A Model for Network Traffic Classification of 5G Networks for QoS Improvement

Mohammed Aqeel Ismail and Okuthe P. Kogeda *

Posted Date: 29 May 2026

doi: 10.20944/preprints202605.2006.v1

Keywords: 5G networks; network traffic classification; Quality of Service (QoS); statistical features; supervised learning; random forest algorithm; machine learning; MIRAGE2019

 doi:10.20944/preprints202605.2006.v1

Article

A Model for Network Traffic Classification of 5G Networks for QoS Improvement

Mohammed Aqeel Ismail and Okuthe P. Kogeda *

School of Agriculture and Science, College of Agriculture, Engineering and Science, University of KwaZulu-Natal, Pietermaritzburg Campus, Pietermaritzburg 3201, Republic of South Africa

* Correspondence: kogedao@ukzn.ac.za

Abstract

The increasing complexity and volume of network traffic in modern 5G network environments pose significant challenges to maintain consistent Quality of Service (QoS) across diverse applications. QoS in 5G networks ensures efficient resource allocation, minimal latency, high throughput, and reliable connectivity. However, without accurate network traffic classification, QoS cannot be effectively optimized. This is because different applications have unique performance requirements. Traditional classification techniques such as port-based identification and Deep Packet Inspection (DPI) have become inadequate due to widespread encryption, port masquerading, and privacy concerns. This paper presents a supervised learning-based approach for network traffic classification specifically aimed at QoS optimization in 5G networks. A Random Forest classifier was implemented using flow level statistical features extracted from the MIRAGE-2019 dataset. The system includes an intelligent QoS policy mapping that categorizes applications according to their network requirements including priority levels, bandwidth needs, latency sensitivity, and jitter tolerance. Experimental results demonstrated 89.76% classification accuracy and 0.897 macro F1-score using Random Forest with RandomOverSampler, confirming the model's effectiveness in accurately classifying encrypted traffic and supporting QoS optimization.

Keywords: 5G networks; network traffic classification; Quality of Service (QoS); statistical features; supervised learning; random forest algorithm; machine learning; MIRAGE2019

1. Introduction

The global independence on internet connectivity has grown exponentially, reshaping how individuals and organizations operate, communicate and access services. This shift is largely driven by advances in networking technologies, most notably, 5G (Fifth Generation) of wireless network technology that succeeds the 4G LTE standard (Azab et al., 2024).

5G introduces transformative capabilities through enhanced mobile broadband (eMBB), ultrareliable low-latency communications (URLLC), and massive machine-type communications (mMTC) that were not previously offered by its predecessors (Maran on and Armando, 2023). 5G's key features include the use of higher radio frequency waves that the previous generation's wireless technologies did not offer. This allows for users to expect greater capacity, faster data transfer rates, and stable connections but also requires that more network infrastructure to be deployed to ensure reliable coverage that supports new applications.

As the number of users and applications grow, with 5G promises to offer significant improvements in network speed and connectivity, the complexity and volume of network traffic increase as well. This creates a challenge to ensure that each application receives the appropriate level of network resources for 5G network technologies to perform optimally. For instance, video conferencing and online gaming require low latency and high reliability, whereas services like software updates are more tolerant to delay (Canever and Wang, 2023). Without an intelligent mechanism to distinguish between these traffic types, network resources may be allocated

 doi:10.20944/preprints202605.2006.v1

inefficiently, leading to Quality of Service being degraded. This is where the application of network traffic classification and QoS come into play.

Network Traffic Classification is a growing domain within Computer Science. Network traffic represents the amount of data transmitted over a network over a given time, for communication and exchange of information between interconnected devices within a network. In the context of Internet Service Providers (ISPs) and network operators, (Mahmood et al., 2023; Mohammed and Ghani, 2025) the accurate identification and categorization of network traffic according to application type is an important element of many network management tasks (Erman et al., 2006).

Accurate classification allows network operators to understand network behaviour, which is a critical step for network security, to detect malicious behaviour, efficient resource management and enforce Quality of Service (QoS) policies tailored to the applications needs (Ganesan et al., 2021). QoS refers to the performance level of a service. QoS in 5G networks ensures that critical services are prioritized, bandwidth is allocated effectively, and latency-sensitive applications receive fast and stable connections in 5G environments (Karim et al., 2019).

It helps to minimize the issues of packet loss (packets lost during transmission), latency (delay in data transmission) and jitter (inconsistencies in delay from one packet to the next). But this will only be possible if the network will be able to identify what type of application is generating the traffic for the application to receive the necessary resources and performance, in essence optimize QoS.

Historically, traffic classification relied on methods such as port-based identification and Deep Packet Inspection (DPI). While simple and effective in the past, these techniques have become obsolete in many modern scenarios due to widespread encryption, port masquerading, port randomization, and privacy regulations. As a result, they fail to accurately identify traffic types or support real-time QoS enforcement in today's networks.

To address these limitations, recent research has turned to Machine Learning classification. Machine Learning models can analyse flow-level statistical features, such as packet size, flow durations, inter-arrival time, and byte count-without the need to inspect the packet payload. Among various algorithms, Random Forest stands out for its high classification accuracy, robustness to noise, and ability to rank features by importance, which is particularly useful for designing QoS policies.

Accordingly, this study addresses the main research question: How can Supervised Machine Learning be designed and evaluated to accurately classify 5G network traffic for the improvement of Quality of Service?

To answer this, this study focuses on the implementation of Random Forest classification system to not only classify mobile application traffic from the MIRAGE-2019 dataset, but also on mapping results to QoS enforcement policies, thereby enabling QoS-aware resource allocation in next generation mobile networks.

The remainder of this paper is organised as follows. In Section 2, we provide literature review and the research gap being addressed in this paper. In Section 3, we provide the Methodology. In Section 4, the results along with testing and evaluation are discussed. In Section 5, provides the practical implications. In Section 6, we provide the conclusion and future work areas of considerations.

2. Literature Review

Traditional traffic identification methods such as port-based detection and Deep Packet Inspection (DPI) were once effective but now struggle with modern traffic, have been gradually replaced by Machine Learning approaches, including supervised, unsupervised, and semi-supervised learning.

2.1. Port-Based and DPI Methods

The earliest techniques for network traffic classification relied on port-based classification. Port-based classification identified application types based on the port numbers associated with network

 doi:10.20944/preprints202605.2006.v1

packets. It will check the source or destination port of the packet headers of transport layer protocols, namely, TCP (Transmission Control Protocol) and UPD (User Datagram Protocol) (Nguyen and Armitage, 2008) and identify the application type based on this mapping. This approach was initially effective as the Internet Assigned Numbers Authority (IANA) registered "well-known port numbers" (Mahmood et al., 2023, p. 1135) to many services. The traffic is identified by the registered port number the application is registered to (Shafiq et al., 2016). For example, E-mail applications use 25 (SMTP) port number to send emails and to receive email 110 (POP3) port is used. And web applications use 80 ports number which is HTTP (Hypertext Transfer Protocol).

Port-based classification has become increasingly unreliable since modern applications often use dynamic or non-standard ports and Peer-to-peer (P2P) applications often disguise their traffic by operating on common port numbers to avoid detection, such as HTTP. As a result, port-based methods are prone to both misclassification and incomplete identification (W.Moore and Papagiannaki, 2005).

For instance, researchers have evaluated that port-based classification could rarely identify 70% of application traffic. Their study noted both underestimation and overestimating of classes, highlighting that "…traffic flows can also be misidentified…" (W.Moore and Papagiannaki, 2005). Similar limitations are noted in 2006 paper (Madhukar and Williamson, 2006) where port-based classification proved to be ineffective in identifying 30%-70% of P2P traffic, primarily due its reliance on standard ports.

Kind et al. (2006) also emphasized that "Knowing the destination IP addresses and port numbers is not sufficient for identifying the application associated with a traffic flow, as ports can be dynamically chosen," (Kind et al., 2008, p. 6). These limitations have led to the decline of port-based classification in modern traffic classification systems.

To address the limitations of port-based classification, Deep Packet Inspection (DPI) emerged as a more accurate alternative by inspecting payload content of packets rather than relying solely on headers or port numbers (Liao et al., 2012). DPI systems compare packet payloads against predefined applications signatures, enabling precise identification of traffic, even in complex protocols. The DPI process involves inspecting packet data for known strings, patterns or byte sequences and matching these against a signature library (Azab et al., 2024).

To improve the efficiency of DPI, especially in resource-constrained environments like edge routers or campus networks, Fernandes et al. (2009) proposed Lightweight Deep Packet Inspection (LW-DPI). Their framework focuses on reducing the computational and memory overhead of traditional DPI while retaining acceptable classification accuracy.

Another significant advancement came with the introduction of the length-based Matching (LBM) framework (Wang et al., 2011), which optimized regex pattern matching for high-speed packet processing. LBM employs a novel pattern matching engine called StriD2FA (Stride-DFA) to reduce the portion of the data stream that needs inspection, thus enhancing both speed and memory efficiency. However, this study shows that even optimized DPI has limitations with modern encrypted traffic.

Despite being more accurate than port-based classification, it comes with several drawbacks. DPI systems are computationally intensive, requiring significant memory and processing power. Moreover, DPI becomes increasingly ineffective since it cannot inspect encrypted payloads (Liao et al., 2012; Mahmood et al., 2023). Additionally, DPI raises privacy and legal concerns, as it involves inspecting user data, which may violate data protection regulations (Fernandes et al., 2009, p. 2; Azab et al., 2024).

These limitations have driven the shift toward statistical and Machine Learning classification methods. The methods adopt the use of statistical flow-level statistical features, which are not affected by encryption and provide a scalable alternative for traffic classification (Nguyen and Armitage, 2008).

 doi:10.20944/preprints202605.2006.v1

2.2. Supervised Learning

As encryption and port randomization became common, Supervised Machine Learning emerged as a dominant method for traffic classification. It has been extensively employed in network traffic classification due to its capacity to manage complex datasets and deliver high classification accuracy. Supervised learning is a type of Machine Learning where the model is trained using labelled data (data where the correct application/ service type is already known). From the labelled dataset the model learns a set of classification rules to predict unknown traffic (Mahmood et al., 2023; Nguyen and Armitage, 2008)

To show a few its outcomes; Shafiq et al. (2016) conducted an extensive comparative evaluation of four Machine Learning algorithms on flow level traffic datasets without payload, IP or port information. They split datasets into training and testing and measured accuracy and computational overhead. In the experiment, supervised learning algorithms such as Support Vector Machines (SVM) and Naïve Bayes algorithm achieved accuracy outcomes of 74,05% and 71,89% respectively (Shafiq et al., 2016).

A recent study (Mohammed and Ghani, 2025) found that algorithms like Random Forest, Naïve Bayes and K-Nearest Neighbours can be used for classification with each achieving a classification accuracy of more than 0.90 after implementing various other Machine Learning techniques like SMOTE. Their results claim that Machine Learning classifiers improved QoS optimization and bandwidth management. Their study is very closely related to what this paper has accomplished.

Ganesan et al. (2021) in their study addressed the challenges of managing and classifying network traffic in smart IoT environments. The authors used supervised learning models in statistical features extracted from traffic flows, testing performance across IoT-specific datasets. They found that Random Forest outperformed other models in terms of accuracy, precision, and execution time, making it ideal for smart and time-sensitive networks (Ganesan et al., 2021).

Azab et al. (2024) present a comprehensive survey of network traffic classification approaches, discussing traditional techniques as well as Machine Learning methods. Their work provides a foundational understanding of classification methods. They conclude that Machine Learning, particularly Supervised Machine Learning is built using statistical flow-level features, it is relatively easy to implement, offers high accuracy, uses minimal computational power, detects small grains, and does not require access to payload contents, making suitable for modern encrypted and heterogeneous traffic (Azab et al., 2024). It does, however, have drawbacks. The examples used in the training dataset have a significant impact on the classifier's performance to recognize applications. For instance, if there is a new application the model will not be able to identify it. For application to be classified the model will have to retrained with the new application.

2.3. Unsupervised Learning

Unsupervised learning is a useful complement to supervised learning especially when using unlabelled datasets. Unsupervised learning methods include the use of clustering algorithms to identify natural grouping in traffic. The goal is finding patterns, similarities, or groups in the data without being told what the data represents (Azab et al., 2024).

Erman et al. (2006) demonstrated the use of K-Means clustering to separate flows into application-based clusters using transport layer statistical features. They concluded that unsupervised learning could achieve meaningful application traffic separation without the need for payload inspection.

Canever and Wang (2023) in their study evaluate various unsupervised clustering algorithms for grouping network traffic flows based solely on statistical flow features, addressing scenarios where labelled data is unavailable. The authors selected clustering algorithms namely k-means, which is known for its simplicity and aims to classify data by dividing it into K clusters. They showed that unsupervised methods can detect unknown traffic types but often lack labelling accuracy, making it less accurate than supervised models (Canever and Wang, 2023).

 doi:10.20944/preprints202605.2006.v1

These techniques are particularly valuable when dealing with unknown or encrypted traffic types. However, unsupervised methods alone do not assign meaningful class labels, which limits their use in QoS enforcement unless paired with manual or rule-based interpretation. In a QoS context, one can assign policies per cluster, but human interpretation is still needed to map clusters to application types. Nevertheless, unsupervised clustering lays the groundwork for identifying traffic structure when labels are unavailable.

2.4. Semi-Supervised Learning

Semi-supervised learning combines the strengths of supervised and unsupervised techniques. This is especially relevant in 5G contexts, where new traffic types emerge regularly and labelling all data is impractical. It starts with unlabelled traffic and cluster it to generate useful structure or labels. Then, you train a supervised model on that processed data to classify future traffic flows.

Labayen et al. (2020) proposed a new three-layer model to classify captured traffic data by analysing user behaviour. Authors used a hybrid approach of unsupervised K-means clustering algorithm and supervised Random Forest to achieve real time classification across common user activities (web browsing, video streaming, VoIP). Even though the model achieved a mean accuracy 96,09%, their experiment shows that their approach has a significant impact on computational speed and memory usage, which led to longer training and testing times.

Nguyen and Armitage (2007) proposed a semi-supervised network traffic classification system that combined K-Means clustering and Naive Bayes classification using flow-level statistical features. Their system required only a small percentage of labelled data (as little as 10%) to accurately classify traffic flows. The study (Erman et al., 2007) highlighted that semi-supervised learning could reduce the dependency on data labelled datasets while still achieving high accuracy, outperforming traditional port-based classification methods.

Even though it provides high accuracy it requires higher computational resources than supervised and unsupervised solutions since it combines both unsupervised and supervised learning.

These findings consistently show that Machine Learning-based traffic classification, especially supervised learning and the use of statistical flow features, provide a practical and effective alternative to traditional methods. The only drawback of Supervised Machine Learning is obtaining a labelled dataset.

The drawback of a labelled dataset is solved by the open dataset, MIRAGE-2019 Dataset. The insights from the literature directly form the methodology for this project, which encourages the implementation a supervised learning using Random Forest-based classifier to optimize QoS in the MIRAGE-2019 dataset.

2.5. Research Gap

Research shows the effectiveness of Machine Learning in traffic classification but does not use the classifier outputs for actionable QoS policy implementations. Most literature prioritizes classification but do not offer the practical QoS policy applications. Here, the contribution of the study lies within the gap in literature by extending flow-based statistical features traffic classification with policy mapping that offers actionable QoS recommendations at the classifier output level.

3. Methodology

3.1. Dataset

A labelled dataset is crucial for Supervised Machine Learning because it provides the model with labelled features that will guide the learning process and enable performance evaluation (Azab et al., 2024).

Aceto et al. (2019) created the MIRAGE-2019 dataset used in this experiment. The MIRAGE-2019 dataset is a human generated dataset that was produced for mobile traffic analysis. It contains

 doi:10.20944/preprints202605.2006.v1

labelled bidirectional traffic flows generated by over 280 experiments using 40 mobile apps across 3 devices viz., Google Nexus 7, Xiaomi Mi5 and Samsung Galaxy A5. Over 280 students (ages 19-25) from three courses at the University of Napoli "Frederico II" voluntarily participated. Each student conducting one or two experimental sessions. The participants performed typical app usage activities, including exploring functionalities, first-time installation, login, and registration processes, to generate realistic traffic data. The dataset is made available in JSON format making it compatible and user friendly. The dataset gathers traffic generated by 40 android apps belonging to 16 different categories according to the Google Play apps distribution platform.

This study is based on only a sample of the dataset made available on Kaggle. It consists of data from 20 applications generated by Google Nexus and Xiaomi Mi5. The mobile-app traffic was collected for a period of 2 years (May 2017-May 2019) and openly released to the research community. This was done to ensure transparency and replicability in traffic analysis and to study traffic patterns as well as to examine models and perform classification tasks (Aceto et al., 2019).

The dataset was selected due to its relevance to modern mobile networking scenarios and its comprehensive labelling of different application types, particularly required for Supervised Machine Learning.

The proposal intended to use the 5G traffic Dataset (Choi et al., 2023), but after pre-processing and flow aggregation, the number of labelled flows were too few to train and evaluate a Supervised Machine Learning model. This could have compromised the model's ability to generalize and classify application types accurately.

The MIRAGE-2019 dataset includes traffic from diverse application categories shown in Table 1:

Table 1. Categories of applications in MIRAGE-2019 Dataset.

No.	Category	Application
1	Navigation	Waze
2	VoIP(Voice over Internet Protocol)/Message	Viber, Messenger
3	Gaming	Sliter.io
4	Music Streaming	Spotify
5	Video Streaming	YouTube
6	Social Media	Facebook, Twitter, Pinterest
7	E-Commerce	Wish, Subito, Groupon
8	Weather	Accuweather
9	Travel	TripAdvisor, FourSquare
10	Sports	Iliga
11	Cloud Storage	Dropbox
12	Productivity	Trello, Duolingo
13	Entertainment	Comics

3.2. Categories of Features in MIRAGE-2019

Each flow in the MIRAGE-2019 dataset is represented as a collection of statistical features derived from bidirectional packet streams. The dataset organizes these into feature categories, each describing different aspects of flow behaviour. For each category, MIRAGE-2019 contains statistical summaries such as minimum, maximum, mean, standard deviation and skewness.

These features capture protocol-level characteristics of network flows, allowing the model to learn fine-grained behavioural signatures unique to each application.

1. Packets: this is the number of packets per flow in forward and back ward direction. These statistical features (min, max, std, skewness) are calculated for each direction separately.
2. Bytes: this is the total bytes in a flow. Statistical features are calculated per direction.
3. Inter-Arrival Time (IAT): IAT measures the time gaps between consecutive packets in a flow. Includes statistical features: *min, max, mean, std, skewness,* for each direction.

 doi:10.20944/preprints202605.2006.v1

4. Flow Duration: The total duration from the first to the last packet in the flow, generally treated as a single feature. It isn't split by directions. *Example:* flow_duration.
5. Active and Idle Times: These measure periods of activity (packets being transmitted) and inactivity within the flow, with statistical features (min, max, mean, std, skewness). *Example:* active_forward_max.
6. Flags: Counts of TCP (Transmission Control Protocol) flags (e.g., PSH, URG, FIN, ACK, SYN, RST) observed in the flow, typically in the forward or backward direction depending on the implementation. *Example:* forward_psh_flags.
7. Throughput or Rates: These are measures of the traffic rate, expressed in packets/sec or bytes/sec, computed separately for forward and backward directions. *Example:* bytes_per_second_backward.

The following statistics are calculated by MIRAGE-2019 for most feature categories it includes min which is the minimum value, max which is maximum value, mean is the average, std the standard deviation, skewness – asymmetry in the feature distribution and sum which is sometimes included for total aggregation.

Thus, combinations of direction (forward/backward) and statistic type (min, mean, std, etc.) might result in numerous derived features (e.g., packet_size_forward_mean, packet_size_backward_std) from a single logical feature, such as packet size.

Because they capture the unique behavioural fingerprints of every application, these statistical flow-level attributes are essential for application-level traffic classification. The Random Forest classifier can efficiently learn to differentiate between different application types and deduce their QoS attributes, including latency sensitivity and bandwidth requirements, by analysing these distributions.

3.3. Ethical Considerations Related to the Dataset

Since participants were alerted of the goals and potential public release of traces. The capture procedure and dataset collecting do not raise any ethical issues. Logged source IP addresses are part of the private IPv4 space, and mobile devices were supplied and utilized in the ARCLAB. From the capture stage until the construction of the dataset and throughout the classification procedure, no participant personal information was used (Aceto et al., 2019). To protect privacy, this project only uses publicly accessible datasets. In accordance with ethical guidelines for network data study, no personally identifiable information is gathered or processed.

3.4. MIRAGE -2019 Dataset Justification and 5G Limitations

There are currently few publicly accessible datasets that depict actual 5G network traffic with identified application classes. Instead of emphasizing the flow-level statistical features necessary for supervised learning, most current datasets are either private, simulation-based, or you must pay to access them.

Therefore, the MIRAGE-2019 dataset, which was gathered prior to 5G, is used in this analysis. One of the few publicly available, labelled mobile application traffic datasets that is appropriate for Supervised Machine Learning is still this one. The rich statistical properties of MIRAGE-2019 can be abstracted to simulate 5G flow behaviour and are in line with the features of 5G networks.

This restriction draws attention to a crucial area of research: the requirement for sizable, publicly accessible, labelled 5G datasets for Machine Learning-based QoS optimization.

A 5G traffic dataset was first taken into consideration, but it had insufficient usable flow samples following feature extraction to enable efficient model training. Because of its greater scale, diversity, and labelled flow-level features that are appropriate for supervised classification, the MIRAGE-2019 dataset was chosen.

 doi:10.20944/preprints202605.2006.v1

3.5. Implementation Environment

The network traffic classification model was developed in Python using the Visual Studio Code environment. It relies on many open-source computing and machine learning libraries and packages, summarized in Table 2 ("Joblib: running Python functions as pipeline jobs — joblib 1.5.2 documentation,").

Table 2. Python Dependencies.

No	Package	*Purpose*
1	Pandas	Data manipulation and analysis
2	Numpy	Numerical computing
3	Scikit-learn	Machine learning algorithms and evaluation metrics
4	matplotlib	Visualization and plotting
5	seaborn	Statistical data visualization
6	Imbalanced-learn	Oversampling and data balancing techniques
7	joblib	Model persistence and serialization
8	scipy	Scientific and mathematical operations

3.6. Flow Aggregation

The MIRAGE-2019 dataset consists of flow-level JSON files, each describing statistical characteristics of bidirectional network communications produced by a particular mobile application. Although the data is derived from raw packet captures, the MIRAGE dataset already provides flow-level statistical features. This simplifies the pre-processing stage by removing the need for flow computation.

In the proposal it was mentioned that flow computation might be necessary if the dataset contains raw PCAP files. Where it will require computing and extracting the flow level statistical features, but the MIRAGE-2019 dataset contains JSON files with the flow-level statistical features.

Each flow has the key "192.168.20.101,51221,216.58.205.42,443,6" which represents,192.168.20.101 is the "Source IP", 51221 is the "Source Port", 216.58.205.42 is the "Destination IP" (Google's domain here, Twitter), 443 is the "Destination Port" (HTTPS) and 6 is the "Protocol Number", in this case 6 is TCP.

The string "192.168.20.101,51221,216.58.205.42,443,6" functions as a flow ID or 5-tuple in the MIRAGE dataset. It uniquely identifies a bidirectional flow by specifying the source IP and port, destination IP and port, and the protocol number. This 5-tuple serves as a key to associate all related data (per-packet info, flow features, metadata) within the dataset.

Every JSON file has a set of flows, each of which includes a variety of network metrics arranged in a layered hierarchical structure, including packet lengths, inter-arrival periods, and TCP window data.

But models like the Random Forest classifier and Machine Learning frameworks like Scikit-learn need tabular, structured numerical input. The data had to be converted from JSON to CSV format

 doi:10.20944/preprints202605.2006.v1

because the MIRAGE JSON files are hierarchical and unstructured for direct ML input. In this structure, a single network flow is represented by each row, a numerical feature is represented by each column, and the originating application is specified by a single label column.

To prepare the dataset for supervised learning, a python script was developed. This script performs the flow aggregation process using two primary methods: *process_file()* and *aggregate()*.

The *aggregate()* function iterates through all JSON files, calling *process_file()* for each file, and then concatenate the individual flow data frames into a single dataset.

Each JSON file is read by the *process_file()* function, which then extracts relevant flow-level statistical features including TCP window attribute, packet length statistics, and inter-arrival time metrics and arranges them into a flat data frame. The function gathers and flattens the hierarchical JSON structure's flow-level statistical features into a format that is suited for Machine Learning.

Flow-level statistical features get saved in the dataframe as the following features, feat_cat which is the category of feature (e.g., packet_size, inter-arrival time), direction which tells flow direction (either forward or backward), stat_name which is statistical measure (mean, std, min, max), val which is the statistic's numerical value and col_name which makes each extracted statistic a column in the dataframe by creating a unique column name such as packet_size_forward_mean.

By merging the JSON data from the Google Nexus and Xiaomi Mi5 smartphones into a single input directory, the model was able to learn from a greater variety of flow patterns across different hardware and software scenarios. The combined dataset produced more realistic and widely applicable results.

Finally, the exported CSV file Mirage_flows.csv represents the complete set of flow-based features and related applications labels. The Random Forest model is utilized for feature scaling, resampling, training, and classification in the supervised learning pipeline, which receives the Mirage_flow.csv file as input.

3.7. Data Pre-Processing

After combining all flow-level statistical variables into a single CSV file, Mirage_flow.csv, the data undergoes a pre-processing procedure to ensure compatibility with the Machine Learning model and enhance classification performance. This stage guarantees that the dataset is suitable for Supervised Machine Learning in terms of cleanliness, balance, and structure.

Each network flow in the Mirage_flow.csv dataset has a single row with hundreds of statistical variables that describe the characteristics of bidirectional communication. The column label identifies the original mobile application (like Facebook, YouTube, or Twitter).

During pre-processing, the dataset is divided into:

- Feature matric (X): containing the numerical flow-level features describing bidirectional flow characteristics such as packet sizes, inter-arrival times and TCP statistics.
- Label vector (y): contains the corresponding mobile application name for each flow (e.g., YouTube, Twitter, Spotify). The labels.

This division ensures that the classification algorithm is trained on the input features (X) to predict the correct output label (y).

3.8. Handling Missing and Contstant Values

A few flows in the MIRAGE dataset had missing or invalid entries due to incomplete network sessions. The pre-processing function replaces all missing values *(NaN)* with zero using *X.fillna(0).*

This approach avoids discarding incomplete flows, preserving dataset size while maintaining numerical compatibility for the classifier.

To ensure the model does not learn noise or redundant patterns missing values are replaced with zero (0) using the *fillna()* method.

Features with zero variance (i.e., identical values for all flows) are removed to improve efficiency and prevent overfitting.

 doi:10.20944/preprints202605.2006.v1

This step ensures that only statistically meaningful features contribute to model training. As such features provide no discriminative power for classification and only increase computational cost.

3.9. Dataset and Application

A few flows in the MIRAGE dataset had missing or invalid entries due to incomplete network sessions. The pre-processing function replaces all missing values *(NaN)* with zero using *X.fillna(0).*

Before transformation, the pre-processing function provides a summary of dataset characteristics including:

- The total number of features and flows: Dataset shape: 121955 flows, 102 features each
- The total number of unique applications: Number of apps: 20
- The number of flows per application (see in Table 3):

This information is crucial for understanding the dataset's structure and identify class imbalance, where certain applications may dominate in number of flows.

Table 3. Number of flows per app.

No	Identity	*Number of flows*
1	Waze	11865
2	Iliga	11048
3	accuweather	10682
4	duolingo	8583
5	subito	8275
6	Wish	6812
7	spotify	6615
8	foursquare	6549
9	youtube	6493
10	twitter	5767
11	comics	5662
12	facebook	5578
13	dropbox	5205
14	pinterest	4253
15	messenger	4133
16	tripadvisor	3676
17	slither	3189
18	viber	3118
19	trello	2380
20	Groupon	2072

3.10. Dataset and Application

A few flows in the MIRAGE dataset had missing or invalid entries due to incomplete network sessions. The pre-processing function replaces all missing values *(NaN)* with zero using *X.fillna(0).* The bar chart (see Figure 1) titled "Application Traffic flows" presents the distribution of traffic flows across different applications. Each bar represents the number of flows associated with a specific application. The colour coding corresponds to the priority classification under a QoS (Quality of Service) policy framework: High Priority (Dark cyan), Medium Priority (Pale orange), and Low Priority (light peach)

 doi:10.20944/preprints202605.2006.v1

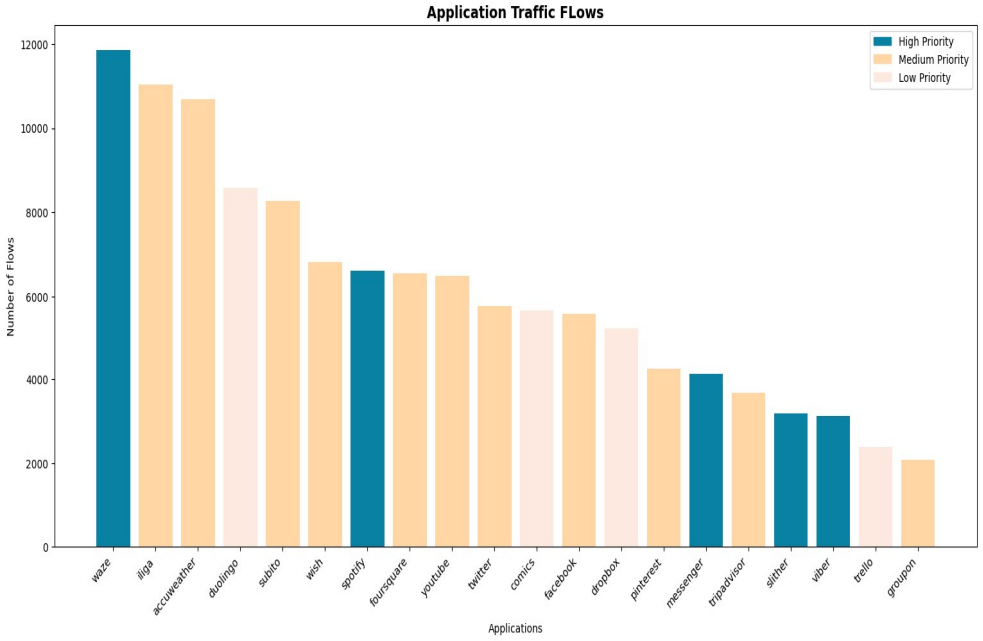

Figure 1. This Bar Chart illustrates the number of flows per app.

The number of flows is displayed on the y-axis, and apps are listed on the x-axis in descending order of traffic volume.

Key observations

1. Highest Traffic application:
   - Waze dominates with nearly 12000 flows, indicating significant usage of navigation services.
   - Accuweather, Duolingo and Iliga also show high traffic volumes (8000-11000 flows).
   - This suggests that navigation, weather, sport and language learning apps are among the most frequently used categories in the dataset.
2. High Priority Apps

   Waze, Spotify, and Viber are classified as High Priority, due to their real time needs:
   - Waze: requires low latency for real time navigation,
   - Spotify: needs stability for continuous listening,
   - Viber: demands high reliability for voice over IP.
3. Medium Priority Apps
   - Majority of applications (e.g., Accuweather, Subito, YouTube, Twitter, Facebook) fall into Medium Priority.
   - These applications need a reasonable amount of bandwidth but can withstand moderate latency.
   - Applications like YouTube and Facebook have significant flow counts, reinforcing their popularity in general usage.
4. Low Priority Apps
   - Duolingo, Dropbox, Trello, and Comics are mapped as Low Priority.
   - These apps generally involve background synchronization or non-real-time activity, making them less sensitive to latency.
   - Their flow counts are lower compared to high and medium priority apps, aligning with their reduced urgency.

From the bar chart it can be noticed that the traffic is unevenly distributed, with a few apps accounting for most flows (Waze, Accuweather, Motain, Duolingo). The bar chart assists in

 doi:10.20944/preprints202605.2006.v1

understanding the contribution of each application to the dataset and guides the choice of balancing techniques.

3.11. Model Training

Once the model was pre-processed and structured, the next stage involved training the supervised learning model. The *train_model()* function in this study performs three critical steps:

- Second bullet; Feature scaling ("StandardScaler,")
- Class balancing through oversampling (Mohammed and Ghani, 2025), and
- Model training using Random Forest (Wang et al., 2015).

Additional, Linear Discriminant Analysis (LDA) can optionally be applied for dimensionality reduction before model fitting (Mohammed and Ghani, 2025).

3.12. Feature Scaling

The dataset contains numerical features measured on different scales. Packet length has values like ~52,0, ~1200 (in bytes). Inter-arrival time might be in milliseconds (with values like ~0,05, ~5445,071). The model could become biased towards features with larger numeric ranges. If the model is given these raw numbers directly to the model, the features with larger numeric values dominate the learning process, while features with smaller values will get ignored even if those smaller values are just as important. To prevent features with larger numeric ranges from dominating the learning process, standardization was performed using the *StandardScaler* from *Scikit-learn* ("scikit-learn: machine learning in Python — scikit-learn 1.7.1 documentation,").

The scaler (*StandardScaler*) solves this by putting all features on the same scale. All input features are rescaled to a standardized distribution with a mean of zero and unit variance, ensuring that all features contribute equally during model training.

Mathematically, each feature x is transformed as given by Equation (1):

$$\mathrm{x} = \frac{(x-u)}{s} \quad (1)$$

where u is the mean of the training sample and s is the standard deviation of the training samples.

The fitted scaler object is serialized and stored (*feature_scaler.pkl*) to guarantee consistent normalization for future inferences and model deployment.

3.12.1. Why Save the Scaler?

The scaler ensures future classification of new network flows to be scaled in the same way as the training of the model. This prevents the risk of: Scaling new data differently, shifting the feature distribution, and getting wrong classification. During inference, when new data comes in, at deployment. The key is to use the same scaler object the model was trained with, so the new flows are on the same scale. This guarantees consistency between training and inference, preventing scaling mismatches.

3.13. Handling Class Imbalance Using Oversampling

The MIRAGE-2019 dataset contains network traffic from 20 mobile applications, each with varying traffic volumes. Applications such as Waze, Iliga and Accuweather have significantly higher number of flows compared to lightweight apps such as Viber or Trello.

This results in a class imbalance problem (see Figure 2), where the classifier is exposed to more examples from dominant classes. Training a model on such imbalanced data can cause the model to bias toward majority classes. This reduces the model's ability to generalize to minority classes and degrading performance metrics such as recall and F1-score for applications with fewer flows.

To address this challenge of imbalance, two oversampling techniques were implemented within the model:

 doi:10.20944/preprints202605.2006.v1

- RandomOverSampler: Randomly duplicates samples from minority classes until class frequencies are balanced. This is a simple yet effective approach that ensures balanced class distribution but may risk overfitting due to repeated samples.
- SMOTE (Synthetic Minority Over-sampling Technique): Unlike random duplication, SMOTE synthesizes new minority-class samples by interpolating between existing samples in the feature space. This improves the model's performance on minority classes and helps it generalize more effectively by producing more realistic synthetic instances (Mohammed and Ghani, 2025).

Third bullet.

Both oversampling techniques were designed as configurable toggles within the training model. The script ensures that only one method is applied at a time. If both are enabled,

RandomOverSampling is selected by default. Each oversampling approach increases the total number of samples to 237300 flows, maintaining the same 102 features. As a result, each application class contained 11865 flows ensuring balanced distribution across classes. In this study, RandomOversampling produced better results for the MIRAGE-2019 dataset compared to SMOTE, which is discussed later in the paper.

The over sampling techniques ensure that the classifier receives equal representation from all application categories. The balancing process significantly improved class representation and increased the diversity of training samples across all applications (Lemaître and Nogueira, 2017).

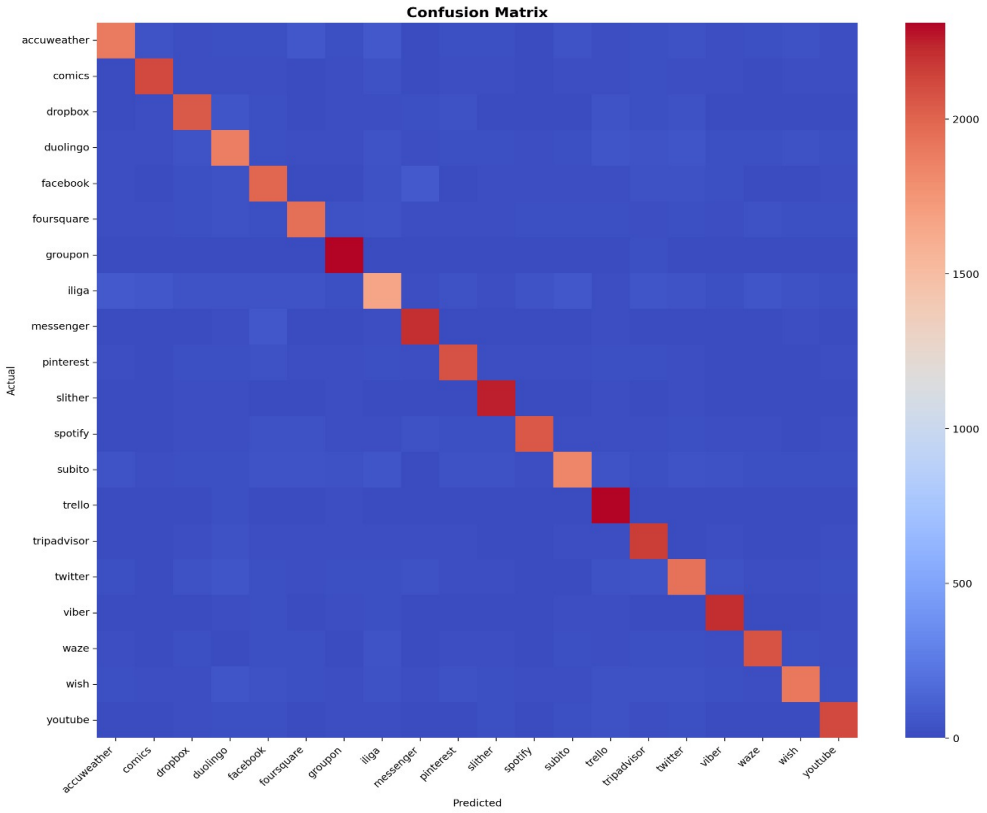

Figure 2. Confusion matrix.

3.14. Dimensionality Reduction (LDA)

From the dataset statistics, the MIRAGE-2019 dataset contains a total of 121955 flows and 102 flow-level statistical features each, including metrics such as packet length, maximum flow duration, and inter-arrival time variance (IAT var). Although having such a rich feature set provides comprehensive flow characterization, but it also introduces redundancy and features may overlap

 doi:10.20944/preprints202605.2006.v1

information. Training a model with redundant features increases computational cost, slows convergence, and may lead to overfitting, reducing the model's ability to generalize to new data.

To mitigate these issues, the model includes an optional Linear Discriminant Analysis (LDA) stage for dimensionality reduction (Mohammed and Ghani, 2025). To ensure that the new feature set retains the most discriminative information for differentiating between applications, LDA projects the data into a lower-dimensional space that maximizes class separability.

LDA was used in this study to simplify the dataset while preserving crucial discriminative features by reducing the feature space from 102 features each to 19 features each. This reduction leads to faster model training, improved generalization, and reduced memory overhead.

LDA's effect on classification performance was assessed by testing it with both SMOTE and RandomOverSampler oversampling techniques. According to Mohammed and Ghani (2025), utilizing LDA throughout the processing step aids in eliminating noise and irrelevant variability, concentrating learning on the most informative features and improving the accuracy and resilience of the model.

3.15. Model Training Using Random Forest Classifier

After feature pre-processing, scaling and optional dimensional reduction, the dataset was used to train a Random Forest classifier. Random Forest is an ensemble-based supervised learning algorithm. The algorithm constructs multiple decision trees during training and outputs their prediction for classification (Breiman, 2001). This method is highly effective for network traffic classification due to its robustness against overfitting, ability to represent non-linear feature interactions, and ability to interpret through feature importance metrics.

The dataset was split into 80% training and 20% testing subsets for supervised model learning and evaluation. This split was stratified, meaning that the relative class distribution was preserved between the training and testing sets. Stratification helps maintain representative samples of all application types, preventing bias toward dominant classes.

- First bullet; 80% training set:
  a. Without oversampling: 97564 flows with 102 features each.
  b. With oversampling: 189840 with 102 features each.
  c. With LDA applied: 189840 with 19 reduced features each.
- 20% testing set:
  a. With oversampling: 24391 flows with 102 features each.
  b. With oversampling: 47460 with 102 features each.
  c. With LDA applied: 47460 with 19 features each.

This split ensures that the model is trained on a sufficiently large and balanced dataset while the unseen test set provides an unbiased evaluation of its generalization performance. The LDA-based dimensionality reduction (from 102 to 19 features) further helps to minimize noise, reduce computational complexity, and enhance the discriminative capabilities of the classifier.

In this implementation, the Random Forest classifier was trained using the Scikit-learn library with the following optimized hyperparameters shown in Table 4.

Table 4. Random Forest configurations.

Parameter	***Value***	***Description***
n_estimators	300	Number of the forest's individual decision trees. A higher value balances accuracy and speed by reducing variance and improving stability while increasing computing time.

 doi:10.20944/preprints202605.2006.v1

max_depth	25	Limits tree depth to prevent overfitting while maintaining model expressiveness.
min_samples_split	5	To ensure balanced growth, a minimum number of samples is needed to split an internal node. Trees with higher values are less susceptible to slight variations in the data. 5 requires somewhat larger splits, which reduces overfitting.
min_samples_leaf	2	To avoid excessively fine partitions, a leaf node must have a minimum number of samples. Ensure leaves have enough samples to represent a group. 2 prevents very small leaves that may represent noise.
class_weight	"balanced"	To address any remaining imbalance, weights are automatically adjusted in an inverse proportion to class frequencies. Ensures minority classes are not ignored.
random_state	42	Ensure the model produces the same splits and results each time you run it.
n_jobs	-1	Utilizes all available CPU cores for faster parallelized training. -1 tells Scikit learn to use maximum parallelism.

Time-stamping functions were used to measure the entire training time, which gave information about the computational efficiency across configurations (e.g., with/without oversampling or LDA.

The trained model was serialized and stored as model.pkl using the Joblib package, allowing for future reuse without retraining. Similarly, the feature scaler and LDA transformer (if used) were also saved to disk to maintain pre-processing consistency during model deployment or evaluation.

Random Forest was selected because of its better generalization skills on flow-level features and track record of effectiveness in related traffic classification studies (e.g., Nguyen & Armitage, 2008; Aceto et al., 2019). The model's ensemble structure allows for interpretability through feature importance analysis and resistance against overfitting.

 doi:10.20944/preprints202605.2006.v1

4. Results and Discussion

4.1. Model Testing

The model's performance was evaluated using both traditional classification metrics and a QOS aware evaluation framework. The evaluations provided insights into how well the model not only classifiers mobile applications but also how its predictions align with network QOS requirements in 5G environments.

Evaluation metrics such as Accuracy, Precision,Recall, and F1-score, were used to assess the model's performance (Azab et al., 2024; Pell et al., 2023;) Nguyen and Armitage, 2008).

The model was evaluated on its results before and after applying techniques like cross validation, SMOTE, RandomOverSampling and LDA. Monitoring time and accuracy.

4.2. Model Performance on RandomOverSampling

The Random Forest classifier achieved strong performance after applying strong performance after applying RandomOverSampling to handle the imbalance in the MIRAGE-2019 dataset. After resampling, each application class contained an equal number of flows of 11865. This led to a balanced dataset of 237300 total flows across 20 applications. The model was then trained on 80% of the data and tested on the remaining 20% (47460 flows).

The model achieved an overall of 89.76% with both the macro and weighted F1-scores equal to 89.71%. These metrics indicate a highly balanced performance across all classes. It demonstrates that oversampling effectively mitigated the bias toward majority classes. The closeness of the macro and weighted F1-scores confirms that performance is consistent for both majority and minority applications.

Table 5 shows the overall model performance after using RandomOverSampling. Equations of evaluation metrics are given below (Pell et al., 2023, Nguyen and Armitage, 2008):

$$\text{Accuracy} = (\text{True positives} + \text{True Negatives}) / (\text{True positives} + \text{True Negatives} + \text{False positive} + \text{False negative})$$

$$\text{Precision} = \text{True Positives} / (\text{True Positives} + \text{False Positives})$$

$$\text{Recall} = \text{True Positives} / (\text{True Positives} + \text{False Negatives})$$

$$\text{F1-Score} = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall})$$

Table 6 is the classification report across all 20 applications. Performance varied slightly depending on the type and nature of the traffic's application but remained consistently high across classes.

Table 5. Overall model performance after RandomOverSampling.

Metric	*Value*
Accuracy	0.8976
Macro F1-score	0.8971
Weighted F1-Score	0.8971

Table 6. Classification report for each application after RandomOverSampling.

Application	*Precision*	*Recall*	*F1-Score*	*Support (Flows)*
AccuWeather	0.9044	0.8491	0.8759	2,373
Comics	0.9256	0.9284	0.9270	2,373
Dropbox	0.8902	0.8883	0.8893	2,373
Duolingo	0.8344	0.8453	0.8399	2,373
Facebook	0.8679	0.8858	0.8767	2,373
Foursquare	0.8988	0.8837	0.8912	2,373
Groupon	0.8920	0.9781	0.9331	2,373
Iliga	0.8857	0.7611	0.8187	2,373
Messenger	0.9188	0.9444	0.9314	2,373

 doi:10.20944/preprints202605.2006.v1

Pinterest	0.9094	0.8925	0.9009	2,373
Slither	0.8919	0.9629	0.9260	2,373
Spotify	0.9542	0.9043	0.9286	2,373
Subito	0.9269	0.8390	0.8808	2,373
Trello	0.8423	0.9789	0.9055	2,373
TripAdvisor	0.8778	0.9351	0.9055	2,373
Twitter	0.8485	0.8588	0.8536	2,373
Viber	0.9040	0.9486	0.9258	2,373
Waze	0.9777	0.9229	0.9495	2,373
Wish	0.9251	0.8432	0.8823	2,373
YouTube	0.9108	0.9119	0.9113	2,373
Overall Accuracy	0.8981	0.8981	0.8981	-
Macro Average	0.8993	0.8981	0.8976	47,460
Weighted Average	0.8993	0.8981	0.8976	47,460

The balanced class distribution allowed the model to learn equally from all application types, improving its generalization and fairness across traffic categories. Key observations:

- The overall classification accuracy achieved by the Random Forest model is 89.81%, indicating a strong ability to differentiate between diverse application traffic flows.
- Applications such as Waze (F1 = 0.9495), Spotify (F1 = 0.9286), and Messenger (F1 = 0.9314) demonstrate excellent classification performance, reflecting the model's strength in recognizing distinct flow-level statistical patterns for high-traffic apps.
- Lower-performing applications, such as iLiga (F1 = 0.8187) and Duolingo (F1 = 0.8399), suggest some feature overlap or limited sample diversity, which can be improved with additional data augmentation or feature tuning.
- The macro and weighted averages (≈ 0.90 across all metrics) confirm consistent performance across both balanced and unbalanced flow distributions.

These results validate that the chosen Random Forest classifier with RandomOverSampler effectively captures discriminative features across multiple mobile applications, enabling accurate QoS-aware traffic classification in 5G environments. The use of RandomOverSampling was pivotal in achieving these balanced results. The model learned an unbiased representation of each application's statistical patterns by duplicating minority application flows until all the flows were represented uniformly.

4.3. Confusion Matrix

Figure 2 represents the confusion matrix heatmap for MIRAGE-2019 dataset.

- First Y-axis is the actual application labels from the dataset
- X-axis is the predicted application labels produced by the model.
- Diagonal cells (Top left to bottom right) are the correct classifications, where the actual application match the predicted app.
- Colour intensity indicates the number of flows. Red shades indicate higher counts, while blue indicated lower counts.

Key observations:

1. First item; Strong Diagonal pattern:

 doi:10.20944/preprints202605.2006.v1

The bright red diagonal shows that most flows are correctly classified. This suggests that the model is learning distinguishing flows level statistical features for each application with high accuracy.

2. High traffic application:

Applications such as Waze show very high diagonal values. There are approximately +2000 flows correctly classified. This indicates strong model performance on applications with larger training samples.

3. Misclassification:

The off-diagonal regions are largely dark blue, meaning few flows are being classified across applications. Some small overlaps which are the lighter blue near the diagonal show minor confusion between apps with similar traffic patterns (e.g., social media apps like Facebook and Twitter.)

QOS Performance implications

- High precision and recall: Application with more training flows such as Waze have high classification accuracy. This ensure that QoS enforcement for these heavy-traffic apps will be reliable.
- Potential overlap of similar services: applications with similar traffic characteristics (e.g., social media platforms) may show occasional misclassification, which could slightly affect policy enforcement.
- Generalization Strength: Since off-diagonal errors are low, the model demonstrates good generalization to unseen flows in the test set.

The confusion matrix demonstrates that the classification pipeline is performing very well. It displays:

- First bullet; High accuracy across most applications
- Minimal misclassification across categories
- Strong and reliable in QoS enforcement, since key apps such as Waze and all social media applications are accurately recognized.

The confusion matrix validates that the chosen feature set and Random Forest Model with RandomOverSampling are suitable for network traffic classification in QoS optimization scenarios.

4.4. Feature Important Analysis

The Random Forest model identified the most discriminative features for traffic classification. Feature importance analysis reveals which flow-level statistical characteristics are most valuable for distinguishing between different application types. Also providing insights for network monitoring and management (Nguyen and Armitage, 2008). After training the model, the Random Forest classifier identified the following 10 important features contributing to classification accuracy (Moore and Zuev, 2005).

To understand why these features matter for QoS traffic classification, Table 7 summarizes the top 10 features along with their traffic behaviour implications and corresponding QoS relevance. The top-ranked features indicate that packet length and inter arrival time (IAT) statistics play the most dominant role in differentiating application behaviours. This aligns with findings from Aceto et al. (2019), which emphasize that packet size variability and timing patterns are key discriminants between real-time and background mobile applications.

Table 7. Feature Importance.

Feature Name	Traffic Behaviour Interpretation	QoS implication
packet_length_ upstream_flow _max	Maximum packet size in uplink traffic (e.g., video calls or uploads).	Large uplink packets require low latency and high throughput

 doi:10.20944/preprints202605.2006.v1

		uplink allocation
packet_length_ upstream_flow _var	Variance of uplink packet sizes; emphasizes extreme differences.	Identifies apps switching between control and payload packets.
packet_length_ upstream_flow _std	Variation in uplink packet sizes; distinguishes browsing vs. streaming	High variability signals adaptive bitrate streaming; needs burst capacity.
packet_length_ upstream_flow _90_percentile	Typical large packet size in uplink	Indicates consistent media chunking; requires smooth uplink QoS
packet_length_ downstream_fl ow_max	Largest packet size in downlink.	Streaming apps (YouTube); high downlink throughput demand
packet_length_ biflow_max	Maximum packet size across both directions	Captures overall session intensity (video vs. browsing).
iat_upstream_fl ow_max	Longest time gap between uplink packets	Low gaps = real-time apps (e.g., calls), High gaps = background sync apps.
iat_biflow_max	Longest time gap across both directions	Distinguishes continuous vs. idle traffic; adaptive scheduling needed
packet_length_ upstream_flow _mean	Average downlink packet size.	Distinguishes text/chat apps from video or file transfer.
iat_downstrea m_flow_max	Longest delay between downlink packets.	Identifies idle/background traffic vs. continuous streaming.

The dominance of packet length and IAT-based features demonstrates that traffic volume and timing are central to app identification. Applications like YouTube and Spotify show large, regular downlink packets, while Viber and Messenger exhibit smaller, more frequent bidirectional packets.

In 5G contexts, such feature importance analysis can guide QoS-aware scheduling, for example, allocating low-latency, high-throughput channels to real-time applications; prioritizing burst capacity for adaptive streaming traffic; deprioritizing background sync flows during congestion.

The Random Forest model's feature importance thus serves not only for accurate classification but also for QoS policy enforcement. This bridges the gap between traffic analysis and network resource optimization.

 doi:10.20944/preprints202605.2006.v1

4.5. Model Performance Under Different Configurations

The model was trained and evaluated using flow-level statistical features extracted from the MIRAGE-2019 dataset. The evaluation compared five different pre-processing configurations to assess the impact of oversampling and dimensionality reduction techniques on the Random Forest classifier's performance. Each experiment used an 80:20 train-test split with stratification to preserve class balance.

Five scenarios were considered:

1. Baseline (no oversampling, no dimensionality reduction)
2. SMOTE (Synthetic Minority Oversampling Technique)
3. SMOTE and LDA (combining oversampling with dimensionality reduction)
4. RandomOverSampler (simple oversampling by duplication)
5. RandomOverSampler and LDA

The key metrics recorded were Accuracy, Macro F1-Score and weighted F1-Score, alongside the total running time. These metrics provide a balanced evaluation of overall correctness, per-class fairness, and computational cost.

The results are summarized in Table 8.

Table 8. Performance Comparison.

Configuration	*Accuracy*	*Macro F1-Score*	*Weighted F1-Score*	*Training Time*	*Remarks*
No Oversampling, No LDA	0.7197	0.6706	0.7225	79.10 seconds ~1 minute and 19 seconds	Baseline model; suffers from class imbalance.
SMOTE	0.8206	0.8206	0.8206	168.30 seconds ~2 minutes and 48. seconds	Significant accuracy gains but high computation time due to synthetic data generation.
SMOTE and LDA	0.7564	0.7561	0.7561	145.77 seconds ~2 minutes and 25 seconds	LDA reduced features but slightly lowered accuracy, improved stability.
RandomOverSampler	0.8976	0.8971	0.8971	146.42 seconds ~2 minutes and 26 seconds	Achieved best accuracy and balance between recall and precision.
RandomOverSampler and LDA	0.8643	0.8631	0.8631	106.29 seconds ~1 minute and 46 seconds	Slight drop in accuracy, but best trade-off between accuracy and computational cost.

 doi:10.20944/preprints202605.2006.v1

The results demonstrate that oversampling substantially improves the model's ability to classify minority classes in the imbalanced MIRAGE dataset.

1. First item; No oversampling, no dimension reduction: Without any oversampling or dimensionality reduction, the model achieved only 71.97% accuracy and a macro F1-score of 0.67. This indicates a strong bias toward majority classes. Minority-class applications were underrepresented, this led to reduced recall and fairness.
2. SMOTE: Applying SMOTE increased both accuracy and macro F1-score to 82.06%. This demonstrated that class balancing improved fairness across all applications. However, SMOTE's synthetic feature generation increased model training time (168 seconds = 2 minutes 48 seconds), and some minority-class noise may have been introduced, explaining why the RandomOverSampler later performed even better.
3. SMOTE and LDA Dimensionality Reduction: When combined with SMOTE, LDA reduced the feature set from 102 to 19 components. While this improved efficiency, it caused a drop in accuracy (75.64%), likely because discriminant analysis compressed too much variance, losing fine-grained distinctions among similar app flows. However, in combination with RandomOverSampler, LDA still achieved 86.43% accuracy while reducing training time to 106 seconds = approximately 2 minutes 46 seconds, indicating a good trade-off for resource-constrained environments.
4. RandomOverSampler Performance: RandomOverSampler achieved the best overall performance (Accuracy: 0.8976, Macro F1: 0.8971) with balanced per-class results. This method replicated minority-class samples rather than synthesizing new ones, resulting in more stable decision boundaries and reduced overfitting compared to SMOTE. The slightly higher training time (146 seconds = 2 minutes, 26 seconds) is justified by the significant improvement in predictive consistency.
5. RandomOverSampler and LDA: RandomOverSampler, LDA achieved an accuracy of 86.43%, a macro F1-score of 0.8631, and a training time of 106.29 seconds which is approximately 1 minute and 46 seconds. This represents a slight drop in accuracy compared to using RandomOverSampler alone (89.76%), but it offers the best trade-off between accuracy and computational cost. The combination effectively reduced model complexity and training time while retaining strong classification performance, making it suitable for deployment in resource-constrained or nearreal-time environments.

Training times varied between 1 and 2 minute and 40 seconds, depending on pre-processing complexity.

The LDA-based configurations consistently reduced computational cost, with the RandomOverSampler and LDA setup completing training in 106.29 seconds (~1 minute and 46 seconds), approximately 27% faster than RandomOverSampler alone, while maintaining 86.43% accuracy.

This demonstrates that dimensionality reduction via LDA can substantially improve computational efficiency without major sacrifices in model accuracy, providing a balanced approach for edge-based QoS classification systems.

4.6. Cross-Validation

During the experimentation phase, attempts to perform cross-validation with SMOTE and Random Forest led to memory allocation failures. Cross validation is a model evaluation technique that splits the dataset into k smaller subsets. The model is trained on k-1 folds and tested on the remaining one. This process repeats k times, each time with a different test fold. The final performance is the average across all folds.

The technique provides a more reliable estimate of model generalization than a single train/test split. It helps detect overfitting by assessing performance on multiple unseen subsets. In this study, the combination of cross-validation, SMOTE and Random Forest caused a memory allocation failure.

 doi:10.20944/preprints202605.2006.v1

This happened because; SMOTE oversampling doubles the dataset size by generating synthetic samples for minority classes. Cross validation further multiplies data processing. For example, if you use 5-fold cross-validation, the training process repeats 5 times on large, oversampled data. Random Forest itself is memory intensive because it builds hundreds of trees, each storing feature splits, thresholds, and intermediate results. Together, these three factors led to memory exhaustion, especially when operating on large flow-level datasets like MIRAGE-2019 which has +120 000 flows and +100 features.

Future work will address this challenge by exploring

- Mini-batch or incremental training,
- Memory-efficient data generators,
- GPU-accelerated learning environments,
- Dimensionality reduction prior to resampling

4.7. Comparative Evaluation with Related Studies

To conceptualize the performance of the system, results were compared with recent related works that implemented Supervised Machine Learning for network traffic classification and QoS optimization. The comparative evaluations are summarized in Table 9.

Table 9. Comparative evaluations with two other studies.

Study	Algorithm	Dataset	Accuracy (%)	Remarks
Mohammed & Ghani (2025)	RF + SMOTE	5G Dataset	90.1	QoS focus
Ganesan et al. (2021)	RF	IoT Dataset	87.0	SDN-based
This study	RF + ROS	MIRAGE2019	89.76	QoS Policy Mapping

Mohammed and Ghani (2025) developed a Random Forest classifier using statistical flow features to improve QoS optimization. Their system achieved an accuracy of 90.1% with the integration of SMOTE improving minority class representation. Our model achieved a comparable accuracy of 89.76% using RandomOverSampler and Random Forest, with slightly reduced computational complexity. Both studies demonstrate that ensemble-based methods offer stability and interpretability for QoS-related applications.

Ganesan et al. (2021) applied various supervised ML models, including Random Forest and SVM, within an SDN-enabled FiWi-IoT environment. Their Random Forest achieved approximately 87% accuracy with moderate training cost. In contrast, this study's model achieved a higher accuracy of 89.76% and macro F1-score of 89.71%, validating the adaptability of Random Forest when trained on diverse mobile application flows from MIRAGE-2019.

All things considered, this comparison analysis demonstrates that the Random Forest framework performs closely to comparable research papers and additionally provides a QoS policy mapping component, which goes beyond classification accuracy to provide useful network management choices.

4.8. QOS Policy Analysis

Figure 3 depicts traffic distribution according to QoS priority classification. This shows how many flows belong to high, medium and low priority application. It helps identify which portion of the network traffic requires more attention or resource allocation. And it supports QoS policy design where the network can allocate more scheduling mechanisms to maintain performance.

 doi:10.20944/preprints202605.2006.v1

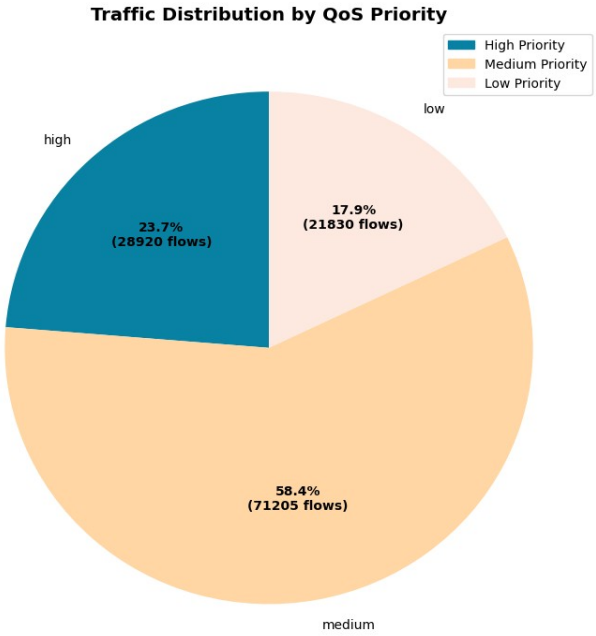

Figure 3. Traffic Distribution by QOS Priority.

The medium-priority category dominates, accounting for approximately 58.4% (71205 flows), which aligns with the large proportion of medium-latency, medium-bandwidth applications observed previously. High-priority traffic comprises 23.7% (28920 flows) of the total, representing latency-sensitive services such as navigation, video conferencing, and interactive applications. These require guaranteed bandwidth and minimal packet delay variation. Low-priority flows account for 17.9% (21830 flows) and primarily correspond to background processes and delay-tolerant transfers.

This priority distribution reflects a realistic network environment where most applications require balanced performance, while a smaller portion demands strict QoS guarantees. The Random Forest-based classification model supports automated QoS policy assignment, ensuring that resource allocation aligns dynamically with the observed priority levels to optimize end-user experience and network efficiency.

Figure 4 presents the distribution of network traffic flows according to their latency sensitivity levels, which directly influence Quality of Service (QoS) requirements in 5G networks. The pie chart (in Figure 9) shows how much of the traffic is sensitive to delay. This is useful for QoS differentiation where the network can minimize delay for real-time flows.

Medium Sensitivity traffic constitutes the majority, accounting for 60.3% (73,567 flows). This category typically represents applications such as video streaming, social media, and general web browsing, which tolerate moderate delays without noticeable user degradation. These applications require consistent throughput but are less dependent on extremely low latency.

Low Sensitivity traffic makes up 21.4% (26,082 flows). These flows correspond to non-real-time services such as file downloads, background synchronization, or cloud storage. Since they can withstand significant delays, they are assigned lower priority in QoS scheduling. They can tolerate longer delays.

High Sensitivity traffic represents 15.7% (19,116 flows). This includes time-sensitive services such as VoIP, messaging, and interactive online gaming, where latency directly impacts user experience. These flows require prioritized handling to maintain real-time responsiveness. Very High Sensitivity traffic accounts for only 2.6% (3,189 flows). This small yet critical subset includes ultralow latency applications such as real-time gaming (e.g., *Slither.io*) or live streaming. These flows demand

 doi:10.20944/preprints202605.2006.v1

strict delay constraints and are key targets for 5G QoS optimization. They demand low end-to-end delay.

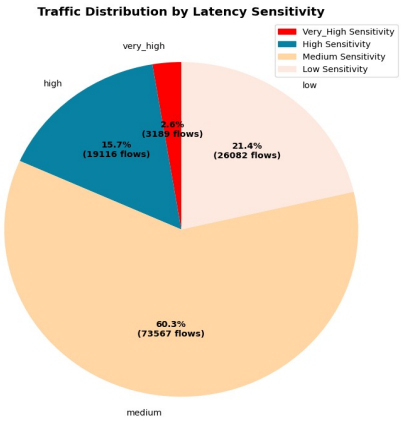

Figure 4. Traffic Distribution by Latency Sensitivity.

The distribution of application traffic based on bandwidth requirements is shown in Figure 5. This shows the proportion of traffic requiring high, medium and low or variable bandwidth. Medium Bandwidth traffic dominates with 63.9% (77,930 flows). This class includes most user applications such as social media, video streaming, and navigation services. These applications require consistent throughput for stable operation but do not need extremely high data rates. The large share of this category reflects the typical multimedia-heavy yet balanced usage patterns of mobile users.

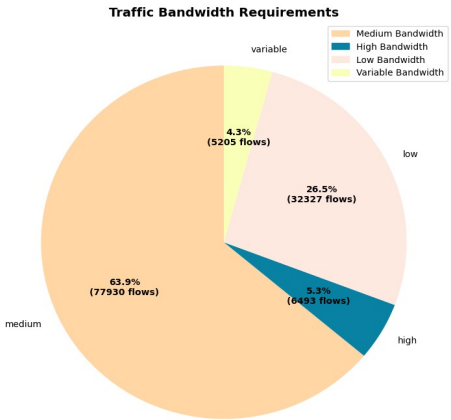

Figure 5. Traffic Distribution by Bandwidth Requirements.

Low Bandwidth traffic represents 26.5% (32,327 flows). This group includes lightweight services such as messaging apps, e-commerce, weather updates, and productivity tools. These applications consume minimal data and can operate under constrained network conditions, making them less demanding on network resources.

High Bandwidth traffic accounts for 5.3% (6,493 flows). These are data-intensive applications like video streaming and online gaming, where large data volumes must be transmitted rapidly to maintain quality. Despite being a smaller proportion, these flows are critical because they exert a disproportionately large impact on network load and require efficient QoS scheduling.

 doi:10.20944/preprints202605.2006.v1

Variable Bandwidth traffic comprises 4.3% (5,205 flows). These flows include applications such as cloud storage and file synchronization services, where bandwidth usage fluctuates dynamically depending on user activity and background transfers.

In the context of network traffic classification for QoS improvement, the pie charts help visualize how different types of application traffic (flows) are distributed according to their Quality-of-Service requirements. Together the pie charts summarize a networks demand profile. They help evaluate and design QoS aware traffic management policies and provide visual evidence of how classified applications map to specific QoS requirements.

4.8.1. QOS Policy Recommendations

The developed script not only performs traffic classification but also generates QoS policy recommendations for each identified application based on the predicted class labels and predefined QoS requirements. For every classified app, the model outputs a confidence score (F1-based) along with recommended priority level, bandwidth allocation, latency sensitivity, and jitter tolerance.

The script includes a QoS policy mapping where it maps each application to its QoS policy. This automated QoS policy recommendation feature demonstrates the practical applicability of the model to recommend QoS polices beyond mere classification. It provides a decision-support layer that can be integrated into network management systems to dynamically allocate resources, prioritize latency-critical traffic, and ensure optimal Quality of Service (QoS) across 5G applications. The confidence scores (ranging between 0.819 and 0.949) further validate the reliability and consistency of the classifier in supporting QoS-aware decision-making.

6. Practical Implications

The findings of the study extend beyond experimental evaluation. It demonstrates meaningful practical advantages for network operators, service providers, and system designers in 5G and beyond-5G environments.

6.1. Network Management Benefits

The traffic classification and QoS policy-mapping framework delivers several networks managements benefits that improve performance, efficiency, and maintainability. By automatically converting classification outputs into QoS rules, the system reduces manual configuration effort and supports the development of self-optimizing networks. Because it relies only on flow-level statistical features, it preserves user privacy and remains compliant with regulations such as the GDPR (General Data Protection Regulation) (Wolford, 2018), enabling accurate traffic recognition even when payloads are encrypted. Its use of Random Forest ensures scalability through efficient parallel processing, making it capable of analysing thousands of flows in real time across 5G slices, IoT traffic, and MEC nodes. The framework also enables adaptive resource allocation by mapping each application to its relevant QoS needs, such as latency sensitivity, bandwidth requirements, and jitter tolerance, allowing the network to prioritise delay-sensitive services like gaming or video conferencing while optimising bandwidth for background tasks. Collectively, these capabilities move the system toward intelligent, autonomous network management aligned with the broader goals of self-optimizing networks in modern 5G environments.

6.2. Deployment Considerations

Deploying such a system requires careful consideration of several operational factors. Model inference and feature extraction introduce additional CPU and memory overhead, meaning production environments may need GPU acceleration or offloading to edge nodes to maintain performance at scale. Because application traffic patterns evolve as software updates, protocols change, and user behaviour shifts, the model must be periodically retrained with fresh data to preserve accuracy over time. Effective deployment also depends on dynamic policy management:

 doi:10.20944/preprints202605.2006.v1

QoS rules should not remain static but instead integrate with SDN controllers or NFV managers so that policies can adjust automatically in response to real-time classification outputs. Maintaining continuous synchronisation between the classifier and the network control plane ensures that the system remains adaptive and responsive as traffic conditions shift across modern 5G environments.

6.3. Limitations and Challenges

The study faces several dataset-related limitations, primarily because MIRAGE-2019 captures traffic patterns from 2017–2019, before widespread 5G deployment and modern encrypted protocols such as TLS 1.3 and QUIC. As a result, the dataset does not fully represent contemporary 5G traffic behaviour. Public, labelled 5G datasets are still scarce. Most existing work relies on simulations or costly proprietary data, so MIRAGE-2019 was used as the closest open alternative. Its traffic was collected in a single academic environment in Italy, meaning the flows may not generalize to diverse networks, devices, or geographical regions. The study also uses only 20 of the 40+ available applications, simplifying training but limiting coverage of niche, evolving, or newly emerging apps.

On the technical side, the Random Forest model performs strongly for known applications but cannot generalize to unseen or newly updated apps without retraining. The current system works offline on pre-captured flows, which is suitable for experimentation but not yet for real-time 5G environments that require fast, low-latency inference at the network edge. Additionally, using more than 100 features and oversampling increases computational and memory overhead, which is manageable in research settings but too heavy for production unless the model is optimized or compressed.

7. Conclusions and Future Work

This study demonstrates the effectiveness of Machine Learning-based network traffic classification for Quality of Service (QoS) optimization in 5G network environments. The research presents a unified system combining data pre-processing, feature scaling, oversampling, dimensionality reduction, and Random Forest-based classification to map applications to QoS policies effectively.

Key contributions include:

Effective Classification System - achieved up to 89.76% accuracy and 0.8971 macro-F1 using RandomOverSampler with Random Forest, outperforming traditional models and baseline configurations;

Comprehensive QoS Framework - introduced a practical QoS policy mapping system that interprets classification outputs into actionable configurations for latency, bandwidth, and jitter management;

Privacy-Preserving Approach - demonstrated the ability to classify encrypted traffic using only flow-level features, eliminating the need for payload inspection while preserving user confidentiality;

Integrated Implementation Pipeline - developed a complete, modular system encompassing feature extraction, scaling, balancing, model training, and QoS visualization, supporting reproducibility and deployment readiness

Compared with prior studies - such as Ganesan et al. (2021), who reported approximately 87% accuracy using Random Forest in the SDN-based environment, and Mohammed & Ghani (2025), who achieved 90% using Random Forest - this study's results align closely with state-of-the-art outcomes while maintaining computational efficiency. This research validates that Supervised Machine Learning, particularly Random Forest, can effectively classify mobile application traffic based solely on statistical flow attributes. The results establish a foundation for integrating classification into autonomous QoS control systems for 5G networks.

 doi:10.20944/preprints202605.2006.v1

7.1. Future Work

Future work focuses on improving both the technical capability and practical deployment of the system. Technically, the model should be extended to support real-time traffic classification and evaluated using more advanced machine learning methods such as Gradient Boosting, XGBoost, CNNs, and LSTMs to improve generalization. System enhancements include introducing adaptive QoS control using reinforcement learning which should allow QoS policies to self-adjust based on feedback, incorporating richer QoS metrics like jitter variance, throughput and packet loss, enabling context-aware decision-making based on live network conditions, and supporting automatic model updates through online learning. Evaluation should be expanded by testing the model on multiple datasets, analysing long-term model drift which includes tracking how the accuracy and performance changes over time as applications and protocols evolve, identifying optimal retraining intervals. Evaluation will include assessing user Quality of Experience in real deployments and contributing to modern 5G-labelled datasets. Finally, standardization and industry alignment are key: integrating with 3GPP QoS standards, releasing the toolkit as open-source, and collaborating with mobile operators to support deployment in real 5G network environments.

7.2. Acknowledgement

The authors would like to express sincere gratitude to University of KwaZulu-Natal for financial support that has made work possible. We also thank the department of computer Science for availing resources that contributed to the development, refinement, and completion of this work.

References

1. Aceto G, Ciuonzo D, Montieri A, et al. (2019) MIRAGE: Mobile-app Traffic Capture and Groundtruth Creation. In: 2019 4th International Conference on Computing, Communications and Security (ICCCS). IEEE, Rome, Italy, pp 1–8 Available at: https://doi.org/10.1109/CCCS.2019.8888137
2. Azab A, Khasawneh M, Alrabaee S, et al. (2024) Network traffic classification: Techniques, datasets, and challenges. Digit Commun Netw 10:676–692. Available at: https://doi.org/10.1016/j.dcan.2022.09.009
3. Breiman L (2001) Random Forests. Mach Learn 45:5–32. Available at: https://doi.org/10.1023/A:1010933404324
4. Canever H, Wang X (2023) Network traffic classification using Unsupervised Learning: a comparative analysis of clustering algorithms
5. Choi Y-H, Kim D, Ko M, et al. (2023) ML-Based 5G Traffic Generation for Practical Simulations Using Open Datasets. IEEE Commun Mag 61:130–136. Available at: https://doi.org/10.1109/MCOM.001.2200679
6. Erman J, Arlitt M, Mahanti A (2006) Traffic classification using clustering algorithms. In: Proceedings of the 2006 SIGCOMM workshop on Mining network data. ACM, Pisa Italy, pp 281–286. Available at: https://doi.org/10.1145/1162678.1162679
7. Erman J, Mahanti A, Arlitt M, et al. (2007) Semi-supervised network traffic classification. In: Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. ACM, San Diego California USA, pp 369–370. Available at: https://doi.org/10.1145/1254882.1254934
8. Ganesan E, Hwang I-S, Liem AT, Ab-Rahman MS (2021) SDN-Enabled FiWi-IoT Smart Environment Network Traffic Classification Using Supervised ML Models. Photonics 8:201. Available at: https://doi.org/10.3390/photonics8060201
9. IANA Service Names and Port Numbers. Available at: https://www.iana.org/assignments/service-names-port-numbers/service-names-portnumbers.xhtml.
10. Karim S, He H, Laghari AA, Madiha H (2019) Quality of Service (QoS): Measurements of Video Streaming. Available at: https://doi.org/10.5281/zenodo.3987056
11. Labayen V, Magan a E, Morato D, Izal M (2020) Online classification of user activities using machine learning on network traffic. Comput Netw 181:107557. Available at: https://doi.org/10.1016/j.comnet.2020.107557

 doi:10.20944/preprints202605.2006.v1

12. Lemaître G, Nogueira F (2017) Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. J Mach Learn Res
13. Liao M-Y, Luo M-Y, Yang C-S, et al. (2012) Design and evaluation of deep packet inspection system: a case study. IET Netw 1:2–9. Available at: https://doi.org/10.1049/ietnet.2011.0048
14. Madhukar A, Williamson C (2006) A Longitudinal Study of P2P Traffic Classification. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation. IEEE, Monterey, CA, USA. Available at: https://doi.org/10.1109/mascots.2006.
15. Mahmood M, Khalil SA, Shah SJ, Ahmad M (2023) Network Traffic Classification Techniques and Comparative Evaluation of Machine Learning Models 8.
16. Maran on T, Armando J (2023) Traffic Classification of 5G Packet Traces
17. Mohammed AQ, Ghani RF (2025) Network traffic classification to improve quality of service (QoS). In: AIP Conference Proceedings. AIP Publishing, Baghdad, Iraq, p 020007. Available at: https://doi.org/10.1063/5.0264880
18. Moore AW, Zuev D (2005) Internet traffic classification using bayesian analysis techniques. In: Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. ACM, Banff Alberta Canada, pp 50–60. Available at: https://doi.org/10.1145/1064212.1064220
19. Nguyen TTT, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Surv Tutor 10:56–76. Available at: https://doi.org/10.1109/SURV.2008.080406
20. Pell R, Shojafar M, Kosmanos D, Moschoyiannis S (2023) Service Classification of Network Traffic in 5G Core Networks using Machine Learning. In: 2023 IEEE International Conference on Edge Computing and Communications (EDGE). IEEE, Chicago, IL, USA, pp 309–318. Available at: https://doi.org/10.1109/edge60047.2023.00053
21. Shafiq M, Yu X, Laghari AA, et al. (2016) Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms. In: 2016 2nd IEEE International Conference on Computer and Communications (ICCC). pp 2451–2455. Available at: https://doi.org/10.1109/CompComm.2016.7925139
22. Wang C, Xu T, Qin X (2015) Network Traffic Classification with Improved Random Forest. In: 2015 11th International Conference on Computational Intelligence and Security (CIS). pp 78– 81. Available at: https://doi.org/10.1109/CIS.2015.27
23. Wang X, Jiang J, Tang Y, et al. (2011) StriD^{2}FA: Scalable Regular Expression Matching for Deep Packet Inspection. In: 2011 IEEE International Conference on Communications (ICC). pp 1–5. Available at: https://doi.org/10.1109/icc.2011.5963289
24. W.Moore A, Papagiannaki K (2005) (PDF) Toward the Accurate Identification of Network Applications. Available at: https://www.researchgate.net/publication/220850310_Toward_the_Accurate_Identification_of_Network_Applications. Accessed 12 July 2025
25. Wolford B (2018) What is GDPR, the EU's new data protection law? In: GDPR.eu. Available at: https://gdpr.eu/what-is-gdpr/. Accessed 19 Oct 2025
26. Joblib: running Python functions as pipeline jobs — joblib 1.5.2 documentation. Available at: https://joblib.readthedocs.io/en/stable/. Accessed 22 Oct 2025
27. StandardScaler. In: Scikit-Learn. Available at:https://scikit-learn/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed 9 Oct 2025
28. scikit-learn: machine learning in Python — scikit-learn 1.7.1 documentation. Available at: https://scikit-learn.org/stable/index.html. Accessed 22 July 2025