Network traffic classification using Unsupervised Learning: a comparative analysis of clustering algorithms

Helena Canever, Xihui Wang

Network traffic classification using Unsupervised Learning: a comparative analysis of clustering algorithms

Helena Canever[1] and Xihui Wang[1]

[1]Talan Research and Innovation Center.

Contributing authors: helena.canever@talan.com; xihui.wang@talan.com;

Abstract

Network traffic classification plays a crucial role in various network management tasks, such as resource allocation, intrusion detection, and Quality Of Service (QOS) optimization. With the increasing complexity and volume of network traffic, the need for efficient and accurate classification algorithms has become paramount. Traffic classification plays a vital role in achieving QoS, but the task has become increasingly difficult due to evolving applications and growing network traffic. This article focuses on leveraging machine learning techniques, particularly unsupervised learning, for network traffic classification. The objective is to replicate the work of Aouedi et al. and compare the performance of various clustering algorithms: k-means, DBSCAN, bisecting k-means, and k-modes. A more refined experimental procedure is proposed, including data preparation, feature selection, hyperparameter analysis, and cluster analysis. This study contributes to the understanding of different clustering algorithms' performance for network traffic classification.

Keywords: Network traffic classification, Clustering, Unsupervised learning, Machine learning

1 Introduction

The digitalization of businesses, public administrations, and individuals has rapidly increased in recent years, fueled by growing trust in cloud infrastructures and 5G networks. This trend was further accelerated by the Covid-19 pandemic, which forced many employees to work remotely. However, this digital transformation has also introduced new challenges for network managers, particularly with regards to security and supervision. In this context, ensuring Quality of Service (QoS) in resource allocation has become a critical issue, especially given the increased heterogeneity of connections and software services.

Achieving QoS in a network necessitates the prioritization of certain traffic types over others. This objective can be accomplished by judicious allocation of bandwidth and other network resources, based on the distinct needs of various applications and services. For instance, real-time applications like video conferencing and online gaming demand low latency and high bandwidth, while file downloads or software updates can accommodate higher latency and may not necessitate as much bandwidth. Traffic classification plays a pivotal role in realizing QoS in a network. It involves the process of identifying the diverse traffic types coursing through the network, such as web traffic, email traffic, or video streaming traffic.

However, the task of manually identifying and classifying network traffic has become increasingly difficult. This is due to both the constantly evolving nature of applications and the continual increase in the volume of network traffic. In response to this challenge, there has been a growing interest in leveraging machine learning techniques to address the problem of network traffic classification. In particular, unsupervised learning methods have garnered attention considering their ability to perform statistical analysis on attributes without a priori labeling[1].

Unsupervised learning is a powerful technique that allows for the discovery of relationships between inputs without any prior knowledge of the outputs[2]. Its main objective is to process data for knowledge discovery, which can be used for reasoning and decision making. Clustering algorithms are one of the most commonly used methods of unsupervised learning, which group input data into distinct clusters based on similarities in the feature values. In particular, the work of Aouedi et al.[3] proposed the implementation of k-means for network traffic classification given its ease of implementation. To further test the efficacy of their method, the authors recommend comparing the performance of k-means with other clustering algorithms. The objective of this article is thus to replicate the work of Aouedi et al.[3] and compare the performance of various clustering algorithms for network traffic classification.

The rest of the paper is organized as follows: Section 2 focus on related work in network traffic classification; the Clustering algorithms selected and the dataset used for this study are described in Section 3 and 4; the experimental results and their analysis are provided in Section 5, and finally the paper concludes with Section 6.

2 Related Work

Network traffic, also referred to as data traffic or network data, represents the amount of data traversing a computer network at a given moment and is used for overall communication and information exchange between interconnected devices within the network. The transmission of network data heavily relies on the encapsulation process, where information is organized into network packets, serving as the fundamental units of network load. These packets consist of a payload, representing the raw data being transmitted, and a header that contains essential metadata such as the source and destination IP addresses.

Network traffic can be classified based on different criteria, including the source and destination of the traffic, the protocols used, the content or payload of the data, and the timing or behavior of the traffic. A well known classic approach is to use port numbers for traffic classification. However, to avoid detection by this method, P2P applications use dynamic port numbers and disguise themselves by using the port numbers for common protocols such as HTTP and FTP, thus rendering this approach obsolete[4]. An alternative approach is the payload-based technique[5], which inspects packet payloads for characteristic signatures of known applications, is more accurate than the port-based technique. Nevertheless, it has limitations including privacy and legal concerns with user data and the need to keep extensive knowledge of application protocol semantics up to date, which impose significant complexity and processing load on the traffic identification device.

Newer approaches to application identification rely on traffic's statistical characteristics, such as flow duration, packet inter-arrival time, and packet lengths as unique properties for certain classes of applications, which has led to the development of new classification techniques based on traffic flow statistical properties, and the introduction of machine learning techniques to deal with large datasets and multi-dimensional spaces of flow and packet attributes[6]. Most ML techniques used for traffic classification focus on the use of supervised and unsupervised learning. There are various supervised learning classification algorithms that have been applied to traffic classification, which differ in the construction of classification models and the optimization algorithm used to search for an effective model, such as Bayesian neural networks [7] and support vector machines[8]. However, the accuracy of supervised learning depends entirely on the dataset being labelled. The reality is that it is very difficult to manually label the data and there are always new arrivals of traffics that may not belong to any of the predefined labels. Since the unsupervised clustering approach does not have this limitation, it has attracted more attention in recent years. Although it is not as rich as supervised learning algorithms, it shows its applicability in constructing high-performance[9].

A series of works have exploited clustering algorithms on traffic data to improve the classification of network data. Among all the clustering algorithms, K-means[10], which aims to divide the data into K clusters, is the most popular thanks to its simplicity. Many works [11] [12] have verified the effectiveness of K-means for classifying network traffic. However, the number of clusters K

has been the focus of debate, which not only seriously affects the classification effectiveness performance of the algorithm, but also affects the algorithm' complexity. Other different clustering algorithms have been used to compare with K-means. For exemple, Erman et al.[13] compared the performances of the k-means to DBSCAN[14] algorithms and AutoClass algorithm[15]. In their experiments, for each different protocol (HTTP, P2P, etc.), the same number of samples are used to avoid clustering bias. This preprocessing method does not allow the identification of flows based on volume and necessary resources and is therefore less effective for practical applications. Singh[16] compared the accuracy of K-means and Expectation Maximization (EM)[17] clustering algorithms in identifying network traffic based on extracting relevant features. Unfortunately, their experimental processes such as feature processing, extraction, etc. were not described in detail, which makes it difficult to replicate their experiments for comparison.

Expanding on prior research, Aouedi et al.[3] proposed a more refined experimental procedure that includes the following two mains steps:

1. Data Preparation, involving data preprocessing and feature selection;
2. Experimental analysis, involving the analysis of hyperparameter values and elucidation of the behavior of distinct clusters.

Regrettably, only a detailed description of the selection of parameters of the k-means algorithm and the analysis of its clusters has been presented by Aouedi et al. Therefore, the thrust of this paper is to extend their work by comparing four different clustering algorithms(k-means, DBSCAN, bisecting k-means and k-modes) and analyzing in detail the different clusters. Over and above, we contend to provide a more rigorous approach to data preprocessing of the first step and performed an internal validation of the clusters on the reprocessed data to evaluate the optimal number of clusters.

3 Clustering algorithms

We have selected four clustering algorithms for comparative analysis, namely, DBSCAN, k-means, and two variations of k-means, namely bisecting k-means and k-modes.

3.1 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is a clustering algorithm that groups data points into density-based clusters. It does this by first identifying points in the dataset that are dense and form a cluster, then expanding the cluster to include nearby points that are also dense. Points that are not part of a dense cluster are considered noise and are not included in the clusters.

3.2 k-means

K-means is a clustering algorithm that divides a group of data points into a specified number of clusters (k) based on the means of the data in each cluster. To do this, it first initializes k centroids, which are points representing the center of each cluster. It then assigns each data point to the cluster whose centroid is closest, based on some similarity metric. Once all points have been assigned to a cluster, the centroids are updated to be the average of the points in each cluster, and the process is repeated until the centroids converge and the point assignments to clusters do not change.

3.3 Bisecting k-means

The k-means bisecting algorithm, as the name suggests, is based on the k-means algorithm. The difference is that the k-means bisecting algorithm first treats the entire data set as a single cluster. It then iteratively divides this cluster into two sub-clusters using the k-means algorithm. This process is repeated on each subcluster until a specified number of clusters (k) is reached. This results in a hierarchical cluster structure, with each cluster being divided into two sub-clusters at each iteration.

3.4 k-modes

k-modes is a clustering algorithm similar to k-means, but instead of using means to define clusters, it uses modes, which are the most frequent values in each cluster. This makes k-modes well suited to clustering categorical, rather than numerical, data. Like k-means, k-modes starts by initializing k centroids, then assigns each data point to the cluster whose centroid is closest. The centroids are then updated to be the mode of the points in each cluster, and the process is repeated until the centroids converge and the assignments of points to clusters do not change.

4 Data Preparation

In our research, we employed the identical dataset utilized in the analysis conducted by Aouedi et al. [3] to compare the four clustering algorithms selected. Specifically, the dataset was derived from the 'IP Network Traffic Flows, Labeled with 75 Apps' dataset, which is publicly available on Kaggle. This dataset was meticulously curated by collecting network data from the Universidad Del Cauca, Popayàn, Colombia, and consists of a substantial 3,577,296 instances.

4.1 Data preprocessing

We begin the data preprocessing by removing data duplicates in the dataset in order to avoid a bias towards said duplicates. Next, we review the dataset's features. The vast majority of the features are numerical and as such do

not require preprocessing other than normalization. However, we identify five features that are not numeric and require additional processing: Timestamp, Source.IP, Destination.IP, Source.Port, Destination.Port. Timestamp instances can be converted to integers. IP addresses and ports are features that are represented numerically but are not numeric. For example, port 80 is assigned to HTTP protocols, port 110 is assigned to POP3 protocols, and port 443 is assigned to HTTPS protocols, but ports 80 and 110 are no closer to each other than port 443. In other words, the numerical nomenclature does not correspond to a correlation between ports. We have therefore opted for an arbitrary grouping of ports into five categories: "3128", "443", "80", "0" and "other". In the case of IP addresses, the dot-decimal notation, for example 192.0.2.1, represents a 32-bit notation, so we adopt the following conversion formula $16777216 \times 4^{th}decimal + 65536 \times 3^{rd}decimal + 256 \times 2^{nd}decimal + 1 \times 1^{st}decimal$.

4.2 Features Selection

The dataset consists of 87 features. Such a number of features imposes the issue of high dimensionality. The issue relates to the fact that with a high number of features comes the risk of encountering redundancy between features or features that do not provide information useful to the clustering, leading to an overall reduction in the performance of the algorithm. In the work of Aoeudi et al. they solve this issue by performing feature selection using Recursive Feature Elimination (RFE).The result is a reduction of the dataset to 15 features. At this phase we select features based on the work of Aouedi et al. reducing the dataset from 87 features to 15. This choice was guided by the desire to maintain a certain level of comparability between our work and that of Aouedi et al. and to reduce computations. Finally, we normalize the remaining 15 features.

5 Experimentation Analysis

5.1 Hyperparameter Analysis

The number of clusters is a hyperparameter of the k-means, bisecting k-means, and k-modes clustering algorithms. It must thus be chosen a priori. Different metrics allow to evaluate the ideal number of clusters, whereby ideal we refer to the best separation and definition of clusters with little overlap between clusters and low dispersion within each cluster.

In this work, we decided to adopt the following methods:

- Elbow method: this method evaluates the intra-cluster variance. In other words, the distance between data points within a cluster. The name of the method refers to the search for the inflection point in the decrease of the variance with the number of clusters. This finds the number of clusters at which the rate of decrease in variance decreases.

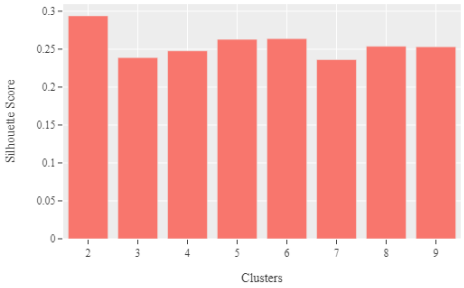

Fig. 1 Silhouette scores for clusters by number of clusters generated by the k-means algorithm.

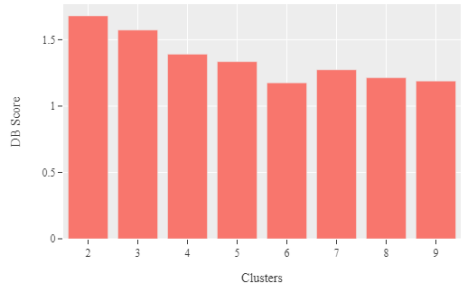

Fig. 2 Davies-Bouldin scores for clusters by number of clusters generated by the k-means algorithm.

- Silhouette score: measures the distance between clusters (and thus their significance). The higher the silhouette score, the more distinct the clusters are.
- Davies-Bouldin score: the score is defined as the average similarity measure of each cluster with its most similar cluster, where similarity is the ratio of intra-cluster distances to inter-cluster distances. Thus, more distant and less dispersed clusters will get a better score. The lower the score, the better.

We have applied these methods to clusters generated by the k-means and bisecting k-means algorithms. In executing the k-means algorithm the best Silhouette and Davies-Bouldin scores are obtained for a number of clusters of 6 (Figures 1 and 2), whereas for the bisecting k-means algorithms the best scores are obtained generating 5 clusters. We therefore decided to set the number of clusters to 6. The Elbow method did not identify a very clear inflection point and was therefore not taken into account when determining the ideal number of clusters (See Appendix).

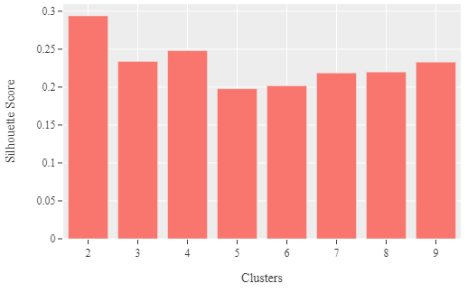

Fig. 3 Silhouette scores for clusters by number of clusters generated by the bisecting k-means algorithm.

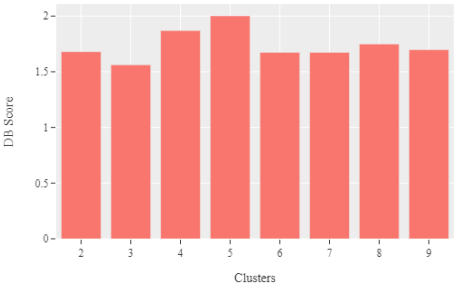

Fig. 4 Davies-Bouldin for clusters by number of clusters generated by the bisecting k-means algorithm.

5.1.1 Selection of DBSCAN hyperparameters

The DBSCAN algorithm, because of the way it defines clusters, does not allow the number of clusters to be defined a priori. Instead, the algorithm creates clusters based on density and requires the parameters `min_samples` and `eps` to be declared. `min_samples` defines the minimum number of neighbors for a data point to be considered a central data point and `eps` defines the distance between these neighbors and the central data point. The central points and their neighbors constitute a cluster.

We chose to obtain the same number of clusters as for the other algorithms. To do so, we performed a grid search of the parameters `min_samples` and `eps`. With the number of clusters, we calculated the silhouette score and the Davies Bouldin score for each iteration (see Table 1).

Based on the results of the grid search, we selected the DBSCAN hyperparameters `min_samples` = 5000 and `eps` = 1. This algorithm produces 6 clusters plus outliers (number of clusters = 7).

Table 1 Grid search of DBSCAN hyperparameters and the resulting Davies-Bouldin and Silhouette scores

`min_samples`	`eps`	Davies-Bouldin score	Silhouette score	Number of clusters
1000	0.75	1.43	0.09	16
1000	1	1.53	0.11	17
1000	1.5	2.16	0.11	16
2500	1	1.38	-0.03	16
2500	0.75	1.46	0.02	10
2500	1	1.58	0.11	11
2500	1.5	2.16	0.14	4
4000	1	1.53	0.07	11
4500	1	1.52	0.06	10
4750	1	1.51	0.07	9
4850	1	1.51	0.07	9
4900	1	1.51	0.07	9
4950	1	1.54	0.06	8
5000	0.5	1.32	-0.12	7
5000	0.75	1.42	-0.01	7
5000	1	1.54	0.05	7
5000	1.5	2.33	0.18	3

5.2 Clustering analysis

In analysing the clusters generated by the unsupervised machine learning algorithms that we selected, we opted to generally reproduce the analysis of Aouedi et al.[3] to ensure comparability of results. In the following section we outline the most meaningful results for each algorithm and our analysis and interpretation of their performance. The complete analysis of the results is available as an annex.

5.2.1 DBSCAN

We begin the analysis from the clusters generated by the DBSCAN algorithm. The algorithm produced six clusters which, in total, correspond to 40.2% of the data points. The remaining 59.8% are classified as outliers.

The distribution of clusters across port numbers reveals that clusters 2 and 3 represent flows to port 3128, clusters 0 and 4 represent flows to port 443 and clusters 1 and 5 represent flows to other ports. In other words, the destination port number determines membership to one cluster or another, with no cluster overlapping with more than one port type. The flows destined to ports 80 and 0 are classified as outliers.

Because of the two different clusters representing flows to ports 3128, 443, and "other", we looked for other characteristics to distinguish membership to one cluster or another. It is immediately obvious that the second characteristic determining cluster membership is the date and time of the flow. For example, among the clusters representing flows to port 3128, cluster 2 contains flows dated April 28 and earlier and cluster 3 contains flows after this date. A similar distinction separates clusters 0 and 4, and 1 and 5.

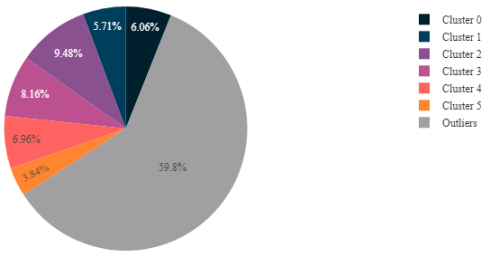

Fig. 5 Distribution of the datapoints across the clusters generated by DBSCAN.

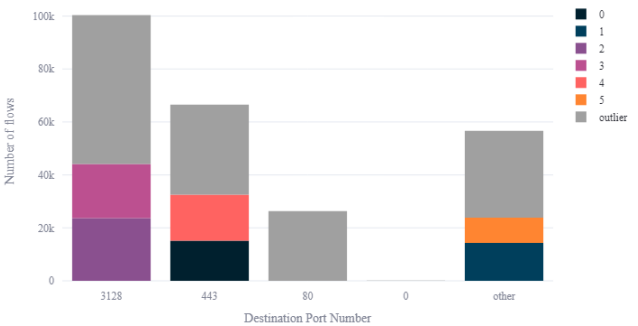

Fig. 6 Distribution of DBSCAN generated clusters across destination ports.

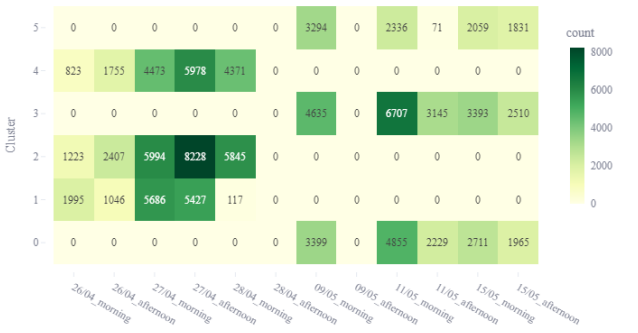

Fig. 7 Distribution of DBSCAN generated clusters across timepoints.

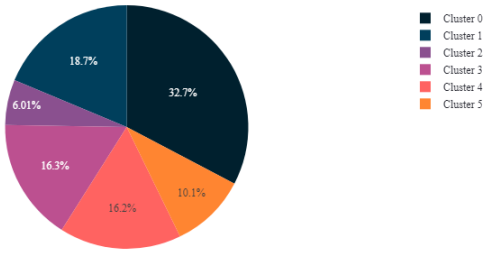

Fig. 8 Distribution of the datapoints across the clusters generated by k-modes.

We can therefore conclude that the algorithm focused only on the Destination.Port and Timestamp features to classify the data points, completely ignoring all other features. This is evidenced by the distribution of data points across all other features: the clusters are similarly distributed across all values of the remaining features.

5.2.2 k-modes

The k-modes algorithm divides the data points into 6 clusters representing 32.7%, 18.7%, 16.3%, 16.2%, 10.1%, and 6.01% of the total sample respectively.

The k-modes algorithm is better suited for handling one-hot encoded categorical variables. Therefore, we first analyzed how cluster membership was distributed between destination and source ports, which constitute the categorical variables of the dataset.

Flows to port 3128 appear to belong exclusively to clusters 0 and 5. Flows to port 443 belong primarily to clusters 4 and 1, with a small percentage (1.5% of flows to port 443) belonging to cluster 0. Flows to port 80 are assigned to clusters 0, 1, and 4. Finally, the flows to the other ports belong to clusters 2 and 3 and, in a small part to cluster 0 and 5 (0.7% and 0.2% of flows to other ports 443).

Flows from ports 443 and 80 belong mostly to cluster 2 (99.9% and 98.3%). Cluster 3 constains flows from port 3128 and consists of the vast majority of flows from this port (97%). Finally, flows to other ports belong mostly to clusters 0, 1, 4 and 5 (99.7%).

Therefore the algorithm identifies a group of flows from port 3128 to other ports (cluster 3), another group of flows from port 443 and 80 to other ports (cluster 2), two clusters destined to port 3128 (clusters 0 and 5), and two other going to port 443 (clusters 1 and 4).

An analysis of the partition of the clusters across applications confirms that datapoints of cluster 3 represent the majority of HTTP and HTTP Proxy applications. Cluster 2 mostly represents Google and HTTP applications, but

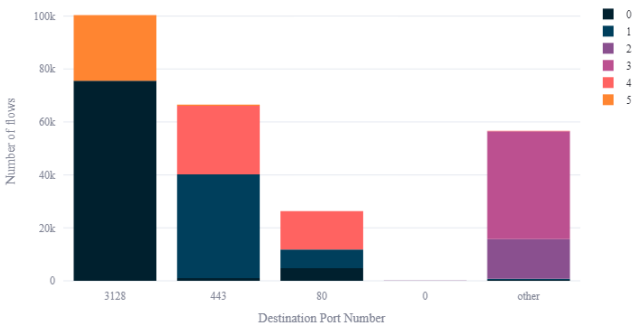

Fig. 9 Distribution of k-modes generated clusters across destination ports.

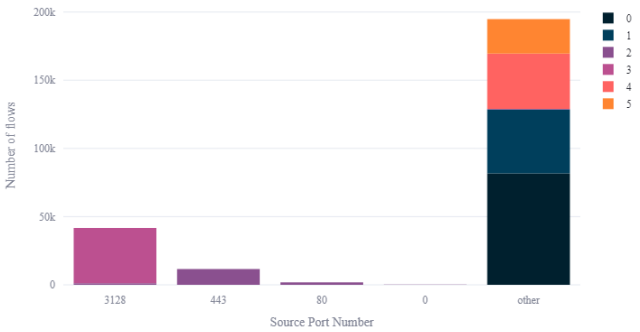

Fig. 10 Distribution of k-modes generated clusters across source ports.

interestingly also the majority of Dropbox fluxes. Clusters 0 and clusters 5 together represent the majority of Google, HTTP Proxy, and HTTP connect applications. Cluster 1 represents mostly SSL and Google fluxes.

5.2.3 k-means

The k-means algorithm divides the data points into seven clusters representing respectively 43.3%, 26.6%, 21.8%, 6.26%, 3.54%, and 0.0064% of the total sample.

We analyzed each cluster individually in order to better understand the clustering trends by k-means.

Clusters 2 and 4 together represent 66.5% of the flows and share similar characteristics. They both represent short and medium flows sending short and medium packets. Packets in the forward direction are small. The backwards packets are short and medium. These characteristics are compatible with

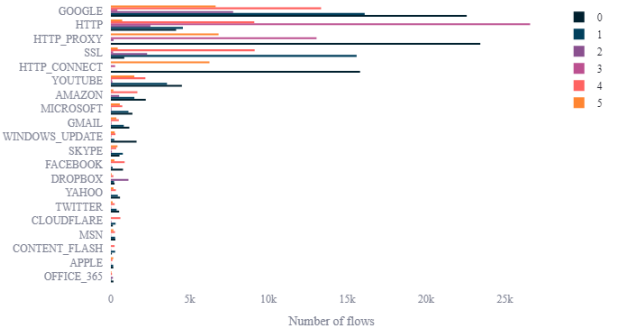

Fig. 11 Distribution of k-modes generated clusters across applications.

browser applications like Google, Amazon, etc. We expect browsing traffic to constitute a large portion of the flows in the sample.

Clusters 1 and 3 constitute 9.5% of flows. They are both characterized by long and medium flows with high maximum inter-packet arrival time (IAT) and large packets in the forward direction. This traffic could correspond to data-intensive applications such as file transfers, backups, or large data downloads.

Cluster 0 (21.8% of flows) contains the majority of the long flows and exclusively flows with a large maximum IAT. The length of packets in both backwards and forwards directions is short and medium. The fact that the cluster contains the majority of flows to applications like Dropbox suggests that this type of cluster contains at least some download traffic. This type of traffic is also typical of streaming applications

Cluster 5 is a very small cluster (0.0064% of flows) that contains medium and long forward flows. This cluster is also characterized by a large maximum IAT and large forward packet lengths. Like for clusters 1 and 3 this type of flow may correspond to data-intensive applications.

We can conclude that this clustering algorithm did not rely too heavily on categorical features such as port class and was successful in capturing the differences in flow type in the sample.

5.2.4 Bisecting k-means

The bisecting k-means algorithm divides the data points into six clusters representing respectively 28.5%, 25.3%, 21.6%, 20.2%, 2.54%, and 1.88% of the total sample.

Cluster 0, the largest cluster with 28.5% of flows, represents flows mostly destined to port 3128. The flows are short and medium in duration with no other distinctive pattern in maximum IAT, packet length and bytes sent in the initial window. It represents a large portion of HTTP and Google traffic and it can be assumed to represent browsing traffic.

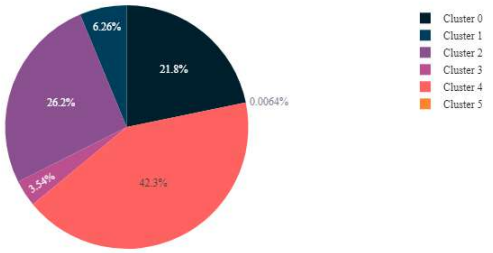

Fig. 12 Distribution of the datapoints across the clusters generated by k-means.

Clusters 1 and 2 (45.5% of flows) contain similar flows: in both, flows are short and medium in duration and are directed to ports in class 443, 80 and "other". These flows are classified as SSL, HTTP and Google applications suggesting mostly browsing activity like for cluster 0. The feature distinguishing flows in the two clusters is the timestamp: cluster 1 contains flows dating 09/05 and after, while cluster contains flows dating before such date. Therefore, the algorithm has used this feature to bisect flows in the two clusters.

Cluster 5 (21.6% of flows) contains Google and HTTP Proxy flows destined to all ports. These flows have a long duration and a high maximum IAT sending short and medium packets. This cluster is similar to cluster 0 generated by k-means and may contain streaming applications flows.

Clusters 3 and 4 together represent a small portion of the sample (4.42%). Both clusters contain long flows with a high maximum IAT. Flows in cluster 3 are mostly destined to ports classified as "other" while flows in cluster 4 are destined mostly to ports 3128, 443, and 80. The two clusters also differ in packet length, both forward and backward: cluster 3 contains smaller backward and longer forward packets while cluster 4 shows the opposite trend. Both clusters don't display belonging to a particular type of applications but are likely to characterize data transfer flows.

6 Discussion

In this work we compared the performance of four clustering algorithms on network flows data to establish which algorithm is best suited for network splicing. We found that the four algorithms can classify the same network flow data with very different results. Two clustering algorithms, DBSCAN and k-mode, give more importance to categorical data to separate the flows into clusters. The k-mode clustering algorithm focuses solely on port classes to group data. This algorithm, which is designed to handle "one-hot encoded" categorical data, appears to prioritize this type of data over numerical data in mixed type

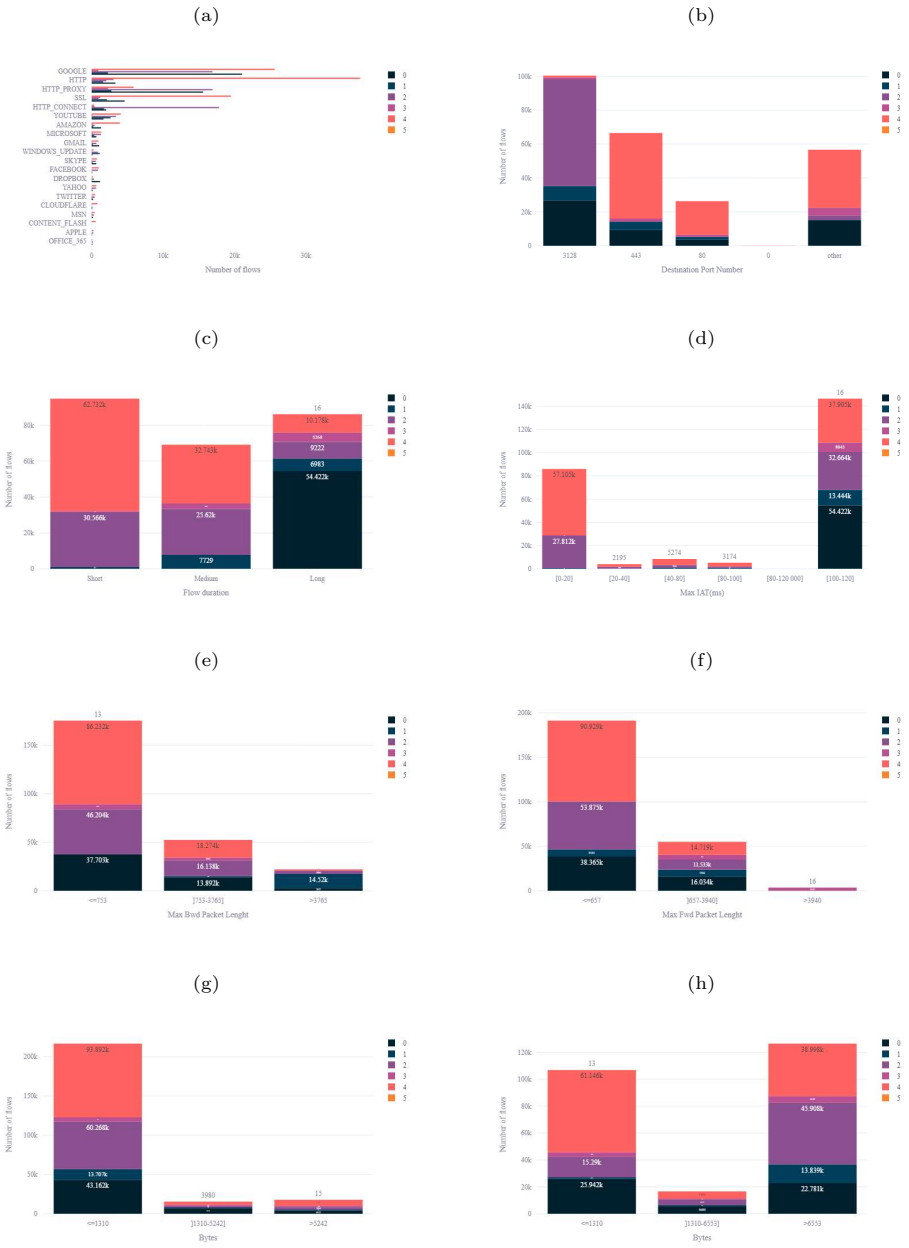

Fig. 13 (a) Distribution of k-means generated clusters across applications. (b) Distribution of k-means generated clusters across destination ports. (c) Distribution of k-means generated clusters across flow durations. (d) Distribution of k-means generated clusters as a function of maximum inter-arrival time (IAT) duration. (e) Distribution of k-means generated clusters as a function of the maximum backwards packet length. (f) Distribution of k-means generated clusters as a function of the maximum forward packet length. (g) Distribution of k-means generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (h) Distribution of k-means generated clusters as a function of the total number of bytes sent in initial window in the forward direction.

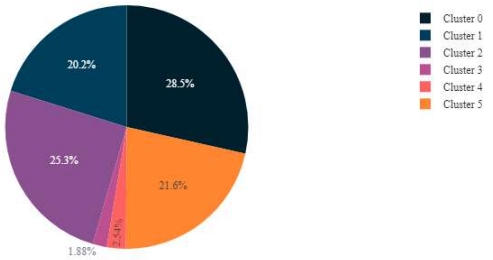

Fig. 14 Distribution of the datapoints across the clusters generated by bisecting k-means.

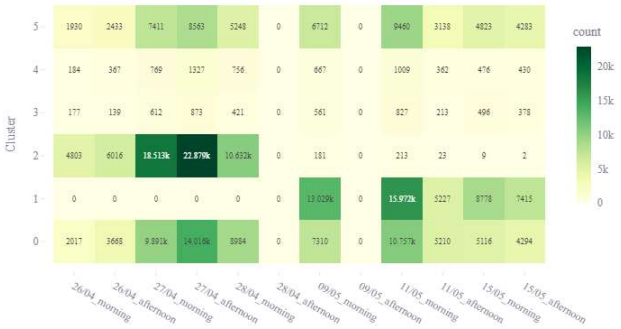

Fig. 15 Distribution of bisecting k-means generated clusters across timepoints.

datasets. We therefore conclude that the algorithm is not suitable for clustering network data. The DBSCAN algorithm, on the other hand, focused on port number classes and flow timestamps to separate the data into clusters, while the rest of the data variance is still represented by the outliers in the dataset (about 60% of the total sample). In general, the DBSCAN algorithm can be challenging to optimize and, of all the algorithms applied in this work, it is the most computationally intensive. We do not entirely reject the use of this algorithm, but we advise using it with some caveats. For example, we suggest excluding or penalizing categorical data. Also, temporal data should be excluded or preprocessed to capture the daily and weekly periodicity of some network flow events.

The k-means and bisecting k-means algorithms perform best in our comparison. The two algorithms are closely related and we detected similarities between the clusters generated by the two algorithms. We tried to interpret their clustering results to detect application patterns (like browsing and data

Fig. 16 (a) Distribution of bisecting k-means generated clusters across applications. (b) Distribution of bisecting k-means generated clusters across destination ports. (c) Distribution of bisecting k-means generated clusters across flow duration. (d) Distribution of bisecting k-means generated clusters as a function of maximum inter-arrival time (IAT) duration. (e) Distribution of bisecting k-means generated clusters as a function of the maximum backwards packet length. (f) Distribution of bisecting k-means generated clusters as a function of the maximum forward packet length. (g) Distribution of bisecting k-means generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (h) Distribution of bisecting k-means generated clusters as a function of the total number of bytes sent in initial window in the forward direction.

transfers). The goal is not to understand the nature of the clustering in detail, but to be able to conclude whether the patterns detected by the algorithms are similar to actual patterns of network usage. These two algorithms not only produced more meaningful clusters, but they are also the least computationally intensive and can more easily scale to datasets containing millions of streams. Finally, both algorithms allow the number of clusters to be defined a priori. This is advantageous when the number of slices in a network needs to be predefined or the number of available slices is less than the optimal number defined by the Silhouette score or other metrics.

One difference between the two is that the bisecting k-means algorithm seems to place more importance on the timestamp information in the dataset than k-means and splits otherwise similar streams on their timestamp only. As in the use of the DBSCAN algorithm, a better way to handle timestamp information would be to extract the day of the week or time of day rather than directly converting timestamps into a numeric variable. Because of this last observation, we conclude that the k-means algorithm is the best clustering algorithm for network data.

7 Conclusion

The aim of this work was to establish a comparison between clustering algorithms for network slicing applications. We were able to confirm that the k-means algorithm is the better suited among the ones we tested for clustering network flow data. In the process, we were also able to improve the preprocessing of the data compared to previous work and we identified good practices that can be applied in future works. Among these is the careful preprocessing of temporal information as it can heavily bias clustering without contributing to a meaningful clustering of flows. For future work, our observations should be tested against different data originating from different networks. Our working dataset captured the activity in a university network, whose usage patterns may not be reflected in different networks such as public networks or private company networks. A comparison between the clustering performances of the k-means algorithm in the presence and absence of categorical features, like the port number, should be conducted to assess the utility of this information and to evaluate alternative preprocessing practices for this type of features. Finally, network slicing based on unsupervised machine learning techniques should be implemented to validate the efficacy of this approach.

Supplementary information. Supplementary information is available as an appendix.

References

[1] Ayoubi, S., Limam, N., Salahuddin, M.A., Shahriar, N., Boutaba, R., Estrada-Solano, F., Caicedo, O.M.: Machine learning for cognitive network management. IEEE Communications Magazine **56**(1), 158–165

(2018)

[2] Barlow, H.B.: Unsupervised learning. Neural computation **1**(3), 295–311 (1989)

[3] Aouedi, O., Piamrat, K., Hamma, S., Perera, J.M.: Network traffic analysis using machine learning: an unsupervised approach to understand and slice your network. Annals of Telecommunications, 1–13 (2021)

[4] Karagiannis, T., Broido, A., Faloutsos, M., Claffy, K.: Transport layer identification of p2p traffic. In: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, pp. 121–134 (2004)

[5] Moore, A.W., Papagiannaki, K.: Toward the accurate identification of network applications. In: Passive and Active Network Measurement: 6th International Workshop, PAM 2005, Boston, MA, USA, March 31-April 1, 2005. Proceedings 6, pp. 41–54 (2005). Springer

[6] Nguyen, T.T., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. IEEE communications surveys & tutorials **10**(4), 56–76 (2008)

[7] Auld, T., Moore, A.W., Gull, S.F.: Bayesian neural networks for internet traffic classification. IEEE Transactions on neural networks **18**(1), 223–239 (2007)

[8] Este, A., Gringoli, F., Salgarelli, L.: Support vector machines for tcp traffic classification. Computer Networks **53**(14), 2476–2490 (2009)

[9] Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.-K.R., Sarsour, M.: Network traffic classification: Techniques, datasets, and challenges. Digital Communications and Networks (2022)

[10] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Inc., ??? (1988)

[11] Wang, Y., Xiang, Y., Zhang, J., Yu, S.: A novel semi-supervised approach for network traffic clustering. In: 2011 5th International Conference on Network and System Security, pp. 169–175 (2011). IEEE

[12] Du, Y., Zhang, R.: Design of a method for encrypted p2p traffic identification using k-means algorithm. Telecommunication Systems **53**, 163–168 (2013)

[13] Erman, J., Arlitt, M., Mahanti, A.: Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, pp. 281–286 (2006)

[14] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *et al.*: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd, vol. 96, pp. 226–231 (1996)

[15] Cheeseman, P.C., Stutz, J.C., *et al.*: Bayesian classification (autoclass): theory and results. Advances in knowledge discovery and data mining **180**, 153–180 (1996)

[16] Singh, H.: Performance analysis of unsupervised machine learning techniques for network traffic classification. In: 2015 Fifth International Conference on Advanced Computing & Communication Technologies, pp. 401–404 (2015). IEEE

[17] Moon, T.K.: The expectation-maximization algorithm. IEEE Signal processing magazine **13**(6), 47–60 (1996)

A Appendix

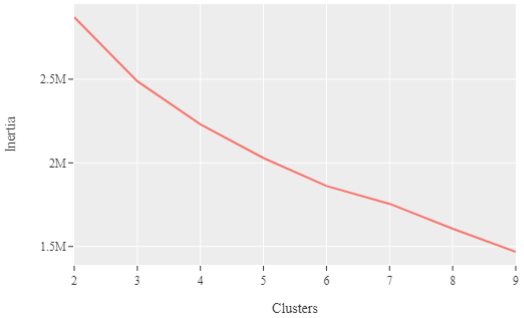

Fig. A.1 Inertia scores by number of clusters generated by the k-means algorithm.

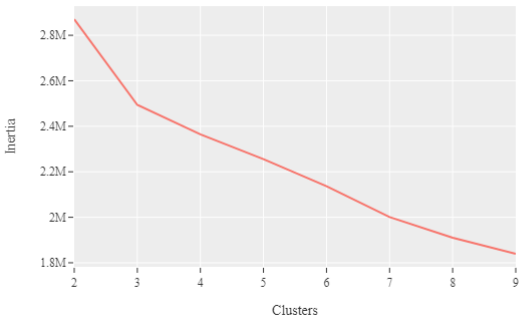

Fig. A.2 Inertia scores by number of clusters generated by the bisecting k-means algorithm.

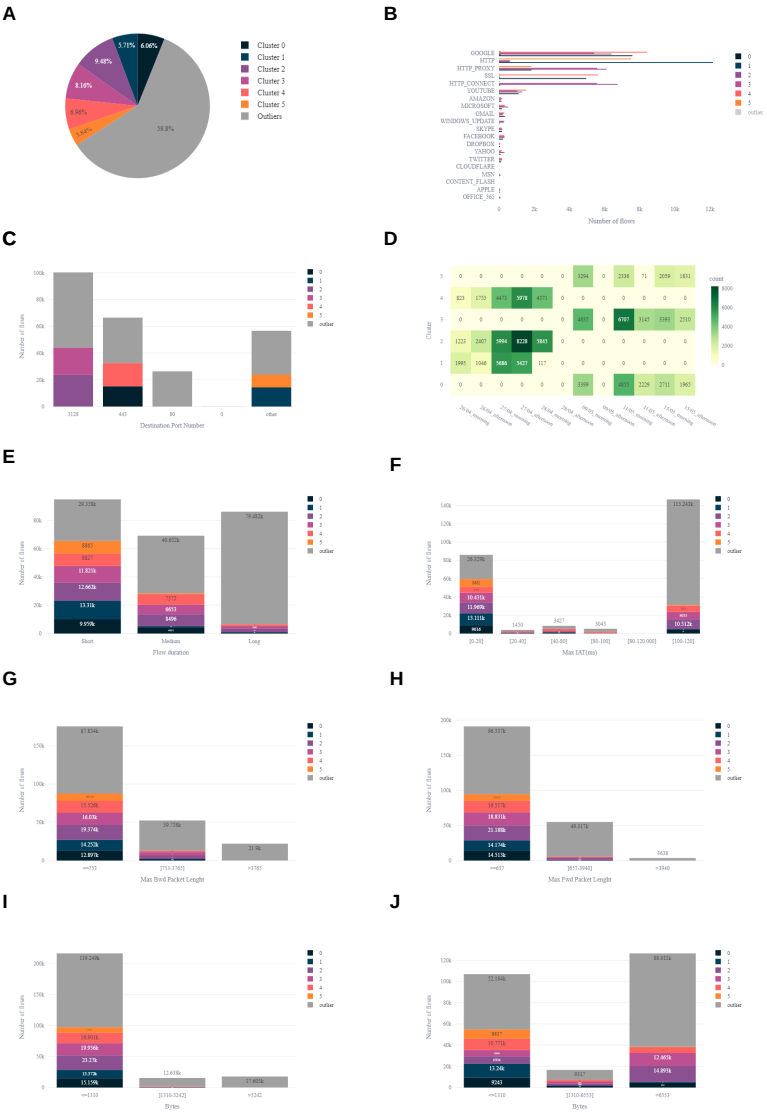

Fig. A.3 (a) Distribution of the datapoints across the clusters generated by DBSCAN. (b) Distribution of DBSCAN generated clusters across applications. (c) Distribution of DBSCAN generated clusters across destination ports. (d) Distribution of DBSCAN generated clusters across timepoints. (e) Distribution of DBSCAN generated clusters across flow duration. (f) Distribution of DBSCAN generated clusters as a function of maximum inter-arrival time (IAT) duration. (g) Distribution of DBSCAN generated clusters as a function of the maximum backwards packet length. (h) Distribution of DBSCAN generated clusters as a function of the maximum forward packet length. (i) Distribution of DBSCAN generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (j) Distribution of DBSCAN generated clusters as a function of the total number of bytes sent in initial window in the forward direction.

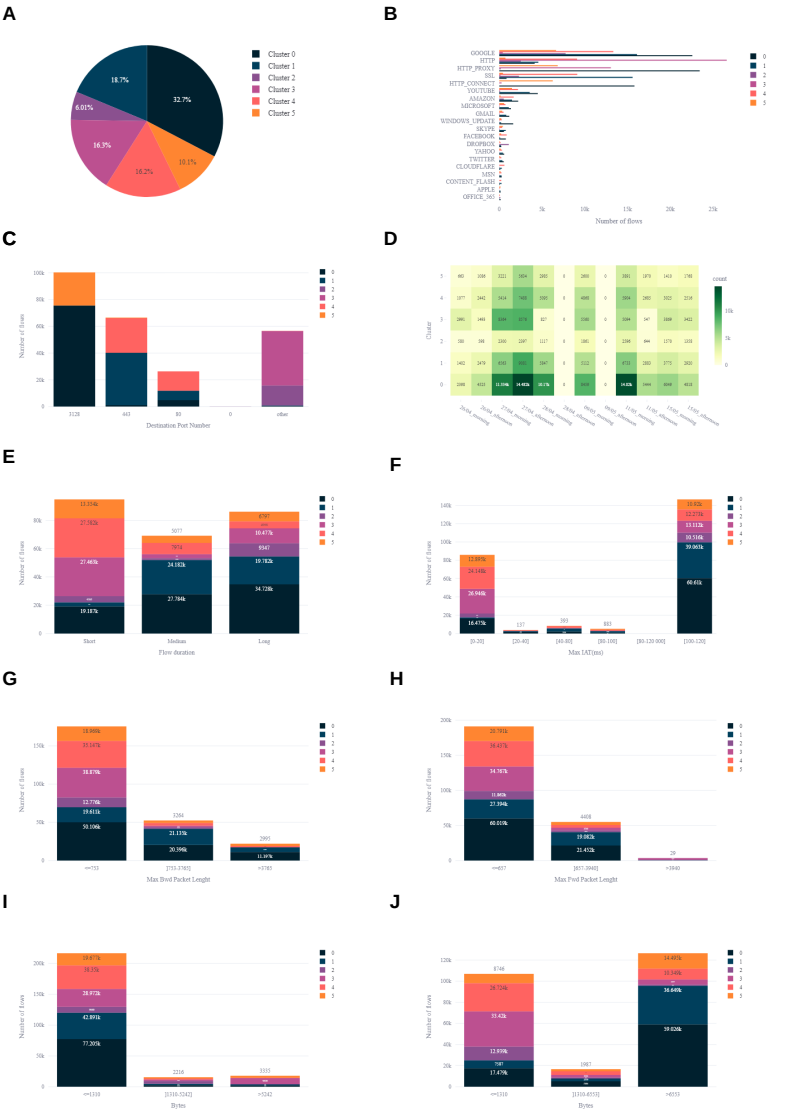

Fig. A.4 (a) Distribution of the datapoints across the clusters generated by k-modes. (b) Distribution of k-modes generated clusters across applications. (c) Distribution of k-modes generated clusters across destination ports. (d) Distribution of k-modes generated clusters across timepoints. (e) Distribution of k-modes generated clusters across flow duration. (f) Distribution of k-modes generated clusters as a function of maximum inter-arrival time (IAT) duration. (g) Distribution of k-modes generated clusters as a function of the maximum backwards packet length. (h) Distribution of k-modes generated clusters as a function of the maximum forward packet length. (i) Distribution of k-modes generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (j) Distribution of k-modes generated clusters as a function of the total number of bytes sent in initial window in the forward direction.

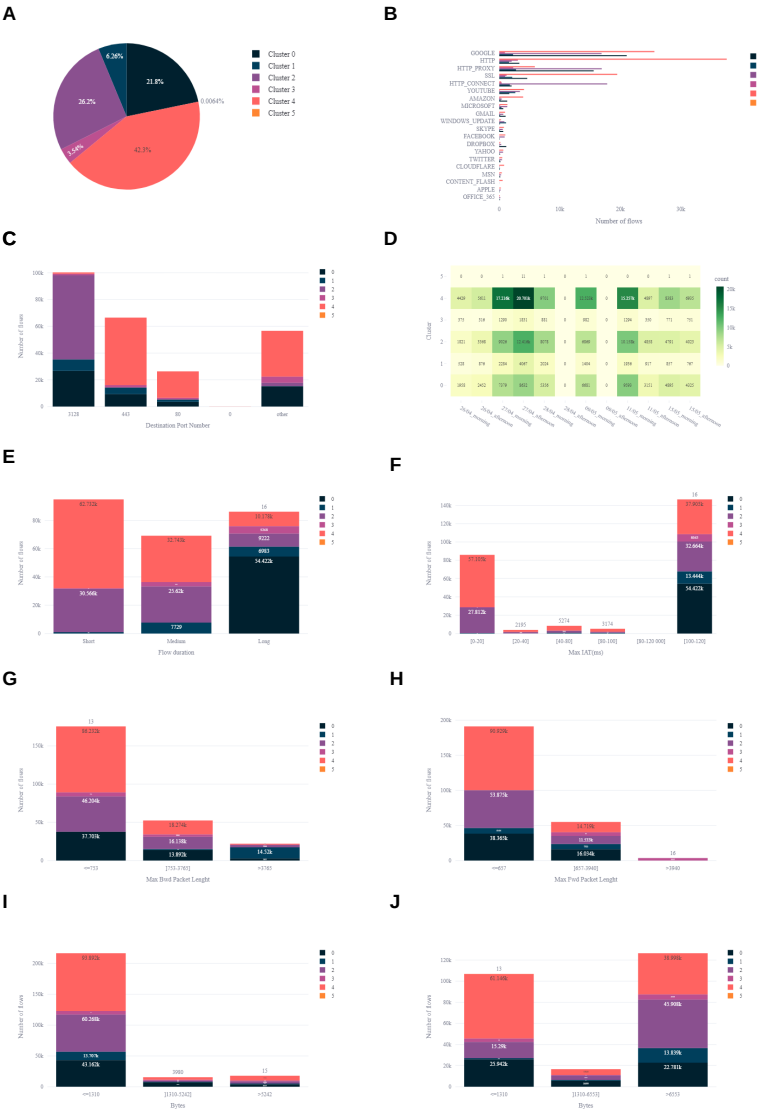

Fig. A.5 (a) Distribution of the datapoints across the clusters generated by k-means. (b) Distribution of k-means generated clusters across applications. (c) Distribution of k-means generated clusters across destination ports. (d) Distribution of k-means generated clusters across timepoints. (e) Distribution of k-means generated clusters across flow duration. (f) Distribution of k-means generated clusters as a function of maximum inter-arrival time (IAT) duration. (g) Distribution of k-means generated clusters as a function of the maximum backwards packet length. (h) Distribution of k-means generated clusters as a function of the maximum forward packet length. (i) Distribution of k-means generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (j) Distribution of k-means generated clusters as a function of the total number of bytes sent in initial window in the forward direction.

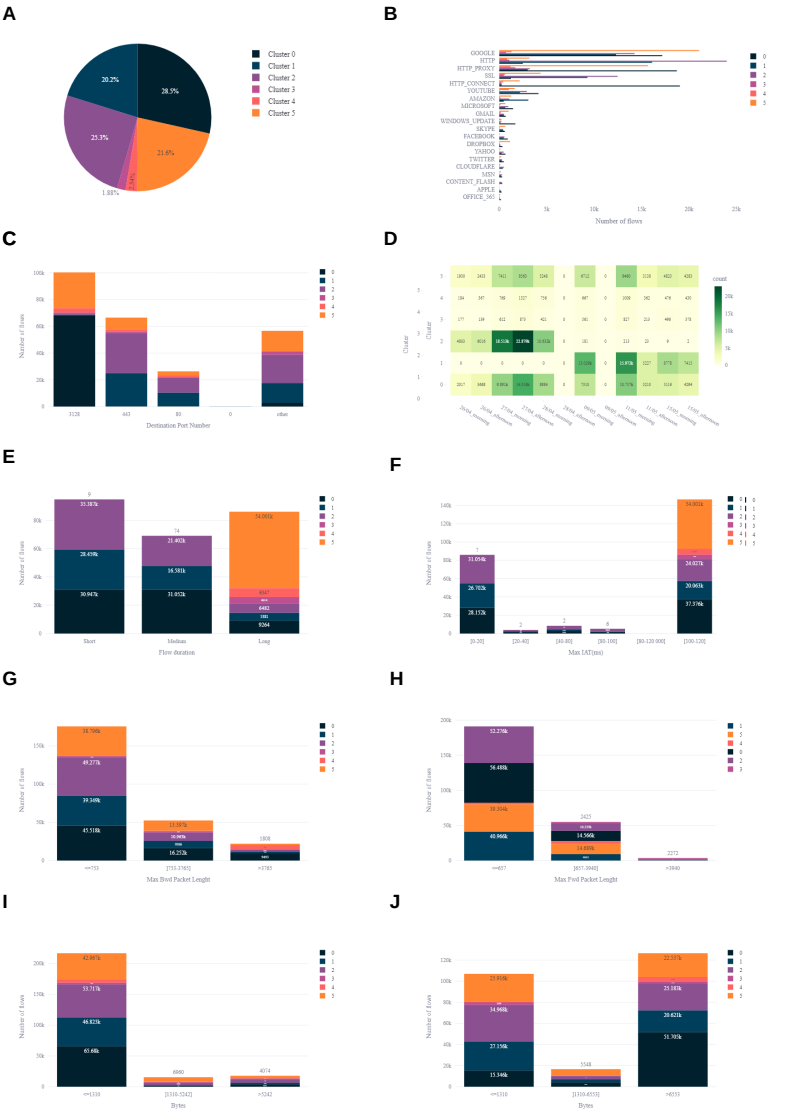

Fig. A.6 (a) Distribution of the datapoints across the clusters generated by bisecting k-means. (b) Distribution of bisecting k-means generated clusters across applications. (c) Distribution of bisecting k-means generated clusters across destination ports. (d) Distribution of bisecting k-means generated clusters across timepoints. (e) Distribution of bisecting k-means generated clusters across flow duration. (f) Distribution of bisecting k-means generated clusters as a function of maximum inter-arrival time (IAT) duration. (g) Distribution of bisecting k-means generated clusters as a function of the maximum backwards packet length. (h) Distribution of bisecting k-means generated clusters as a function of the maximum forward packet length. (i) Distribution of bisecting k-means generated clusters as a function of the total number of bytes sent in initial window in the backwards direction. (j) Distribution of bisecting k-means generated clusters as a function of the total number of bytes sent in initial window in the forward direction.