided into training and test data. Models are trained on subsets of original datasets, called the training datasets, and then examined for their generalizations on new or previously unknown datasets or test sets. As a result, train and test datasets are two fundamental ideas in machine learning, with the training dataset used to fit the model which is evaluated using the test dataset. The training data is the largest (in terms of size) subset of the original dataset used to train or fit the machine learning model. To begin, the training data differs depending on whether we are using Supervised Learning or Unsupervised Learning Algorithms. The training data for unsupervised learning comprises unlabeled data points, which means that the inputs are not tagged with the matching outputs. In order to produce predictions, models must detect patterns in the supplied training datasets. In contrast, the training data for supervised learning comprises labels in order to train the model and generate predictions. The type of training data we supply to the model has a significant impact on the model's accuracy and prediction capabilities. That is, the greater the quality of the training data, the better the model's performance. Training data accounts for around 60% of total data for an ML project. After we've trained the model with the training dataset, it's time to put it to the test. This dataset examines the model's performance and guarantees that the model can generalize effectively to new or unknown datasets. The test dataset is a separate subset of the original data from the training dataset. They however, include some comparable sorts of features and class probability distributions and utilize it as a baseline for model evaluation once model training is complete. A well-organized dataset including data for each sort of scenario for a specific problem that the model would face if employed in the actual world is referred to as test data. The test dataset for an ML project is typically 20-25% of the entire source data. Figure 3 explains the preceding processes:

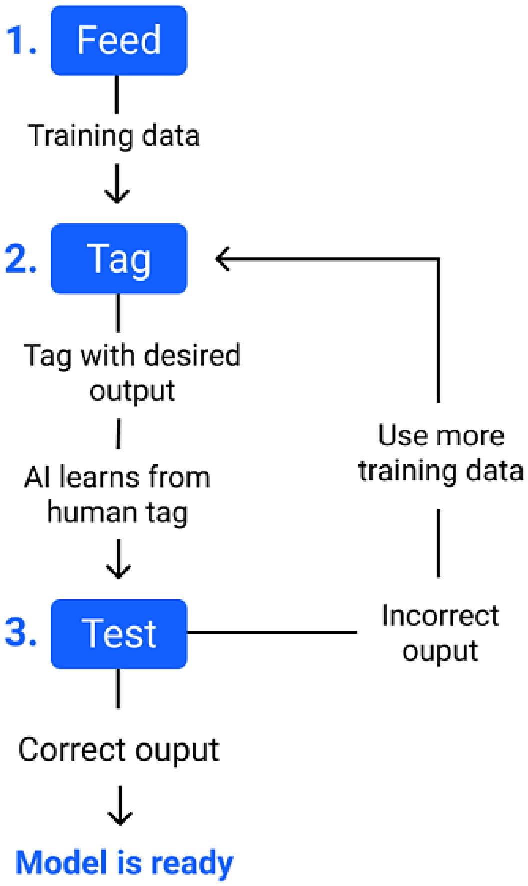
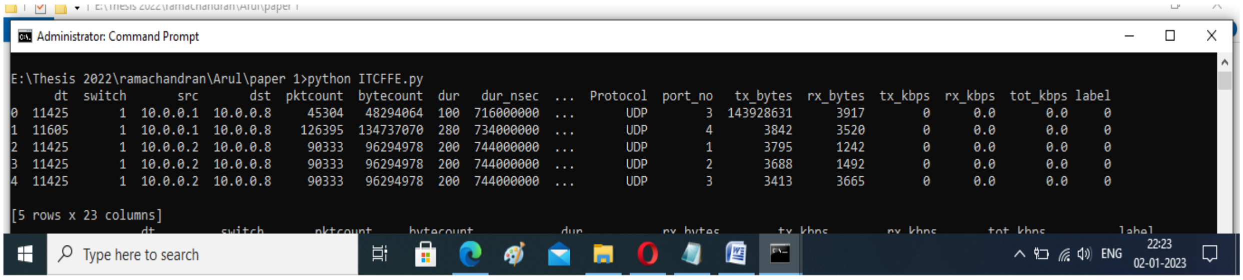


Fig. 3 – *ITCFFE – Feature Bifurcations (flowchart for Training/testing of Models)*

Results and Discussions

This section displays stage wise experimental results of the proposed scheme executed using Python 3.9 on an AMD athelon processor with 4 GB memory. Python 3.7.5 was used for implementations. The experiments were coded for Traffic dataset of SDNs obtained from Kaggle. The necessary features were obtained from SDN's traffic dataset. The dataset included the following features: dt (date), switch (switch no), src (packet's source IP), dst (packet's destination IP), pktcount (counts of packets), bytecount (counts of bytes), dur (data flow durations), dur_nsec (data flow durations in nano seconds), tot_dur (total flow dutations), flows (counts of flows), packetins (Input Packets from Devices), pktperflow (counts of packets in flows), byteperflow (counts of bytes in flows), pktrate (rates at which packets arrive), Pairflow (flows in pairs), Protocol (UDP, ICMP, TCP), port_no (port nos), tx_bytes (counts of bytes sent by functions), rx_bytes (counts of bytes received by functions) and tot_kbps (data flows in terms of total kilobytes per second). Figure 4 shows a snapshot of the SDN traffic data set.

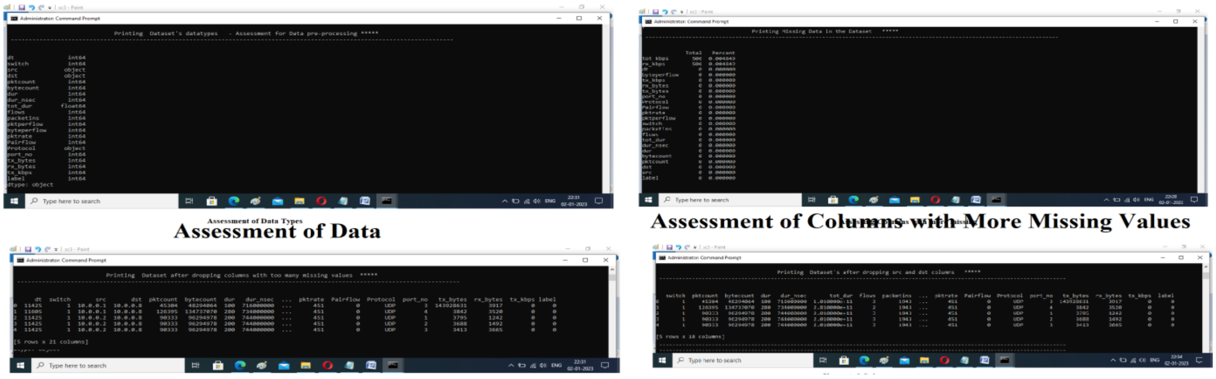


```
E:\Thesis 2022\ramachandran\Arul\paper 1>python ITCFFE.py
      dt  switch       src       dst  pktcount  bytecount  dur   dur_nsec  ...  Protocol  port_no   tx_bytes  rx_bytes  tx_kbps  rx_kbps  tot_kbps label
0  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  ...       UDP        3  143928631      3917        0      0.0       0.0     0
1  11605       1  10.0.0.1  10.0.0.8    126395  134737070  280  734000000  ...       UDP        4       3842      3520        0      0.0       0.0     0
2  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...       UDP        1       3795      1242        0      0.0       0.0     0
3  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...       UDP        2       3688      1492        0      0.0       0.0     0
4  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...       UDP        3       3413      3665        0      0.0       0.0     0

[5 rows x 23 columns]
              dt        switch       pktcount     bytecount          dur           rx_bytes        tx_kbps        rx_kbps        tot_kbps          label
```

Fig. 4 – *Snapshot of Traffic Dataset*

ITCFFE – Data preparation

In a data set, missing values cannot be examined. They must be dealt with. Furthermore, many models do not allow missing values. There are various ways for dealing with missing data; selecting the best one is critical. The approach used to deal with missing data is determined by the issue domain and the purpose of the data mining process. To fill in missing values, global constants like "NA" or 0 can be used. When missing values are difficult to anticipate, this strategy is utilised. Furthermore, due to redundant records, the model may not produce correct findings since the duplicate data interferes with the analysis process, giving more weight to the repeated values. Use attribute mean or median: The attribute's mean or median is utilised to fill in the missing value. ITCFFE handles Missing Values by normalizing them and removing unwanted data. It also drops columns where there are too many missing values. Figures 5 depict the output of ITCFFE data Preparation.



```
                    Printing  Dataset's datatypes   - Assessment for Data pre-processing *****
---------------------------------------------------------------------------------------------

dt                int64
switch            int64
src              object
dst              object
pktcount          int64
bytecount         int64
dur               int64
dur_nsec          int64
tot_dur         float64
flows             int64
packetins         int64
pktperflow        int64
byteperflow       int64
pktrate           int64
Pairflow          int64
Protocol         object
port_no           int64
tx_bytes          int64
rx_bytes          int64
tx_kbps           int64
label             int64
dtype: object
```

Assessment of Data Types
Assessment of Data

```
                         Printing Missing Data in the Dataset   *****
---------------------------------------------------------------------

              Total   Percent
tot_kbps        506  0.004842
rx_kbps         506  0.004842
dt                0  0.000000
byteperflow       0  0.000000
tx_kbps           0  0.000000
rx_bytes          0  0.000000
tx_bytes          0  0.000000
port_no           0  0.000000
Protocol          0  0.000000
Pairflow          0  0.000000
pktrate           0  0.000000
pktperflow        0  0.000000
switch            0  0.000000
packetins         0  0.000000
flows             0  0.000000
tot_dur           0  0.000000
dur_nsec          0  0.000000
dur               0  0.000000
bytecount         0  0.000000
pktcount          0  0.000000
dst               0  0.000000
src               0  0.000000
label             0  0.000000
```

Assessment of Columns with More Missing Values

```
                    Printing  Dataset after dropping columns with too many missing values  *****
------------------------------------------------------------------------------------------------

      dt  switch       src       dst  pktcount  bytecount  dur   dur_nsec  ...  pktrate  Pairflow  Protocol  port_no   tx_bytes  rx_bytes  tx_kbps label
0  11425       1  10.0.0.1  10.0.0.8     45304   48294064  100  716000000  ...      451         0       UDP        3  143928631      3917        0     0
1  11605       1  10.0.0.1  10.0.0.8    126395  134737070  280  734000000  ...      451         0       UDP        4       3842      3520        0     0
2  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...      451         0       UDP        1       3795      1242        0     0
3  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...      451         0       UDP        2       3688      1492        0     0
4  11425       1  10.0.0.2  10.0.0.8     90333   96294978  200  744000000  ...      451         0       UDP        3       3413      3665        0     0

[5 rows x 21 columns]
```

Missing Valu Corrections

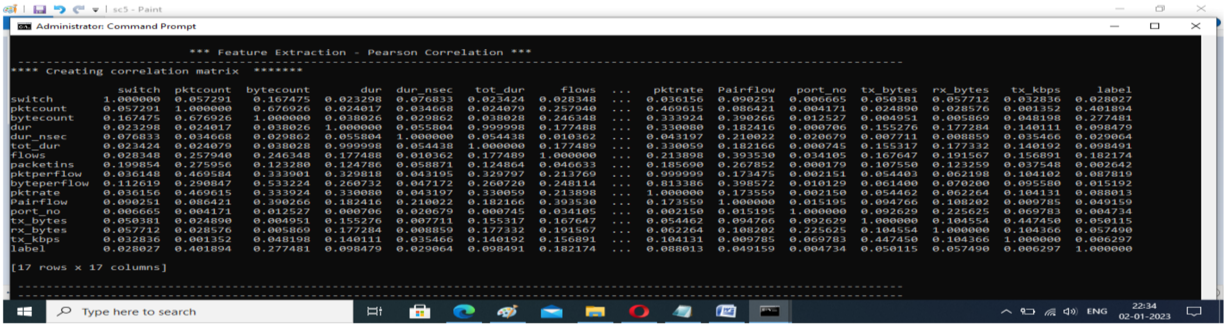
```
                    Printing  Dataset's after dropping src and dst columns   *****
----------------------------------------------------------------------------------

   switch  pktcount  bytecount  dur   dur_nsec       tot_dur  flows  packetins  ...  pktrate  Pairflow  Protocol  port_no   tx_bytes  rx_bytes  tx_kbps  label
0       1     45304   48294064  100  716000000  1.010000e+11      2       1943  ...      451         0       UDP        3  143928631      3917        0      0
1       1    126395  134737070  280  734000000  2.810000e+11      2       1943  ...      451         0       UDP        4       3842      3520        0      0
2       1     90333   96294978  200  744000000  2.010000e+11      3       1943  ...      451         0       UDP        1       3795      1242        0      0
3       1     90333   96294978  200  744000000  2.010000e+11      3       1943  ...      451         0       UDP        2       3688      1492        0      0
4       1     90333   96294978  200  744000000  2.010000e+11      3       1943  ...      451         0       UDP        3       3413      3665        0      0

[5 rows x 18 columns]
----------------------------------------------------------------------------------
----------------------------------------------------------------------------------
```

Dropping Unwanted Columns

Fig. 5 - *ITCFFE data Preparation Outputs*

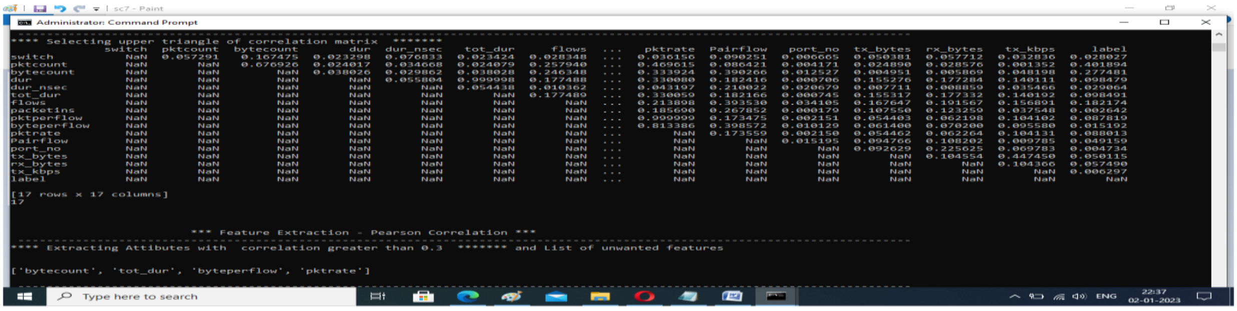
ITCFFE Feature Selections

In multiple regression settings with numerous components, to create more feasible models with greater accuracies, it is critical to establish correlations between all dependent and independent variables. High correlation values between dependent variables and independent factors were suggested in this work based on significant impacts of independent variables on outputs. It should also be noted that more characteristics do not guarantee greater accuracy and in fact may reduce accuracies when they contain irrelevant features that create unnecessary noises in models. Hence, ITCFFE Feature Selections use person's correlations to determine the optimal required features from datasets. Dummy variables are widely used in Data Science and Machine Learning due to the qualitative nature of dependent and independent variables. Qualitative includes categorical variables which mean variables can be classified into different categories. Numeric variables can also be dummy coded to explore nonlinear effects. Figures (6) to (8) display stage wise outputs of **ITCFFE** Feature Selections.



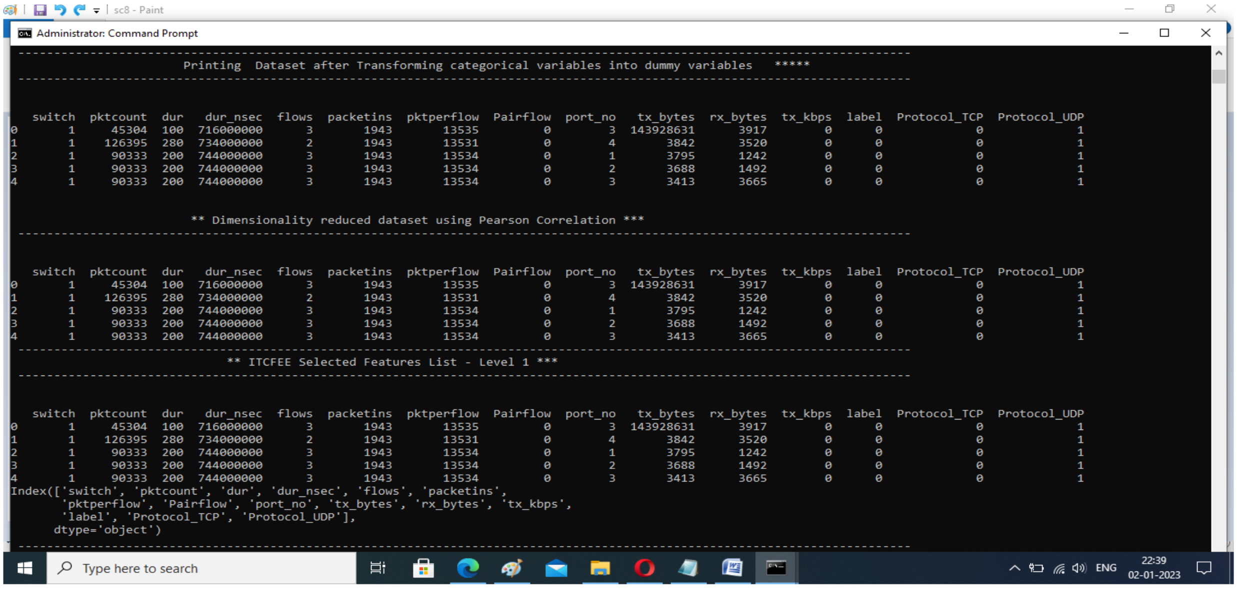
```
                  *** Feature Extraction - Pearson Correlation ***
-----------------------------------------------------------------------------------------------------------------------
**** Creating correlation matrix  *******

               switch  pktcount  bytecount       dur  dur_nsec   tot_dur     flows  ...   pktrate  Pairflow   port_no  tx_bytes  rx_bytes   tx_kbps     label
switch       1.000000  0.057291   0.167475  0.023298  0.076833  0.023424  0.028348  ...  0.036156  0.090251  0.006665  0.050381  0.057712  0.032836  0.028027
pktcount     0.057291  1.000000   0.676926  0.024017  0.034668  0.024079  0.257940  ...  0.469615  0.086421  0.004171  0.024890  0.028576  0.001352  0.401894
bytecount    0.167475  0.676926   1.000000  0.038026  0.029862  0.038028  0.246348  ...  0.333924  0.390266  0.012527  0.004951  0.005869  0.048198  0.277481
dur          0.023298  0.024017   0.038026  1.000000  0.055804  0.999998  0.177488  ...  0.330080  0.182416  0.000706  0.155276  0.177284  0.140111  0.098479
dur_nsec     0.076833  0.034668   0.029862  0.055804  1.000000  0.054438  0.010362  ...  0.043197  0.210022  0.020679  0.007711  0.008859  0.035466  0.029064
tot_dur      0.023424  0.024079   0.038028  0.999998  0.054438  1.000000  0.177489  ...  0.330059  0.182166  0.000745  0.155317  0.177332  0.140192  0.098491
flows        0.028348  0.257940   0.246348  0.177488  0.010362  0.177489  1.000000  ...  0.213898  0.393530  0.034105  0.167647  0.191567  0.156891  0.182174
packetins    0.199854  0.275956   0.123280  0.124786  0.058871  0.124864  0.046633  ...  0.185690  0.267852  0.000179  0.107550  0.123259  0.037548  0.002642
pktperflow   0.036148  0.469584   0.333901  0.329818  0.043195  0.329797  0.213769  ...  0.999999  0.173475  0.002151  0.054403  0.062198  0.104102  0.087819
byteperflow  0.112619  0.290847   0.533224  0.260732  0.047172  0.260720  0.248114  ...  0.813386  0.398572  0.010129  0.061400  0.070200  0.095580  0.015192
pktrate      0.036156  0.469615   0.333924  0.330080  0.043197  0.330059  0.213898  ...  1.000000  0.173559  0.002150  0.054462  0.062264  0.104131  0.088013
Pairflow     0.090251  0.086421   0.390266  0.182416  0.210022  0.182166  0.393530  ...  0.173559  1.000000  0.015195  0.094766  0.108202  0.009785  0.049159
port_no      0.006665  0.004171   0.012527  0.000706  0.020679  0.000745  0.034105  ...  0.002150  0.015195  1.000000  0.092629  0.225625  0.069783  0.004734
tx_bytes     0.050381  0.024890   0.004951  0.155276  0.007711  0.155317  0.167647  ...  0.054462  0.094766  0.092629  1.000000  0.104554  0.447450  0.050115
rx_bytes     0.057712  0.028576   0.005869  0.177284  0.008859  0.177332  0.191567  ...  0.062264  0.108202  0.225625  0.104554  1.000000  0.104366  0.057490
tx_kbps      0.032836  0.001352   0.048198  0.140111  0.035466  0.140192  0.156891  ...  0.104131  0.009785  0.069783  0.447450  0.104366  1.000000  0.006297
label        0.028027  0.401894   0.277481  0.098479  0.029064  0.098491  0.182174  ...  0.088013  0.049159  0.004734  0.050115  0.057490  0.006297  1.000000

[17 rows x 17 columns]

-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
```

Fig. 6 – *ITCFFE 's Person Correlations Output*



```
**** Selecting upper triangle of correlation matrix  *******
               switch  pktcount  bytecount       dur  dur_nsec   tot_dur     flows  ...   pktrate  Pairflow   port_no  tx_bytes  rx_bytes   tx_kbps     label
switch            NaN  0.057291   0.167475  0.023298  0.076833  0.023424  0.028348  ...  0.036156  0.090251  0.006665  0.050381  0.057712  0.032836  0.028027
pktcount          NaN       NaN   0.676926  0.024017  0.034668  0.024079  0.257940  ...  0.469615  0.086421  0.004171  0.024890  0.028576  0.001352  0.401894
bytecount         NaN       NaN        NaN  0.038026  0.029862  0.038028  0.246348  ...  0.333924  0.390266  0.012527  0.004951  0.005869  0.048198  0.277481
dur               NaN       NaN        NaN       NaN  0.055804  0.999998  0.177488  ...  0.330080  0.182416  0.000706  0.155276  0.177284  0.140111  0.098479
dur_nsec          NaN       NaN        NaN       NaN       NaN  0.054438  0.010362  ...  0.043197  0.210022  0.020679  0.007711  0.008859  0.035466  0.029064
tot_dur           NaN       NaN        NaN       NaN       NaN       NaN  0.177489  ...  0.330059  0.182166  0.000745  0.155317  0.177332  0.140192  0.098491
flows             NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...  0.213898  0.393530  0.034105  0.167647  0.191567  0.156891  0.182174
packetins         NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...  0.185690  0.267852  0.000179  0.107550  0.123259  0.037548  0.002642
pktperflow        NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...  0.999999  0.173475  0.002151  0.054403  0.062198  0.104102  0.087819
byteperflow       NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...  0.813386  0.398572  0.010129  0.061400  0.070200  0.095580  0.015192
pktrate           NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN  0.173559  0.002150  0.054462  0.062264  0.104131  0.088013
Pairflow          NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN  0.015195  0.094766  0.108202  0.009785  0.049159
port_no           NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN  0.092629  0.225625  0.069783  0.004734
tx_bytes          NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN       NaN  0.104554  0.447450  0.050115
rx_bytes          NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN       NaN       NaN  0.104366  0.057490
tx_kbps           NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN       NaN       NaN       NaN  0.006297
label             NaN       NaN        NaN       NaN       NaN       NaN       NaN  ...       NaN       NaN       NaN       NaN       NaN       NaN       NaN

[17 rows x 17 columns]
17


                    *** Feature Extraction - Pearson Correlation ***
-----------------------------------------------------------------------------------------
**** Extracting Attibutes with  correlation greater than 0.3  ******* and List of unwanted features

['bytecount', 'tot_dur', 'byteperflow', 'pktrate']
```

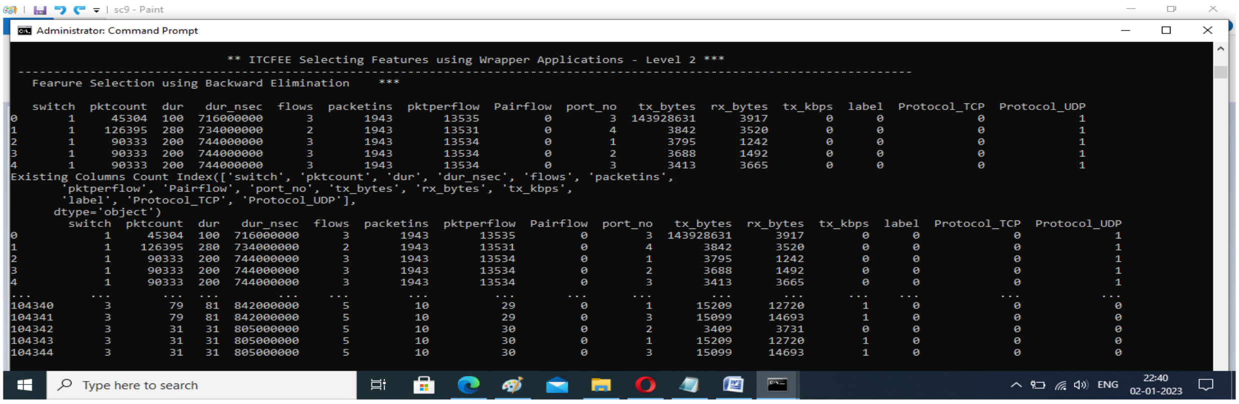
Fig. 7 – *Person Correlation's Upper Triangle Matrix*



```
-----------------------------------------------------------------------------------------
                  Printing  Dataset after Transforming categorical variables into dummy variables   *****
-----------------------------------------------------------------------------------------

   switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no   tx_bytes  rx_bytes  tx_kbps  label  Protocol_TCP  Protocol_UDP
0       1     45304  100  716000000      3       1943       13535         0        3  143928631      3917        0      0             0             1
1       1    126395  280  734000000      2       1943       13531         0        4       3842      3520        0      0             0             1
2       1     90333  200  744000000      3       1943       13534         0        1       3795      1242        0      0             0             1
3       1     90333  200  744000000      3       1943       13534         0        2       3688      1492        0      0             0             1
4       1     90333  200  744000000      3       1943       13534         0        3       3413      3665        0      0             0             1

                ** Dimensionality reduced dataset using Pearson Correlation ***
-----------------------------------------------------------------------------------------

   switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no   tx_bytes  rx_bytes  tx_kbps  label  Protocol_TCP  Protocol_UDP
0       1     45304  100  716000000      3       1943       13535         0        3  143928631      3917        0      0             0             1
1       1    126395  280  734000000      2       1943       13531         0        4       3842      3520        0      0             0             1
2       1     90333  200  744000000      3       1943       13534         0        1       3795      1242        0      0             0             1
3       1     90333  200  744000000      3       1943       13534         0        2       3688      1492        0      0             0             1
4       1     90333  200  744000000      3       1943       13534         0        3       3413      3665        0      0             0             1
-----------------------------------------------------------------------------------------
                     ** ITCFEE Selected Features List - Level 1 ***
-----------------------------------------------------------------------------------------

   switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no   tx_bytes  rx_bytes  tx_kbps  label  Protocol_TCP  Protocol_UDP
0       1     45304  100  716000000      3       1943       13535         0        3  143928631      3917        0      0             0             1
1       1    126395  280  734000000      2       1943       13531         0        4       3842      3520        0      0             0             1
2       1     90333  200  744000000      3       1943       13534         0        1       3795      1242        0      0             0             1
3       1     90333  200  744000000      3       1943       13534         0        2       3688      1492        0      0             0             1
4       1     90333  200  744000000      3       1943       13534         0        3       3413      3665        0      0             0             1
Index(['switch', 'pktcount', 'dur', 'dur_nsec', 'flows', 'packetins',
       'pktperflow', 'Pairflow', 'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps',
       'label', 'Protocol_TCP', 'Protocol_UDP'],
      dtype='object')
-----------------------------------------------------------------------------------------
```

Fig 8 – *ITCFFE's Level 1 feature extraction Output*

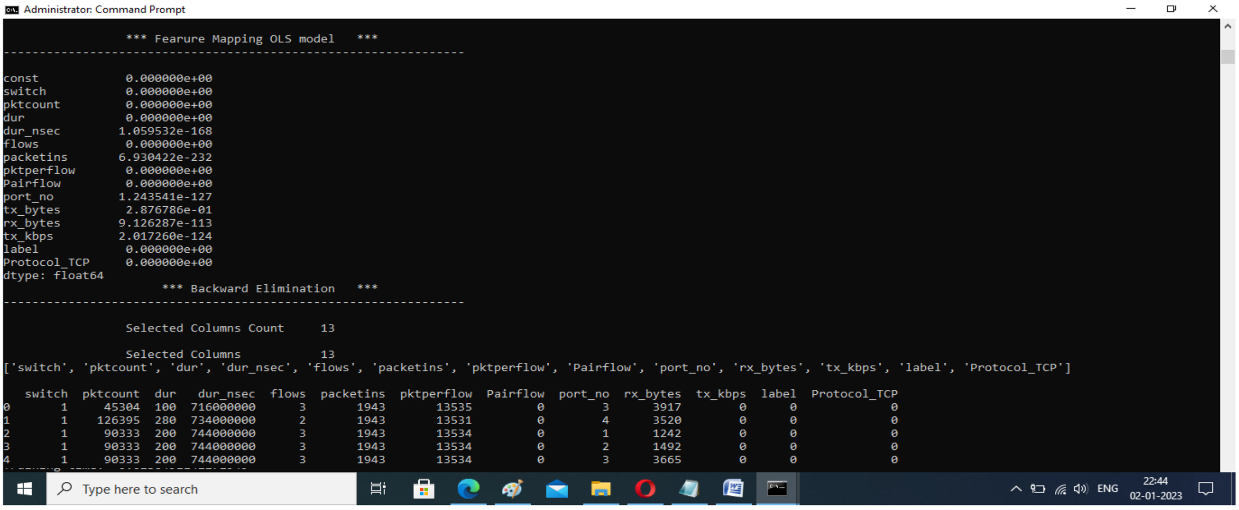
BEs are superior strategies in huge collections of potential features and are more efficient with their feature selections. ITCFFE uses BEs. They are methodical strategies that begin with full sets of characteristics and remove them one by one until model's performances achieve a peak. These approaches are computationally efficient, but may not yield the best selections of characteristics. Figure 9 depicts ITCFFE's wrapper output.



```
                     ** ITCFEE Selecting Features using Wrapper Applications - Level 2 ***
-----------------------------------------------------------------------------------------
   Fearure Selection using Backward Elimination    ***

   switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no   tx_bytes  rx_bytes  tx_kbps  label  Protocol_TCP  Protocol_UDP
0       1     45304  100  716000000      3       1943       13535         0        3  143928631      3917        0      0             0             1
1       1    126395  280  734000000      2       1943       13531         0        4       3842      3520        0      0             0             1
2       1     90333  200  744000000      3       1943       13534         0        1       3795      1242        0      0             0             1
3       1     90333  200  744000000      3       1943       13534         0        2       3688      1492        0      0             0             1
4       1     90333  200  744000000      3       1943       13534         0        3       3413      3665        0      0             0             1
Existing Columns Count Index(['switch', 'pktcount', 'dur', 'dur_nsec', 'flows', 'packetins',
       'pktperflow', 'Pairflow', 'port_no', 'tx_bytes', 'rx_bytes', 'tx_kbps',
       'label', 'Protocol_TCP', 'Protocol_UDP'],
      dtype='object')
        switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no   tx_bytes  rx_bytes  tx_kbps  label  Protocol_TCP  Protocol_UDP
0            1     45304  100  716000000      3       1943       13535         0        3  143928631      3917        0      0             0             1
1            1    126395  280  734000000      2       1943       13531         0        4       3842      3520        0      0             0             1
2            1     90333  200  744000000      3       1943       13534         0        1       3795      1242        0      0             0             1
3            1     90333  200  744000000      3       1943       13534         0        2       3688      1492        0      0             0             1
4            1     90333  200  744000000      3       1943       13534         0        3       3413      3665        0      0             0             1
...        ...       ...  ...        ...    ...        ...         ...       ...      ...        ...       ...      ...    ...           ...           ...
104340       3        79   81  842000000      5         10          29         0        1      15209     12720        1      0             0             0
104341       3        79   81  842000000      5         10          29         0        3      15099     14693        1      0             0             0
104342       3        31   31  805000000      5         10          30         0        2       3409      3731        0      0             0             0
104343       3        31   31  805000000      5         10          30         0        1      15209     12720        1      0             0             0
104344       3        31   31  805000000      5         10          30         0        3      15099     14693        1      0             0             0
```

Fig. 9 - *ITCFFE's wrapper Output*

ITCFFE then reduces the generated sets using OLSs, which are LRs (Linear Regressions) that create predictions or model dependent variables based on their correlations to sets of explanatory variables. They are commonly used strategies for estimating the coefficients of equations of LRs that explain the connection between one or more independent quantitative variables and a dependent variable (simple or multiple LRs). The term "least squares" refers to the smallest squares mistakes. Figure 10 displays ITCFFE’s output from application of OLSs.



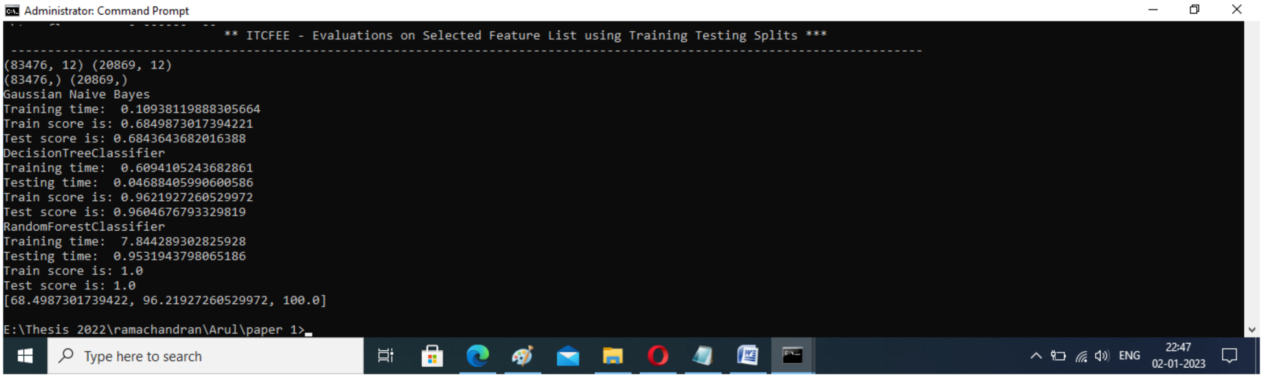
```
                 *** Fearure Mapping OLS model   ***
----------------------------------------------------------------

const            0.000000e+00
switch           0.000000e+00
pktcount         0.000000e+00
dur              0.000000e+00
dur_nsec        1.059532e-168
flows            0.000000e+00
packetins       6.930422e-232
pktperflow       0.000000e+00
Pairflow         0.000000e+00
port_no         1.243541e-127
tx_bytes         2.876786e-01
rx_bytes        9.126287e-113
tx_kbps         2.017260e-124
label            0.000000e+00
Protocol_TCP     0.000000e+00
dtype: float64
                      *** Backward Elimination   ***
----------------------------------------------------------------

                 Selected Columns Count     13

                 Selected Columns           13
['switch', 'pktcount', 'dur', 'dur_nsec', 'flows', 'packetins', 'pktperflow', 'Pairflow', 'port_no', 'rx_bytes', 'tx_kbps', 'label', 'Protocol_TCP']

   switch  pktcount  dur   dur_nsec  flows  packetins  pktperflow  Pairflow  port_no  rx_bytes  tx_kbps  label  Protocol_TCP
0       1     45304  100  716000000      3       1943       13535         0        3      3917        0      0             0
1       1    126395  280  734000000      2       1943       13531         0        4      3520        0      0             0
2       1     90333  200  744000000      3       1943       13534         0        1      1242        0      0             0
3       1     90333  200  744000000      3       1943       13534         0        2      1492        0      0             0
4       1     90333  200  744000000      3       1943       13534         0        3      3665        0      0             0
```

Fig. 10 - *ITCFFE’s output from application of OLSs (Final Output)*

ITCFFE – Feature Bifurcations

Splitting the dataset into train and test sets is a key element of data pre-processing since it allows us to increase the performance of our model and hence provide greater prediction. We may think of it this way: if we train our model with one dataset and then test it with a completely other dataset, our model will be unable to recognise the correlations between the features. As a result, training and testing the model with two separate datasets will reduce the model's performance. As a result, it is critical to divide a dataset into two portions, namely the train and test sets. The dataset with selected features were evaluated with classifiers for comparative performances.

Performance Evaluations: The selected features sets are classified for evaluations using RFs (random Forests), GNBs (Gaussian Naïve Bayes) and DTs (decision Trees). The proposed ITCFFE’s selected feature sets were used by the aforesaid classifiers and evaluated in terms of training and testing accuracies. Figure 11 depicts the comparative outputs of evaluations of classifier performances on ITCFFE selected feature sets in Training. Figure 12 shows comparative performances of the same classifiers in test dataset.



```
                             ** ITCFEE - Evaluations on Selected Feature List using Training Testing Splits ***
 -------------------------------------------------------------------------------------------------------------
(83476, 12) (20869, 12)
(83476,) (20869,)
Gaussian Naive Bayes
Training time:  0.10938119888305664
Train score is: 0.6849873017394221
Test score is: 0.6843643682016388
DecisionTreeClassifier
Training time:  0.6094105243682861
Testing time:  0.04688405990600586
Train score is: 0.9621927260529972
Test score is: 0.9604676793329819
RandomForestClassifier
Training time:  7.844289302825928
Testing time:  0.9531943798065186
Train score is: 1.0
Test score is: 1.0
[68.4987301739422, 96.21927260529972, 100.0]

E:\Thesis 2022\ramachandran\Arul\paper 1>_
```

Fig. 11 – *Comparative performances of classifiers on ITCFFE selected feature sets in Training*

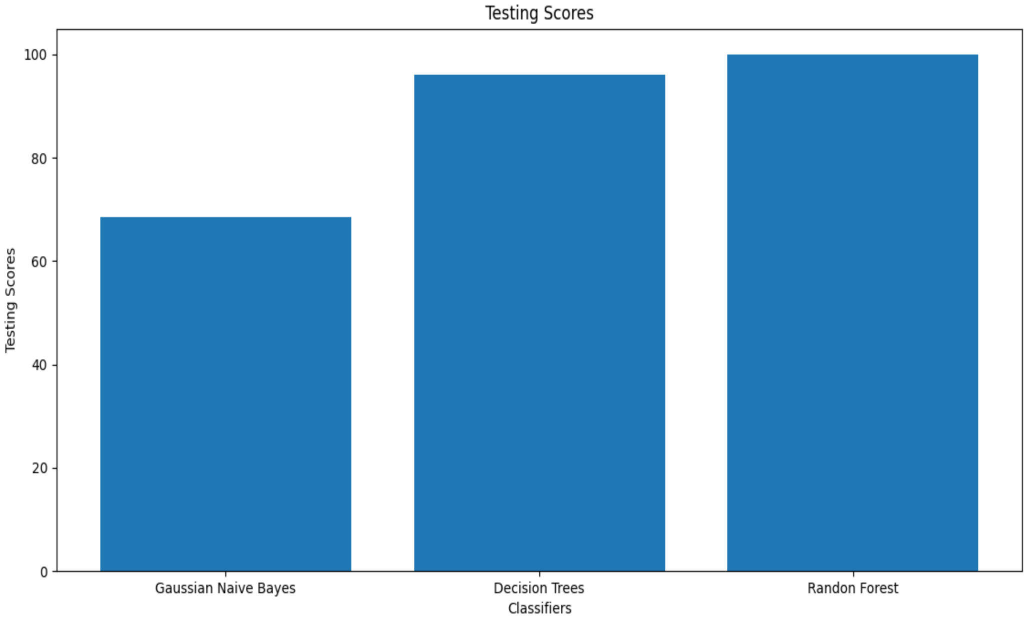


Fig. 12 – *Comparative performances of classifiers on ITCFFE selected feature sets in Testing*

ITCFFE – Classifications:

Conclusion

Conventional networks were formerly used to transfer data between nodes. The main issue with these networks was that they weren't particularly dependable and couldn't accommodate freshly added devices. As a result, conventional networks are being replaced with SDNs which carry data of multiple networking applications. In essence, SDNs are dynamic and can be used as foundations for applications that need a lot of data like big data. Centralizations of SDNs are their major advantages. Network evolutions are responsible for new kinds of assaults which stem as known and unknown hazards, and zero-day exploits. Since there are currently no histories of prior real-case attacks on SDNs, it is difficult to identify current weaknesses and create protections around these network controllers. A taxonomy of possible attacks can assist in establishing foundations of security. The centralized controllers create issues. New network technologies might pose previously unknown hazards or perhaps make matters worse, since controllers and links to control planes present novel security issues that are specific to SDNs. Using a dataset, this paper has explored a difficult area linked to malicious packets in traffics of SDNs. Popular network assaults may also affect SDNs which are more vulnerable to malicious traffics than traditional networks. In traditional networks, assaults may only damage subsets of networks from the same vendor without bringing the entire network down. However, in SDNs, hacked switches or end-users might overwhelm controllers, causing widespread network disruptions. Hence, this study proposed ITCFFE assesses this intensity, has focused on detection systems for identifying malicious packets in SDNs. created for selecting features required to classify network parameters for implementing HDAs has been proposed in this work. Identifying features that are relevant, minimal and apt have to be chosen either manually or automatically. Moreover, recurrences or duplication of fields while analyzing them increase the dataset size and consume costly processing time in computers. Hence dimensionality reduction techniques are applied for better results. This proposed work contributes to researches on SDNs by proposing a model based on DLTs for improved classifications of malicious packets. Future work would be using DLTs for classifications of SDN's traffic data.

References

H. Cui, Y. Zhu, Y. Yao, L. Yufeng, Y. Liu, "Design of intelligent capabilities in SDN," IEEE International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE), pp. 1-5, 2014.

Z. Arslan, A. Alemdaroglu, B. Canberk, "A traffic-aware controller design for next generation software defined networks," IEEE International Conference on Communications and Networking (BlackSeaCom), pp. 167-171, 2013.

J. Zhang, Y. Xiang, W. Zhou, Y. Wang, "Unsupervised traffic classification using flow statistical properties and IP packet payload," Journal of Computer and System Sciences, vol. 79, no. 5, pp. 573-585, 2013

Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, G. Noubir, "Application-awareness in SDN," ACM SIGCOMM computer communication review, vol. 43, no. 4, pp. 487-488, 2013.

Jadidi Z, Muthukkumarasamy V, Sithirasenan E, Sheikhan M (2013). 'Flow-based anomaly detection using neural network optimized with gsa algorithm'. In: 2013 IEEE 33rd international conference on distributed computing systems workshops, pp. 76–81.

Kokila R, Selvi ST, Govindarajan K (2014). 'Ddos detection and analysis in sdn-based environment using support vector machine classifier'. In: 2014 sixth international conference on advanced computing (ICoAC). IEEE, pp. 205–210.

Ashraf J, Latif S (2014). 'Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques'. In: 2014 national software engineering conference, Rawalpindi, pp. 55–60.

Dhanabal L, Shantharajah P (2015). 'A study on NSL-KDD dataset for intrusion detection system based on classification algorithms'. Int J Adv Res Comput Commun Eng 446–452.

Ingre B, Yadav A (2015). 'Performance analysis of NSL-KDD dataset using ANN'. In: 2015 International conference on signal processing and communication engineering systems, Guntur,pp. 92–96.

Phan TV, Van Toan T, Van Tuyen D, Huong TT, Thanh NH (2016). 'Openflowsia: an optimized protection scheme for software-defined networks from flooding attacks'. In: 2016 IEEE sixth international conference on communications and electronics (ICCE). IEEE, pp. 13–18.

Tang T, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M (2016). 'Deep learning approach for network intrusion detection in software defined networking'. In: 2016 International conference on wireless networks and mobile communications (WINCOM) (WINCOM16), Fez, Morocco, Oct 2016

Louridas P, Ebert C (2016). 'Machine learning'. IEEE Softw 33(5):110–115.

Abubakar A, Pranggono B (2017). 'Machine learning based intrusion detection system for software defined networks'. In: Proceedings of the 2017 eighth international conference on emerging security technologies (EST). IEEE.

C. T. Huawei Press Centre and H. unveil world's first commercial deployment of SDN in carrier networks (28 Feb 2018). Retrieved from http://pr.huawei.com/en/news/hw-332209-sdn.htm.

['Open Networking Foundation, ONF SDN Evolution' (25 Feb 2018). Retrieved from http://3vf60mmveq1g8vzn48q2o71awpengine.netdnassl.com/wpcontent/uploads/2013/05/TR-535-ONF-SDN-Evolution.pdf

OpenDaylight: a Linux foundation collaborative project' (11 March 2018). Retrieved from http://www.opendaylight.org