Network Transmission Flags Data Affinity-based Classification by K-Nearest Neighbor

Nahla Aljojo

Department of Information System and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia

***Abstract*—This research is concerned with the data generated during a network transmission session to understand how to extract value from the data generated and be able to conduct tasks. Instead of comparing all of the transmission flags for a transmission session at the same time to conduct any analysis, this paper conceptualized the influence of each transmission flag on network-aware applications by comparing the flags one by one on their impact to the application during the transmission session, rather than comparing all of the transmission flags at the same time. The K-nearest neighbor (KNN) type classification was used because it is a simple distance-based learning algorithm that remembers earlier training samples and is suitable for taking various flags with their effect on application protocols by comparing each new sample with the K-nearest points to make a decision. We used transmission session datasets received from Kaggle for IP flow with 87 features and 3.577.296 instances. We picked 13 features from the datasets and ran them through KNN. RapidMiner was used for the study, and the results of the experiments revealed that the KNN-based model was not only significantly more accurate in categorizing data, but it was also significantly more efficient due to the decreased processing costs.**

***Index Terms*—Transmission control protocol flags, K-nearest neighbors, Investment, Financial risk, Deep learning.**

I. Introduction

The transmission control protocol (TCP) operations is conducted with the use of flags which are very important component of the TCP protocol that must be understood to perform transmission of data over a network and for the network to function properly (Hartpence and Kwasinski, 2020). Those transmission flags are contained within the seventh field of the TCP header and can be set to either 0 or 1 depending on the situation, which they regulate and determine how connection states are managed as well as the manner in which packet transfers are carried out (Kadhim and Abed, 2017). It is also possible to use flags to control the establishment of connections as well as the closure and termination of connections; in other words, when a flag is turned on, it is referred to as being set; conversely, when a flag is turned off, it is referred to as being unset. A total of nine TCP flags can be set, six of which are commonly used in network communications and the other three are not. When a flag is set to 1, it indicates that a control was set for the function of that flag in the and when it is set to 0, it means it's off. One of the major functions of TCP in transmission is to "control" the transmission process in general. A TCP segment should be processed first, for example, if there is a problem with priority, TCP would be able to give priority of one segment over other segments with the use of flag. Similarly, TCP would be able to provide control over transmission and retransmission and many other transmission tasks.

The main research problems that this present study highlighted lie with the use of data. That is in this study, the usage of data created during the transmission session linked with the activities of flags was identified as one of the most significant research problems that needed to be addressed. When the sending computer sends, for example, a "push flag," it is usually to inform the receiving computer that the sending computer should flush the TCP buffers and send whatever data are still present in them at the time the push flag is sent. The push flag can be used to indicate a variety of different things associated with the payload in different circumstances. It means gathering such scenarios, there will be some insight that will be gain in order to properly understand the transmission operations fully. In addition, it is usual for the transmitting computer to send a "push flag" to inform the receiving computer that it should flush the TCP buffers and deliver any data that are still present in them at the time the push flag is sent. When used in a variety of various contexts, the push flag can be used to convey a variety of different things that are linked to the payload. By analyzing such circumstances, some information can be gathered that can then be used to correctly identify the transmission flaws as well. Moreover, given that these flags must be toggled on or off, their impact on various application layer protocols on different transmission sessions will be extremely significant for understanding network communication in general, something that has hitherto been overlooked by the academic community.

ARO-The Scientific Journal of Koya University
Vol. X, No. 1 (2022), Article ID: ARO.10880. 43 pages
DOI: 10.14500/aro.10880
Received: 17 September 2021; Accepted: 21 February 2022
Regular research paper: Published: 25 April 2022
Corresponding author's email: nmaljojo@uj.edu.sa

 http://dx.doi.org/10.14500/aro.10880

Considering the research problems highlighted above, the objective of the present study is to examine the use of data generated during the transmission session in conjunction with the activities of transmission flags, based on the fact that this has been outline as the current research gaps and as one of the most critical research concerns that required more investigation. There are many motivations for achieving this objective. The crucial one lies with the use of reset flag. Consider a transmission where an attempt to establish a connection result in the return of a reset flag; however, it is possible that an attempt will be made and a reset flag will be returned because a port may not be open at the time of the attempt. It means that information will be generated in various aspects that would require further analysis. Furthermore, the previous research studies have identified that TCP flags can be used in packet analysis to determine the state of the communications process at any given point in the TCP conversation or to trace a session from its inception to its conclusion, depending on the protocol (Kumar, et al., 2018; Chow, Li, Mountrouidou, 2017; Muelas, et al., 2017; Kushwah, et al., 2019; Hartpence and Kwasinski, 2020; Tomar, 2019; Sahi, et al., 2017; D'souza, et al., 2020).

The present study makes it obvious that the unit of analysis is "data," and as a result, the problem of connection construction between two TCP segments is handled as the core major problem. TCP flags can be used to determine the current state of the communication process at any given point in the conversation by studying the contents of the flags (Gital, et al., 2016). In reality, malicious users can take advantage of TCP flags to their own advantage; they can be configured in such a way that they can be used to launch denial of service attacks and other malicious actions on the network (Amanowicz and Jankowski, 2021). Considering a network scenario, where a central source and four destinations are in transmission (Fig. 1). In all the four cases, the various TCP flags can be found in the TCP header, despite coming from a single source and they are responsible for the transmission and flow of packets across the network connection. As a result, they are approximately in control of how data are transmitted, and how data are processed. Each of the TCP flags is on its way out to carry out its responsibilities on the target in all the four connections. The urgent flag is used to specify that a packet must be processed immediately, and you're attempting to pass that information along to the target or to any other device that will be processing that packet. The push flag is used to transmit data as soon as it is received. The Fin brings the transmission to a close. The ACK flag indicates that a packet has been successfully received. To initialize the connection between two devices, the SYN or synchronization flag is used.

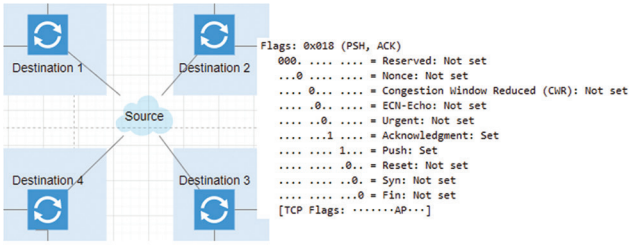

Fig. 1. The types of TCP flags in a transmission session.

II. Related Work

Analyzing transmission session data are mostly carried out in response to network monitoring (Demertzis, et al., 2021), crucial to that is employing an architecture that monitors the entire traffic and analyzes them for identification of either attacks or real-time problems. Data are generated within a network operation for various reason, an organization's network activity generates data, such as new benchmark datasets for evaluating data-driven intrusion detection systems (Abubakar, et al., 2015). The generation of data from any network process or operation is necessary, but the analysis of that data is much more crucial. Poorzare and Calveras (2021) generate network transmission data to reveals an understanding of why TCP cannot differentiate between congestion and other network flaws that can cause packet drops. Atan, et al. (2021) utilized some traffic data to gain an understanding of degradation attacks and TCP performance. Although data can be collected from various area of network, the analysis of such kind of data is the key problems. That is why there are many machine learning approaches to analysis data to draw out some values from such data.

The K-nearest neighbors (KNN) classifier has been used in a number of the previous studies to better understand the functioning of networks. Gordon et al. (2021) revealed that to identify and classify Internet of Things devices, as well as to detect several types of DDoS attacks, including TCP-SYN, UDP, and ICMP, KNN has performed. Dini and Saponara (2021) utilized KNN for instruction detection in a network transmission session. In analysis with data, the amount of recorded patterns makes the approach more efficient. KNN provides a set of patterns in the training set. Given KNN and other machine learning algorithms operations, it is easy to see how the KNN method of classification is a simple but extremely effective method of categorizing network data associated with network transmission (Nikam, 2015). In a similar vein, it has been demonstrated that a classification algorithm known as the KNN is critical in the solution of fundamental classification problems for network transmission operations (Alweshah, et al., 2020). KNN has also been recognized as the most appropriate option when dealing with long-term network operations datasets (Jannach and Ludewig, 2017). Despite having a wide acceptance, KNN also faces some drawbacks.

It has been recognized that analysis associated with KNN is mostly affected by the preparation of the "good value" for the parameter k, which is also required before constructing a network of nodes. It's a significant problem in the KNN algorithm because it makes selecting a "good value" for k more difficult than it should be, which makes it less efficient (Zhang, et al., 2017). Furthermore, another significant shortcoming of the algorithm is that it does not take into consideration any of the input data, which is another serious

flaw in the algorithm (Liao, et al., 2021). In a survey of a techniques, datasets, and challenges in intrusion detection systems, KNN has been identified to be among the crucial intrusion detection technique (Khraisat, et al., 2019), several points were highlighted on the reason why KNN retains all of the training data for classification, which many other algorithms discard those portions of it to improving their performance.

Using KNN for anomaly detection in TCP/IP networks, Zanero and Savaresi (2004) did a critical classification analysis for the aim of anomaly detection in TCP/IP networks, and the findings were published. To perform their analysis, the research team used a clustering approach followed by normal anomaly detection techniques. In a network transmission session, Ponmaniraj and Anand (2018) were able to analyses both the usual traffic pattern and the anonymous traffic pattern using KNN. Wenke and Stolfo (1998) developed an established framework capable of taking a classification and clustering techniques for detection intrusion detection on specific network scenarios with the purpose of detecting hostile activity and applying them to specific network situations. A better grasp of how many potentially harmful patterns can be discovered by an intrusion detection system was demonstrated to gain a better comprehension of the concept. A majority distance-based weighting (Zhang, 2020) has been demonstrated to extend the application of KNN to the setting of classification in a variety of situations, as demonstrated by Zhang, et al. KNNs in network management and analysis, on the other hand, have gained widespread acceptance as a result of the demand for the selection of a suitable value for k, which has a substantial impact on the performance of the classification method when it is combined with other algorithms. A recent study on KNN suggests that an alternative KNN technique should be used. To summarize, the KNN model approximately recalls all of the training samples and compares the current sample with the k nearest points to draw conclusions from the data. Whereas there are a variety of approaches for determining the k value, the easiest is to run the algorithm several times with different k values and then choose the one that performs the best on average.

III. The KNN Model

It was discovered over the course of adoption of KNN, there are many ways to implement it. This study utilized two KNN-related algorithms (Algorithm 1 and Algorithm 2). The algorithms are concerned with the cost of categorizing the datasets with all of the information associated with transmission flags and network application layer protocols, and the techniques are designed to minimize this cost to the greatest extent possible, according to the requirements of the specifications. As a result, the conceptualization process follows a similar pattern to the classification process. It has been previously stated that the reason for this is due to the simple fact that practically all the computation that occurs during the classification process depend on the computational resources and value of k, and also the size of the training samples. Algorithm 1 entails running KNN for the first round and then following the procedure where the k in KNN was picked at random with no consideration for its impact on the outcome.

Algorithm 1: The first round KNN Algorithm
Input: *s, x, y*;
Output: *Class of I*
Initialize the distance *d* (*x*′λ′) between the points (x, xi)" in the dataset
Set s{*I*} where 1, 2, 3., *n*
d within points (1+*n*, ∞,).
set *n*⟶ ∞ and find *k* distances
$k \geq 1$
end.
match k-points & *d*
if ($k_i > k_j$) and $I >$ j,
set $x \in 1$
end
end

On completion of the successful implementation of Algorithm 1, it was discovered that the KNN had been implemented, but not in the most efficient manner. Because it cannot be used to areas where dynamic categorization is required for a where the value of k is required, the first model can be seen as pre-modeling to maximize its efficiency. As a result, in Algorithm 2, validation was accomplished by maximizing the value of k. Attempts are made to tackle these concerns, and an optimization parameter is presented to obtain the desired value of k.

Algorithm 2: The second round KNN Algorithm
Input: *s, x, y*;
Output: *Class of s*
Initialization: *f*or (*x*′λ′) ∈ *y* set task and do
Normalize *x* and *y*;
match flags labels;
end
for each set of s;
set the distance(*d*) within (*x*, *y*) ∈ R
sort (d)
Obtaining class labels k-nearest point to (d)
end
end

IV. Experimental Analysis and Evaluation Technique

This section of the study describes in detail the experimental analysis processes that were carried out on the basis of the conceptualization of transmission flags that had an impact on the network-aware application (application layer protocols). Preparation of data, pre-processing of data, and final analysis are all necessary steps in the experimental analysis and evaluation of outcomes. The KNN was used to forecast the classification of the model before it was implemented.

A. Dataset

Kaggle provided the raw data for this study, which contained some transmission sessions through IP flow with 87 features and 3.577.296 occurrences, which were employed in this investigation. Imagine that we are looking at a transmission pool, which is primarily concerned with the transmission of data via specific ways; once the data have been processed, it will be transferred to one or more networks, for example (processes). Because of the unpredictability of the data gathering technique, the amount of data collected will vary from session to session. Furthermore, it is critical to emphasize the amount of information that can be obtained through network transmission, as this allows any filtering and aggregation capabilities to be performed to any of the packets that were delivered as a result of the transmission operation to be highlighted. Remember that the information gathered is primarily intended for the development of models that classify the interplay between network transmission flags associated with network-aware applications on transmission sessions, which will then be used to construct models based on the classification models. The dataset indicates that only 13 of the 87 features associated with flags were utilized, out of a total number of 87 features in the dataset overall (Table I). A number of variables must be considered, including: The number of times the push flag was set in packets transmitting in the forward direction (Fwd.PSH. Flags), the number of times the push flag was set in packets transmitting in the backward direction (Bwd.PSH.Flags), the number of times the urgent flag was set in packets transmitting in the forward direction (Fwd.URG.Flags), and the number of times the push flag was set in packet (Bwd. URG.Flags). Following that, is the count of the finish flag (FIN.Flag.Count), the count of the starting communication flag (SYN.Flag.Count), the count of the reset flag (RST.Flag. Count), the count of the push count flag (PSH.Flag.Count), the count of the acknowledgement flag (ACK.Flag.Count), the count of the urgent flag (URG.Flag.Count), and the count of the common weakness enumeration flag, which follows by the (ECN-Echo ECE.Flag. Count). The goal of all of them is to have some sort of impact on the network-aware application in some form, which means that they are conceptualized to have some sort of influence on the network-aware application (application layer protocols).

The network-aware apps are the most significant aspects of a network because they are at the heart of the underlying applications over IP flow, and they are responsible for either monitoring the status of the underlying network or receiving information about the status of the underlying network from network monitors. Although less important, the ability of the network to change its behavior in response to the information it receives is equally important, and it is associated with transmission flags, which are used to identify which protocols are being used at the application layer and are associated with transmission flags. A network transmission session is defined as a period of time during which an application delivers acceptable and predictable performance. A total of 60 of these applications were acquired during a transmission session, and they are included in the current dataset (Table II). Google was determined to have the greatest number of transmission sessions, whilst NFS count only had one, making it the least amount of transmission sessions among the dataset's participants. This study hypothesized that transmission flags had an impact on these protocols.

B. Data Pre-processing

KNN method uses a distance measure, which is determined by the scale of the variables being compared, to get classification results. When the unit of measurement

TABLE I
FEATURES ASSOCIATED WITH FLAGS IN THE DATASET

Fwd. PSH. Flags	Bwd. PSH. Flags	Fwd. URG. Flags	Bwd. URG. Flags	FIN. Flag. Count	SYN. Flag. Count	RST. Flag. Count	PSH. Flag. Count	ACK. Flag. Count	URG. Flag. Count	CWE. Flag. Count	ECE. Flag. Count	ProtocolName
0	0	0	0	0	0	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	0	1	1	0	0	HTTP_PROXY
1	0	0	0	0	1	0	0	1	0	0	0	HTTP
0	0	0	0	0	0	0	0	1	1	0	0	HTTP
1	0	0	0	0	1	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	0	1	0	0	0	HTTP_PROXY
1	0	0	0	0	1	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	1	0	0	0	0	HTTP_CONNECT
1	0	0	0	0	1	0	0	1	0	0	0	SSL
0	0	0	0	0	0	0	1	0	0	0	0	GOOGLE
1	0	0	0	0	1	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	0	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	1	1	0	0	0	HTTP_PROXY
0	0	0	0	0	0	0	1	1	0	0	0	SSL
1	0	0	0	0	1	0	0	1	0	0	0	HTTP
1	0	0	0	0	1	0	0	1	0	0	0	HTTP
1	0	0	0	0	1	0	0	1	0	0	0	HTTP

Table II
The underlying applications over IP flow in the dataset

#	Protocol Name	Transmission Sessions	#	Protocol Name	Transmission Sessions	#	Protocol Name	Transmission session
1	Google	256726	21	Instagram	1159	41	WAZE	52
2	HTTP	254525	22	WhatsApp	829	42	NTP	40
3	HTTP_Proxy	154026	23	Wikipedia	741	43	Easytaxi	34
4	SSL	131461	24	Netflix	699	44	Twitch	24
5	HTTP_Connect	94362	25	MS_One_Drive	654	45	Unencryped_Jabber	19
6	Youtube	46236	26	DNS	516	46	Deezer	16
7	Microsoft	19389	27	IP_ICMP	503	47	Citrix	11
8	Amazon	15495	28	Apple_Itunes	376	48	Whois_Das	10
9	Windows_Update	11996	29	Ebay	345	49	Opensignal	9
10	Gmail	9565	30	Apple_Icloud	322	50	Skinny	8
11	Skype	7497	31	SSL_NO_CERT	300	51	Oracle	7
12	Yahoo	7450	32	HTTP_Download	157	52	Edonkey	6
13	Facebook	7020	33	Spotify	136	53	MSSQL	4
14	Dropbox	6780	34	Teamviewer	130	54	UPNP	4
15	Twitter	5315	35	TOR	110	55	Mail_Imaps	3
16	Cloudflare	4228	36	Google_Maps	102	56	Openvpn	2
17	MSN	3791	37	Ubuntuone	93	57	Oscar	2
18	Apple	2103	38	SSH	74	58	Simet	2
19	Content_Flash	1610	39	MQTT	72	59	Starcraft	2
20	Office_365	1373	40	FTP_Data	53	60	NFS	1

is changed, the distance between two objects with the same length and mass will change dramatically. Because of this, all variables should be brought into the same range to be able to compare values that have been measured with more consistency. Finding any anomalies in the data, such as null values and outliers, was the first step in the process, which took several hours. A number of columns were omitted from the dataset because they were judged unimportant for analysis and prediction, such as those requiring natural language processing. Additional columns from the dataset were excluded from the dataset since they were deemed to be unrelated to the research on the basis of the problem domain.

C. Performance Evaluation

The performance of the classification algorithms is presented in Table III, it relies on the evaluation metrics such as accuracy, recall, precision.

The evaluation is critical when estimating the performance of a machine learning algorithm. Typically, performance is measured using indicators such as precision and recall. Precision and recall are two different metrics that describe how well a prediction algorithm performs when rejecting a non-relevant class, and precision and recall are two different metrics that describe how well the algorithm finds all relevant classes. A binary label is used to differentiate between what happened in real life and what happened in the prediction when evaluating precision and recall.

Finally, the evaluation of the model will follow using the sensitivity and specificity measures. Considering that substantial research studies on prediction utilized rules based scores, sensitivity and specificity in identifying and predicting problems, the sensitivity-based approach reveals the efforts of each flags contribution and the least effect on the protocol. Hence, the positive and negative class of performance measure present true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). TP: Precisely predict, FP: Erroneously predict, FN: Erroneously rejected, TN: Precisely rejected. This is used for measuring the "Sensitivity (S_e)," and "Specificity (S_p)," based on the following evaluates the performance measure of the models used:

Table III
Performance evaluation metrics

	True+ve	True –ve	Precision
Pred+ve	Count of TP	Count of FP	PPV
Pred–ve	Count of FN	Count of TN	NPV
Recall	Sensitivity	Specificity	Accuracy

- Accuracy $=TP+TN/TP+FP+FN+TN$
- Classification error =1−Accuracy
- Positive precision value (PPV) $=TP/TP+FP$
- Negative precision value (NPV) $=TN/FN+TN$
- Sensitivity/true positive rate (TPR) $=TP/TP+FN$
- Specificity/true negative rate (TNR) $=TN/FP+TN$.

D. Experimental Simulations

RapidMiner Studio 9.9 was used for the processing component of the analysis, which allowed for a wide range of options to be used in the data preparation and analysis. Under all conditions in this experiment, the ideal numbers split (0.7–0.3) was employed for both training and testing in all situations, respectively, irrespective of the context. All of the model's attributes were implemented as a result of the model's correctness being determined. To conduct this experiment, an Intel® CoreTM i7-10750H CPU running at 5.0 GHz and 16 GB of total RAM were employed in a computer system powered by an Intel® CoreTM i7-10750H processor. When it comes to network-aware application variables, the envisioned influencing of transmission flags

http://dx.doi.org/10.14500/aro.10880

is based on comparing multiple flags one after another in terms of their impact on the protocol for throughout the transmission session, which is why KNN was utilized. The simplicity of the KNN algorithm's design makes it an easy distance-based supervised learning algorithm that merely remembers earlier training samples and compares a new sample with the K-nearest points to make a decision.

V. Presentation of the Results and Discussion

There was a large amount of data entered into the training dataset, and there are no missing data records inside the record. This set of datasets was divided into training and testing datasets using a variety of percentage splits. A series of investigations were conducted; the model is presented in Fig. 2. The performance of the prediction model for each partition was recorded and analyzed in detail. Algorithm 1 was used to conduct the first round of analysis. The accuracy of the model's performance was greater than 80%, and the sensitivity and specificity of the model were all reported in Table IV to demonstrate their effectiveness.

As a result, the value of k was adjusted in the following round of analysis, but not in accordance with the optimization prediction; as a result, the analysis with the same large amount of data entered into the training dataset, where it was divided into training and testing datasets using a model presented in Fig. 3 shows that the performance of the prediction model for each partition was recorded and analyzed. The first phase of analysis was carried out with the help of Algorithm 1. Performance of the model was more than 80% accurate; the model's sensitivity and specificity were all presented in Table V to indicate its efficacy.

The successful implementation of Algorithm 1 was followed by the discovery that the KNN had been implemented, but that it had not been done in the most efficient manner.

Table V
The performance of implementation of the modified Algorithm 1

	True SF	True S0	True REJ	Class precision
Pred. SF	4244	269	306	88.07%
Pred. S0	246	1820	640	67.26%
Pred. REJ	2	14	17	51.52%
Class recall	94.48%	86.54%	1.77%	

Table IV
The performance of implementation of the Algorithm 1

	True SF	True S0	True REJ	Class precision
Pred. SF	4244	269	306	88.07%
Pred. S0	246	1820	640	67.26%
Pred. REJ	2	14	17	51.52%
Class recall	94.48%	86.54%	1.77%	

Table VI
The performance of the final implemented Algorithm 2

	True SF	True S0	True REJ	Class precision (%)
Pred. SF	2959	184	228	87.78
Pred. S0	185	1287	439	67.35
Pred. REJ	0	1	7	87.50
Class recall	94.12%	87.43%	1.04%	

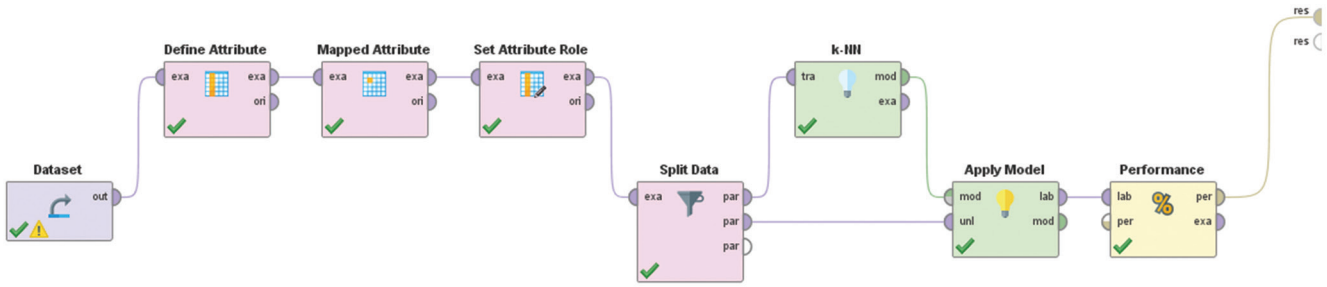

Fig. 2. The model for implementation of the Algorithm 1.

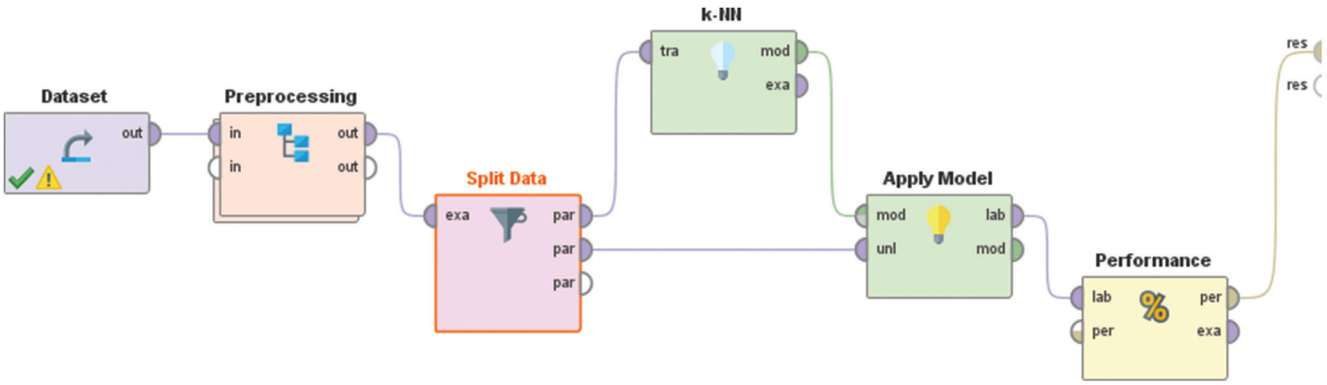

Fig. 3. The model for implementation of the modified Algorithm 1.

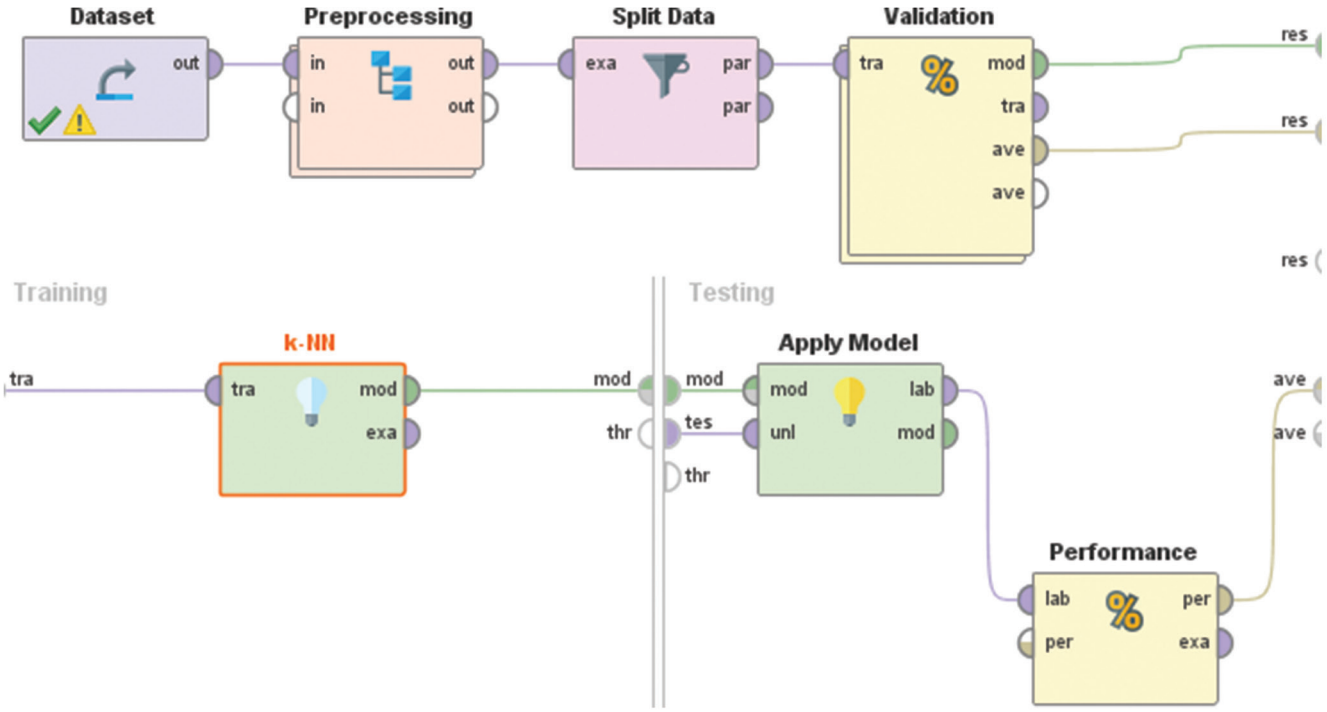

Fig. 4. The final model for implementation of the Algorithm 2.

It is possible to think of the first model as pre-modeling to maximize its efficiency because it cannot be applied to areas where dynamic categorization is necessary and where the value of k is required. As a result, Algorithm 2 was developed, in which the optimization was performed by increasing the value of k to its maximum. Following the completion of the optimization, the required value of k was determined, and this value was utilized to modify the model (Fig. 4).

The model's sensitivity and specificity were all provided in Table VI to demonstrate its effectiveness, and as a result, we obtain the greatest performance scores of the model, which was more than 80% accurate.

VI. Discussion

According to the findings of this study, there are six different most essential transmission flags that TCP uses (the push flag, the reset flag, the fin flag, the synchronization flag, the acknowledgement flag, and the urgent flag). These flats are critical for the transmission session to be successful. Typically, to terminate a connection, the fin flag must be used, whereas the syn flag must be used when sending a connection request, and the ACK flag must be used if we wish to send an acknowledgment of a request. An instance of the push flag is one in which the sender and receiver intend to engage in interactive conversation, which may begin with the transmission of two bytes. When a sender's payload is 2 bytes, a minimum of 20 bytes IPv4 header and a maximum of 60 bytes will be appended at the transport layer, resulting in a total of [2+20=22] bytes. A minimum of 20 bytes IPv4 header will be added at the network layer, and a maximum of 60 bytes will be added at the application layer, so let's assume the minimum, which will result in 22+20=42 bytes. It is expected that an extra minimum of 48 bytes will be added at the data connection layer, increasing the total amount of bytes received at the receiver to 90 bytes. To put it another way, to convey two bytes of information, a total of 90 bytes of information must be sent out. And what is the efficiency of sending 2 bytes over a 90 byte transmission, to put it another way. This is when the usage of a push flag proves to be quite advantageous. Rather of waiting until a particular amount of data has been compiled into a segment, it enables the transmission to push the two bytes immediately after they are received. Following the arrival of a pair of bytes in a given transmission session, they are sent out using the push flat method. When the push flag is set to the first position, this occurs. When the current connection fails, the reset flag is used to attempt to re-establish the connection with the server again. It is possible to priorities interconnected transmissions by referring to them as urgent and urgent pointer, respectively.

The classification of the interrelationships among the variables was accomplished through the application of KNN. There was a significant amount of data entered into the training dataset, and there are no missing data records contained inside the record itself. With the help of a number of percentage splits, this collection of datasets was divided into training and testing datasets. Investigations were carried out in a number of different ways. The outcome of the prediction model's performance for each partition was recorded and studied in great detail. Both algorithms outperformed their counterparts.

It's also worth noting that the network transmission session is considered a data generating tool, because it generates data and adds some value to the organization and management in the long run. Numerous solutions can be derived from this data. Some key qualities linked with them can be found

using those data in this essential area of data science. Since the data created during a network transmission session might be used to extract value, our research addressed the problem. So that any study could be conducted, this research conceptualized the influence of each transmission flag on network-aware applications by comparing the flags one by one on their impact to the application during the transmission session rather than comparing all of them at the same time.

VII. Conclusion

Data generated during a network transmission session is studied to discover the optimal method of extracting value from the data provided and being able to perform actions. However, instead of comparing them all simultaneously, this paper conceptualized each transmission flag's impact on network-aware apps by comparing each flag's effects one by one during the transmission session. By comparing each flag's impact on the application during the transmission session, this article conceptualized the impact of each transmission flag on network-aware apps. Because it is an easy-to-use distance-based learning algorithm that remembers prior training samples and can be applied to a variety of flags that have variable effects on application protocols, KNN type classification was chosen. Researchers found that the KNN machine learning algorithm was more accurate at categorizing data, but it was also more efficient due to the reduction in processing costs. Denoted as a data-generating instrument, the network transmission session generates valuable information for the company. These data can be used to find a variety of solutions. Those data can be used in this crucial area of data science to discover some of their most important characteristics. This was a concern of ours because the data generated during a network transmission session could be utilized to extract value. Therefore, to conduct a study on the impact of each transmission flag on network-aware apps, this research evaluated each flag individually, rather than comparing them all at once, to determine their impact on the programmer during the transmission session.

References

Abubakar, A.I., Chiroma, H., Muaz, S.A. and Ila, L.B., 2015. A review of the advances in cyber security benchmark datasets for evaluating data-driven based intrusion detection systems. *Procedia Computer Science*, 62, pp.221-227.

Alweshah, M., Al Khalaileh, S., Gupta, B.B., Almomani, A., Hammouri, A.I. and Al-Betar, M.A., 2020. The monarch butterfly optimization algorithm for solving feature selection problems. *Neural Computing and Applications*, 32(13), pp.1-15.

Amanowicz, M. and Jankowski, D., 2021. Detection and classification of malicious flows in software-defined networks using data mining techniques. *Sensors*, 21(9), pp.2972.

Atan, F.M., Zulkifl, N., Idrus, S.M., Ismail, N.A. and Zin, A.M., 2021. Understanding degradation attack and TCP performance in next generation passive optical network. *Journal of Physics: Conference Series*, 1933, p.012107.

Available from: https://www.kaggle.com/jsrojas/ip-network-traffic-flows-labeled-with-87-apps [Last accessed 2021 Jun 20].

Chow, J., Li, X. and Mountrouidou, X., 2017. Raising flags: Detecting covert storage channels using relative entropy. In: *2017 IEEE International Conference on Intelligence and Security Informatics* (*ISI*), pp.25-30.

D'souza, J., Kaur, M.J., Mohamad, H.A. and Maheshwari, P., 2020. Transmission Control Protocol (TCP) Delay Analysis in Real Time Network. In: *2020 Advances in Science and Engineering Technology International Conferences* (*ASET*), pp.1-6.

Demertzis, K., Tsiknas, K., Takezis, D., Skianis, C. and Iliadis, L., 2021. Darknet traffic big-data analysis and network management for real-time automating of the malicious intent detection process by a weight agnostic neural networks framework. *Electronics*, 10(7), p.781.

Dini, P. and Saponara, S., 2021. Analysis, design, and comparison of machine-learning techniques for networking intrusion detection. *Designs*, 5(1), p.9.

Gital, A.Y.U., Ismail, A.S., Chiroma, H. and Abubakar, A., 2016. TCP Skudai: A High Performance TCP Variant for Collaborative Virtual Environment Systems. In: *2016 6th International Conference on Information and Communication Technology for The Muslim World* (*ICT4M*), pp.118-121.

Gordon, H., Batula, C., Tushir, B., Dezfouli, B. and Liu, Y., 2021. Securing smart homes via software-defined networking and low-cost traffic classification. *arXiv*, 2021, p.00296.

Hartpence, B. and Kwasinski, A., 2020. Combating TCP Port Scan Attacks Using Sequential Neural Networks. In: *2020 International Conference on Computing, Networking and Communications* (*ICNC*), pp.256-260.

Hartpence, B. and Kwasinski, A., 2020. Combating TCP Port Scan Attacks Using Sequential Neural Networks. In: *2020 International Conference on Computing, Networking and Communications* (*ICNC*), pp.256-260.

Jannach, D. and Ludewig, M., 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-based Recommendation. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp.306-310.

Kadhim, J.M. and Abed, A.E., 2017. Steganography Using TCP/IP's Sequence Number. *Al-Nahrain Journal of Science*, 20(4), pp.102-108.

Khraisat, A., Gondal, I., Vamplew, P. and Kamruzzaman, J., 2019, Survey of intrusion detection systems: Techniques, datasets, and challenges. *Cybersecurity*, 2, p.20.

Kumar, P., Tripathi, M., Nehra, A., Conti, M. and Lal, C., 2018. SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN. *IEEE Transactions on Network and Service Management*, 15(4), pp.1545-1559.

Kushwah, D., Singh, R.R. and Tomar, D.S., 2019. An Approach to Meta-Alert Generation for Anomalous TCP Traffic. In: *International Conference on Security and Privacy*. Springer, Singapore, pp.193-216.

Liao, T., Lei, Z., Zhu, T., Zeng, S., Li, Y. and Yuan, C., 2021. *Deep Metric Learning for K Nearest Neighbor Classication*. IEEE Transactions on Knowledge and Data Engineering.

Muelas, D., de Vergara, J.E.L., Ramos, J., García-Dorado, J.L. and Aracil, J., 2017, On the impact of TCP segmentation: Experience in VoIP monitoring. In: *2017 IFIP/IEEE Symposium on Integrated Network and Service Management* (*IM*), pp.708-713.

Nikam, S.S., 2015. A comparative study of classification techniques in data mining algorithms. *Oriental Journal of Computer Science and Technology*, 8(1), pp.13-19.

Ponmaniraj, S., Rashmi, R. and Anand, M.V. 2018, IDS Based Network Security Architecture with TCP/IP Parameters Using Machine Learning, *2018 International Conference on Computing, Power and Communication Technologies* (*GUCON*), 2018, pp.111-114.

Poorzare, R. and Calveras, A., 2021. FB-TCP: A 5G mm wave friendly TCP for urban deployments. *IEEE Access*, 9, pp.82812-82832.

Sahi, A., Lai, D., Li, Y. and Diykh, M., 2017. An efficient DDoS TCP flood attack detection and prevention system in a cloud environment. *IEEE Access*, 5, pp.6036-6048.

Tomar, D.S., 2019. An Approach to Meta-Alert Generation for Anomalous TCP Traffic. Vol. 939. In: *Security and Privacy: Second ISEA International Conference, ISEA-ISAP 2018*, Jaipur, India, January, 9-11, 2019. Springer, Berlin, p.193.

Wenke, L. and Stolfo, S.J., 1998. Data mining approaches for intrusion detection. In: *Proceedings of the 7th USENIX Security Symposium*, 7, pp.6-6.

Zanero S. and Savaresi, S.M., 2004. Unsupervised learning techniques for an intrusion detection system. In: *Proceedings of the 2004 ACM symposium on Applied computing SAC* 04, pp.412-419.

Zhang, S., 2020. Cost-sensitive KNN classification. *Neurocomputing*, 391, pp.234-242.

Zhang, S., Li, X., Zong, M., Zhu, X., and Cheng, D., 2017. Learning k for knn classification. *ACM Transactions on Intelligent Systems and Technology*, 8(3), pp.1-19.