IoT Traffic Multi-Classification Using Network and Statistical Features in a Smart Environment

Aroosa Hameed
Software and Information Technology Engineering
École de technologie supérieure
Montreal Canada
aroosa.hameed.1@ens.etsmtl.ca

Aris Leivadeas
Software and Information Technology Engineering
École de technologie supérieure
Montreal Canada
aris.leivadeas@etsmtl.ca

***Abstract*—As the number of Internet of Things (IoT) devices and applications increases, the capacity of the IoT access networks is considerably stressed. This can create significant performance bottlenecks in various layers of an end-to-end communication path, including the scheduling of the spectrum, the resource requirements for processing the IoT data at the Edge and/or Cloud, and the attainable delay for critical emergency scenarios. Thus, it is required to classify or predict the time varying traffic characteristics of the IoT devices. However, this classification remains at large an open challenge. Most of the existing solutions are based on machine learning techniques, which nonetheless present high computational cost while non considering the fine-grained flow characteristics. To this end, in this paper we design a two-stage classification framework that utilizes both the network and statistical features to characterize the IoT devices in the context of a smart city. We firstly perform the data cleaning and preprocessing of the data and then analyze the dataset to extract the network and statistical features set for different types of IoT devices. The evaluation results show that the proposed classification can achieve 99% accuracy as compared to other techniques with Mathews Correlation Coefficient of 0.96.**

***Index Terms*—Internet of Things, machine learning, network features, statistical features, traffic classification**

I. Introduction

Internet of Things (IoT) allows tens of billions devices to be connected and communicated over the Internet. Nonetheless, the rapid increase of IoT devices has also resulted in a colossal increase of the IoT data generated. Specifically, the total data has quadrupled in just five years from 145 ZB in 2015 to 600 ZB in 2020 [1]. IoT not only enables new applications, but at the same time it introduces new devices as well. These devices are usually of limited capabilities and cannot meet the time and performance constraints for mission critical scenarios. Thus, the computational burden is usually offloaded to the Edge and/or Cloud infrastructures. However, it is difficult to pre-allocate the resources during this computational offloading, since the total amount of requests/data may present a random generation behavior.

At the same time, when there is a large number of IoT devices generating data, the total communication delay may be affected due to the constrained nature of the IoT access networks. Thus, it is required to predict the time varying characteristics of the devices' traffic, in order to guarantee a specific level of Quality of Service (QoS). Therefore, by classifying IoT devices into different categories (i.e. hubs, cameras, air quality sensors etc.) it will allow to better predict the traffic generated and the services' requirements. Furthermore, it will better schedule the available spectrum and computational and communication resources.

Existing IoT traffic classification techniques either utilize aggregated traffic models [2]–[4], device fingerprinting technique [5]–[8], or the state-of-art machine learning techniques [9]–[12] . However, the above approaches present a number of limitations in terms of fine-grained device characterization, high cost data extraction processing, and scalability. To this end, in this paper, we propose a two-stage machine learning approach using two feature sets: i) a statistical feature set including packet inter-arrival time, burstiness rate, traffic flow rate, and ii) a fine-grained network feature set including IP addresses, IP protocol type, port numbers, MAC addresses, Time-to-Live (TTL) and packet size. The reason of using a two-stage classification approach with different features at each stage is to avoid a high dimensionality and overfitting of training data.

Specifically, the contributions of this study are threefold: (i) in order to classify the IoT traffic and devices, we introduce two traffic flow feature sets, namely, statistical feature set and network level feature set. This approach provides fine grained characterization of traffic flow with less computational complexity for IoT devices classification, which is a key building block of our method; (ii) we propose a relevance weighting that is assigned to each nominal (representing the qualitative data with numeric codes) features during the data preprocessing; (iii) a multistage machine learning based classification framework is presented with 99% accuracy in order to provide a scalable classification. To determine IoT device classification, we compute the classes for certain nominal and multivalued attributes at stage 0 using logistic regression and perform the final classification for numeric and single valued features at stage 1 using Gradient boosting algorithm along with the suitable adjustment of hyper-parameters i.e., learning rate and n-estimator.

The rest of the paper is structured as follows: In Section

Aroosa Hameed and Aris Leivadeas are partially supported by the CHIST-ERA-2018-DRUID-NET project

II a brief review of the related work is presented. Section III provides the network model and necessary preliminaries for comprehending the classification problem. Section IV elucidates the proposed classification model. Section V explains our proposed model algorithmic form and its asymptotic analysis. Section VI provides the performance evaluation results. Finally, Section VII concludes the paper.

II. Related Work

For an IoT device classification, a significant emphasis has been put into aggregated traffic models, fingerprinting, and machine learning solutions. Regarding aggregated traffic models, Laner et al. [2] proposed a Coupled Markov Modulated Poisson Processes (CMMPP) framework to capture the traffic behavior of a single machine-type communication along with the collective behavior of tens of thousands of M2M devices. In [3] a classification strategy is designed for a fleet management use case incorporating three different classes of M2M traffic states, namely periodic update, event-driven, and payload exchange. The authors in [4] proposed a traffic model that estimates M2M traffic volume generated in a wireless sensor network-enabled connected home. However, the above works do not consider the fine-grained characterization of IoT traffic and the complexity of such methods grows linearly with an increase in the number of M2M devices. Furthermore, common communication patterns were identified that can be attributed to any sensing device if using a specific use case (limitation 1).

Thus, there is a great interest of how to also classify and identify the type of devices used in each IoT application, an approach called fingerprinting . For example, "IoT Sentinel" [5] is a classification system that can recognize and identify the IoT devices immediately after they are connected to the network using a single attribute vector with 276 network features. The "IoT Sentinel" can be further improved by extracting an additional network features such as payload entropy, TCP payload length, and TCP window size [6]. Similarly, in [7], 300 network attributes are used from each TCP traffic session for a device classification using majority voting for every 20 consecutive sessions. In [8], patterns of encrypted network flows are used to reveal the existence of a specific device inside a home network. However, obtaining such a great number of features require specialized hardware accelerators, thus resulting in high computational cost, longer classification duration and limited scalability due to the need of a deep packet inspection functionality (limitation 2).

Some related works also employed machine learning in order to perform traffic and device classification. Lippmann et al. [9] compare the K-nearest neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT) and Multilayer Perceptron (MLP) algorithms, using the packet header information with the conclusion that KNN and DT resulting in better performance. Lopez-Martin et al. [10] classify the network traffic using the multi-class neural network, which is proven to be effective in complex data structures. The authors in [11] propose an individual binary classification model for each device class in order to eliminate the complexity issue of multi-class classification. Sivanathan et al. [12] utilize the statistical attributes, signaling patterns and cipher suites along with machine learning for IoT device classification. Nonetheless, these approaches are affected by the high data dimensionality, they are sensitive to the hyper-parameter tuning and they require a large number of training data. Moreover, the main constraint of the multi-class classification is scalability as the high number of classes makes the classifier complex and updating requires full retraining (limitation 3). While all the above works make an important contribution, each of them has several shortcomings. Our work aims at providing a classification framework to address the above cited limitations in such a way that in order: (i) to overcome the limitation 1, we incorporate the fine-grained features from the network layer, transport layer, and data link layer; (ii) to address the high computational costs of complex features, we propose the statistical feature set that can be calculated using a probability distribution; (iii) to address the limitation 3, we propose a two-stage classification framework. We also assign some relevance weighting to each nominal feature and distribute the features across these two stages along with appropriate hyperparameters.

III. Problem Definition and System Model

A. Multi-class Traffic Classification Problem

An IoT traffic classification problem can be formally defined as the task of estimating the class label c_i to the input vector x, where x belongs to a subset of a feature set F, $x \in X \subset F$ and $c_i \in C = \{c_1, c_2, ..., c_q\}$, where c_i represents the i^{th} possible class. This task is accomplished using a classification rule or function $f(x) : X \rightarrow C$ able to predict the label of new patterns. In a supervised setting, we are given a training set D of N points, from which $f(x)$ will be adjusted, $D = \{(x_i, c_i), i = 1, ..., N\}$.

B. Smart City Use Case

Assuming a smart city domain, we can find a set of classes $C = \{c_1, c_2, ..., c_q\}$, where each class represent a category of IoT devices according to the type of application they are used. For example, we may have 6 classes, where c_1 represents cameras, c_2 represents switches and triggers, c_3 represents healthcare devices, c_4 represents air quality sensors, c_5 represents light bulbs, and c_6 represents hub classes in the IoT smart city. In this case, the feature set F represents the distinctive properties of the traffic flow that we want to classify according to the class label c_i.

C. Feature Set

In this work, we consider two possible feature sets: i) a network flow feature set and ii) a statistical feature set.

Definition 1. *(Network Flow Feature Set): A network flow feature set, NF is a set consisting of features from network, transport, and data link layer as: $NF = \{f_1, f_2, ..., f_8\}$, where f_1 represents the source IP address, f_2 is the destination*

IP address, f_3 represents the IP protocol used, f_4 shows the source port number, f_5 represents the destination port number, f_6 is TTL information, f_7 is the source Ethernet address, and f_8 is the destination Ethernet address.

Definition 2. *(Statistical Feature Set): A statistical feature set, SF is a set consisting of features that are extracted from the network features using an appropriate probability distributions and is represented as: $SF = \{s_1, s_2, s_3, s_4\}$, where s_1 determines the inter-arrival time, s_2 is the average packet size, s_3 is the traffic rate, and s_4 represents the burstiness rate of the traffic.*

IV. Proposed Classification Method

A. Dataset Extraction

The publicly available data set [13] is used, collected over one month consisting of up to 21 IoT devices. Each data file consists of more than 500,000 records.

B. Preprocessing

After dataset filtering (i.e., removal of non-meaningful packets such as ping, etc.), we noted that some of the features such as "set of port numbers", "set of IP protocols" and "set of Ethernet addresses" are nominal and multivalued (having more than one value within a single data instance). As machine learning classifiers cannot deal with such data, we converted these features into a numerical form using a two-step procedure. Firstly, we perform the data cleaning by passing the nominal vectors to the Bag-of-Word (BoW) model [14]. Secondly, as the BoW assigns the same importance to each vector word, we have proposed the relevance weighting to assigned a prioritized importance to each word within each vector. These relevance weights against each feature vector are passed to the Stage 0 classifier and is given by (1):

$$Relevance\,Weight = wf_{w,v} \times vf_{w,v} \tag{1}$$

where $wf_{w,v}$ denotes the word frequency of words within a vector and $vf_{w,v}$ represents the total vector frequency. Herein, the vectors consist of the "port numbers vector", "IP protocols vector", and "Ethernet addresses vector". The word frequency $wf_{w,v}$ of a word w in vector v is defined as the number of times that w occurs in v and is given using (2):

$$wf_{w,v} = \frac{number\,of\,occurence\,of\,a\,word\,in\,a\,vector}{number\,of\,words\,in\,that\,vector} \tag{2}$$

Because frequent words are less informative than rare words, the vector frequency, $vf_{w,v}$ is given as (3).

$$vf_{w,v} = \log \frac{number\,of\,vectors}{number\,of\,vectors\,containing\,word\,w} \tag{3}$$

C. Feature Description

As presented above we considered both the statistical and network features. Network features are typical network attributes found in packet headers and thus self-explanatory. However, we provide a better insight into the statistical features as below:

1) Inter-arrival time: The inter-arrival time is the amount of time that elapses between a packet reception and the arrival of the one following it. After analyzing the data set, we extracted the time between the successive incoming traffic packets which follows an exponential distribution. The Probability Density Function (PDF) of an exponential inter-arrival time distribution is given using (4) with λ taken as 1 because at each time unit, one packet arrives. Furthermore as λ will get larger than 0, the event i.e., packet arrival tends to happen more quickly since λ is a rate parameter [15]. The X in (4) representing the time as a continuous variable.

$$f(X|\lambda) = \begin{cases} \lambda e^{-\lambda x} & x \geq 0 \\ 0 & x < 0 \end{cases} \tag{4}$$

2) Traffic flow rate: It represents the amount of an IoT traffic packets moving across a network at a given point of time.
3) Burstiness rate: It is the rate of consecutive packets whose inter- packet arrival time is shorter than the inter arrival time arriving before or after of these packets.

D. Proposed Classification Method

The proposed classification method, as shown in Fig. 1, consists of two stages. We follow this approach because of two reasons: firstly, in this way features are divided across two stages, thus high dimensional data is easily classified and secondly both classifiers at each stage use a different subset of features from the overall features (network + statistical), and hyperparameters are easily adjusted for each classifier thus avoiding overfitting. The BoW and feature relevance weights are fed into the stage 0 which is specially applied for the nominal and multi-valued features. The output of stage 0 classifier is the tentative classes, passed to the stage 1 classifier along with other numeric features providing the final output of the classification.

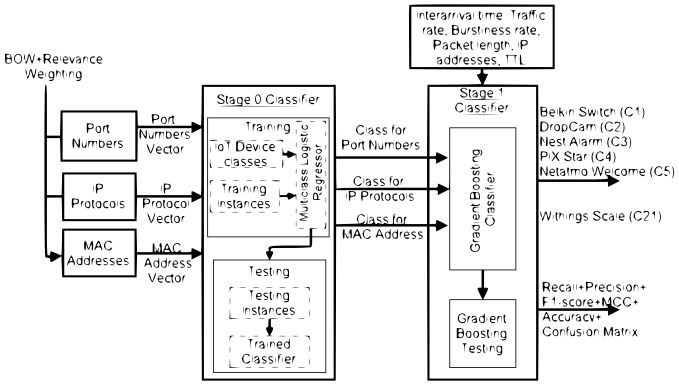

Fig. 1. Architecture of proposed classification method

1) Stage 0 classifier: The Logistic Regression acts as our Stage 0 classifier and takes as input the "set of port numbers", "set of IP protocols" and "set of Ethernet addresses" as BoW vectors along with the weights for the training. The reason

that we have selected this classifier is because it has been proven to perform well for very large data sets [16], as in the case of a smart city environment. Logistic regression investigates the association among the independent variables and the dependent variables of the problem. In our scenario, the port numbers, IP protocols and Ethernet addresses vectors are the independent variables and the device categories (e.g. hubs, cameras, etc.) are the dependent variables. The goal is to estimate the probability p for a combination of the independent variables using the following logit function:

$$logit(p) = ln\frac{p}{1-p} \tag{5}$$

where, ln is the natural logarithm and p denotes the probability of an independent variable. The antilog of (5) allows us to find the estimated regression equation given by (6):

$$logit(p) = ln\frac{p}{1-p} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 \Rightarrow$$
$$p = \frac{e^{\beta_0+\beta_1*x_1+\beta_2*x_2+\beta_3*x_3}}{1 + e^{\beta_0+\beta_1*x_1+\beta_2*x_2+\beta_3*x_3}} \tag{6}$$

where β_0 is an intercept, β_1, β_2, and β_3 are the regression coefficients, while x_1 denotes the first independent variable (i.e. port number), x_2 is the second independent variable (i.e. IP protocol), and x_3 is the third (i.e. Ethernet address). In order to calculate β_0, β_1, β_2, and β_3, we employed the Gradient Descent method [17]. The general form of (6) is given as:

$$p(y_i|x_1, x_2, x_3) = \frac{1}{1 + e^{-(\beta_0+\beta_1*x_1+\beta_2*x_2+\beta_3*x_3)}} \tag{7}$$

where y_i represents the dependent variable i.e., the i^{th} IoT device class, which we predict based on x_1, x_2, and x_3

After calculating the regression coefficients the testing component is come into effect, where the classifier uses the regression coefficients and computes the estimated regression for each testing instance using (7). Finally, the prediction effectuated by the Stage 0 classifier is based on the following equation:

$$\begin{cases} y_i = 1 & if\, p(y_i|x_1, x_2, x_3) \geq 0.5 \\ y_i = 0 & if\, p(y_i|x_1, x_2, x_3) < 0.5 \end{cases} \tag{8}$$

where $y_i = 1$ means that data with features x_1, x_2, and x_3 belong to i^{th} class and $y_i = 0$ means that they don't belong to the particular class.

2) Stage 1 classifier: In stage 1 classifier, a gradient boosting algorithm [17] takes all quantitative features such as packet inter-arrival time, traffic flow rates, burstiness rate, packet length, IP source address, IP destination address and TTL along with the output from the stage 0 classifier. The input of gradient boosting algorithm also includes the training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, and the number of iterations M. The differentiable loss function is the log of data likelihood given the prediction and is calculated as follows:

$$L = -observed * log(odds) + log(1 + e^{(log(odds)}) \tag{9}$$

The gradient boosting firstly finds the optimal initial prediction using (10) and then the pseudo-residuals for $m = 1$ to M using (11):

$$f_0(x) = arg\,min \sum_{i=1}^{M} L(y_i, \gamma_m) \tag{10}$$

$$r_{im} = -[\frac{\partial L(y_i, F(x_i)}{\partial F(x_i)}]_{F(x)=F_{m-1}x} \tag{11}$$

where γ_m is computed using (12) and the model is updated according to Eq. 13

$$\gamma_m = arg\,min \sum_{x_i \in \mathbb{R}_{i,j}} L(y_i, F_{m-1}(x_i) + \gamma) \tag{12}$$

where L is the loss function, y_i refers to the i^{th} observed value, $F_{m-1}(x_i)$ is the prediction function of x_i and is calculated using (13), $arg\,min$ signifies that we need to find the $log(odds)$ that minimize the summation, and $gamma$ denotes the $log(odds)$ value.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{13}$$

In the above equation, $h_m(x)$ represents the additional model (regression tree) for the prediction function. Lastly, using the output of $F_M(x)$, the classification of the testing data is done into the classes as discussed above.

V. Classification Algorithm

A. Algorithm Description

The proposed algorithm is depicted in Algorithm 1 and consists of three procedures, namely, $PREP$, $LOGREG$ and $GBOOST$. The $PREP$ procedure firstly generates the BoW representations using the function $generate\_BOW()$. Then, the relevant weights are calculated by employing the $word\_Freq()$ and $vector\_Freq()$ functions, which takes BoW as an input. Next the inter-arrival time, traffic rate and burstiness rates are calculated using the $interarrival()$ function, taking as an input the time t, $traffic\_rate()$ and $burstiness()$. In the $LOGREG$ procedure, the input labels x and output labels y are splits into training and testing data using the function $split()$. Then the $LogisticRegression()$ generates and fit the model using the $fit()$ function. The prediction is done using the $predict()$ function which contains the x_{tst} as testing dataset. The $GBOOST$ procedure generates the classification results. Specifically, the $GBoost()$ function firstly generates and fit the model with $fit()$ and then calls the $predict()$ function.

B. Computational Complexity

Proposition 1. *The computational complexity of $PREP$ procedure is $O(n)$*

Proof. The PREP procedure running time depends on the number of vectors, represented as n. The lines 1-3 takes a constant time as it splits the vectors into words, thus $O(1)$. The lines 4-5 and 7-9 are assignment statements and require time of $O(1)$. For the rel_{weight} statement (line 6) the complexity is

Algorithm 1 Classification Algorithm

PREP(t,win,BR,P,prt$_s$,prt$_d$,MAC$_s$,MAC$_d$)
// t is time; win is the TCP window size; BR is the burstiness rate; P is the type of IP protocol, prt$_s$ and prt$_d$ are the source and destination port numbers; MAC$_s$ and MAC$_d$ are the source and destination MAC addresses.
1. $BOW_1 \leftarrow generate\_BOW(prt_s, prt_d)$
2. $BOW_2 \leftarrow generate\_BOW(P)$
3. $BOW_3 \leftarrow generate\_BOW(MAC_s, MAC_d)$
4. $wf \leftarrow word\_Freq(BOW_1, BOW_2, BOW_3)$
5. $vf \leftarrow vector\_Freq(BOW_1, BOW_2, BOW_3)$
6. $rel_{weight} \leftarrow wf \times vf$
7. $Int \leftarrow interarrival(t)$
8. $T_r \leftarrow traffic_rate(win)$
9. $B_r \leftarrow burstiness(BR)$
Output:BOW$_1$,BOW$_2$,BOW$_3$,rel$_{weight}$,T$_r$,B$_r$
LOGREG(BOW$_1$,BOW$_2$,BOW$_3$,rel$_{weight}$,devices)
// devices represents the class labels
10. set $x \leftarrow dataset(BOW_1, BOW_2, OW_3, rel_{weight})$
11. set $y \leftarrow dataset(devices)$
12. set $x_{tr}, x_{tst}, y_{tr}, y_{tst} \leftarrow split(x, y, test_{size} \leftarrow 0.2)$
13. set $model \leftarrow LogisticRegression(max_{iter} \leftarrow 3000)$
14. set $fit \leftarrow model.fit(x_{tr}, y_{tr})$
15. set $y_{pred} \leftarrow model.predict(x_{tst})$
Output: y_{pred} ▷ Stage 0
GBOOST(y$_{pred}$,IP$_s$,IP$_d$,TTL,l, Int,T$_r$,B$_r$, devices)
// y_{pred} is the output of Stage 0 classifier; IP_s and IP_d are the source and destination IP addresses; TTL is the packet time to live; l is the packet length; Int is the interarrival time; T_r is the traffic flow rate; B_r is the burstiness rate set.
16. set $x \leftarrow dataset(y_{pred}, IP_s, IP_d, TTL, l, Int, T_r, B_r)$
17. set $y \leftarrow dataset(devices)$
18. set $x_{tr}, x_{tst}, y_{tr}, y_{tst} \leftarrow split(x, y, test_{size} \leftarrow 0.2)$
19. set $m \leftarrow GBoost(n_{estimators} \leftarrow 5000, LR \leftarrow 0.1)$
20. set $fit \leftarrow model.fit(x_{tr}, y_{tr})$
21. set $y_{pred} \leftarrow model.predict(x_{tst})$
Output: $y_{pred} : FS \leftarrow devices$ ▷ Stage 1

$O(1) * O(n) = O(n)$. Thus, the overall complexity of PREP procedure is linear i.e., $O(1) + O(1) + O(n) = O(n)$. □

Proposition 2. *The computational complexity of $LOGREG$ procedure is $O(n)$.*

Proof. The lines 10-12 are simple assignment statements (i.e., $O(1)$), while the training time (lines 13-14) of $LOGREG$ is $O(n * d)$ where n is the number of training examples and d is the number of data features used for the classifier training. However, the testing time taken by the line 15 is $O(n)$. Thus, the $LOGREG$ takes $O(1) + O(n * d) + O(n) = O(n)$ time and is suitable for low latency applications. □

Proposition 3. *The computational complexity of $GBOOST$ procedure is $O(ndk)$.*

Proof. In the $GBOOST$ procedure, lines 16-18 consist of simple assignments i.e., $O(1)$. The training lines 19-20 takes $O(ndk)$ time where, n is the number of training instances, d denotes the number of features for classification and k is the number of trees generated during training. However, the testing phase (line 21) requires $O(dk)$ time. Thus, the complexity is: $O(1) + O(ndk) + O(dk) = O(dk)$. □

The overall complexity of the proposed classification is: $O(n) + O(n) + O(ndk) = O(n)$. Thus, it is a linear time classification.

VI. Performance Evaluation

A. Experiment Setup

A total of 101922 labeled instances were collected from the traffic traces of 21 devices provided by [12]. The classification is being implemented in Python (version 3.8.2), while we split the dataset instances into two groups of 70% training instances and 30% testing instances. For the performance evaluation of the classification, we have considered the following metrics: precision (ability of a classifier not to label an instance positive that is actually negative), recall (finding all positive instances, also called true positive rate), F1-score (harmonic mean of precision and recall), accuracy (proportion of correctly classified instances), and confusion matrix (also called error matrix). The values of all these metrics are calculated between [0,1] with 1 indicating the best and 0 the worst performance. The Matthews Correlation Coefficient (MCC) is also used to measure the classification quality for different class sizes and ranges between [-1,1] where 1 is perfect prediction, 0 is random prediction and -1 indicates total disagreement between prediction and actual value. The comparison is done with [12] and other well known machine learning algorithms i.e., Support Vector Machine (SVM) and Decision Trees for both stages.

B. Results and Discussion

1) Performance of Stage 0 classifier: Fig. 2 illustrates the performance of logistic regression compared to the naïve Bayes (used in the stage 0 of [12]), SVM and decision tree using precision, recall and F1-score. Regarding precision, it is observed that the decision tree performs best as 70% of the results are positively predicted, followed by the logistic regression with 69% and SVM with 67%. However, naïve bayes performs poorly i.e., 0.6 which means 40% of the results were wrongly classified as positive. This is because for naïve bayes, the precision values for some devices were zero, while it shows a precision value of less than 0.17 for many more.

When looking into the recall metric, we see that the logistic regression performs well as compared to other algorithms i.e., 0.65. However, once again naïve bayes gives the least average recall of all devices as 0.61. The reason for this behavior is that out of 21 classes, 8 classes were 100% incorrectly classified. Furthermore, we found that the logistic regression has high F1-score i.e., 0.67 followed by decision tree (0.66), SVM (0.64) and naïve bayes (0.6). The dotted lines in Fig. 2 show the

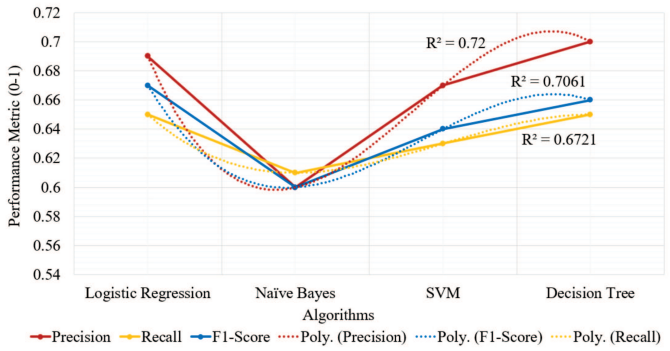

Fig. 2. Metrics comparison for classification Stage 0

algorithmic polynomial trendlines indicating the fluctuation of evaluation metrics with their R-squared values. The trendlines of all metrics give an order 3 trend (i.e., one downward hill and one upward hill) showing polynomial relation among algorithms and their metrics. It is worth noticing that the highest R-squared value among all three trendlines is 0.72 for precision which means precision accounts for 72% of variance.

2) Performance of Stage 1 classifier: Fig. 3 shows the comparison of the stage 1 algorithm i.e., gradient boosting with the random forest (used in stage 1 of [12]), SVM, and decision tree. The gradient boosting performs well, as the precision is 0.9, followed by the 100% correct classification (recall) and 94% of F1 score. The SVM also gives reasonable results followed by the decision tree. However, random forest indicates the worst performance with 0.78 for recall, 0.8 for precision and 0.77 for F1-score, because 3335 training instances of the Belkin switch classs, 374 instances of the HP printer class, 262 instances of the TP link camera class and 31 iPhone class instances were incorrectly classified.

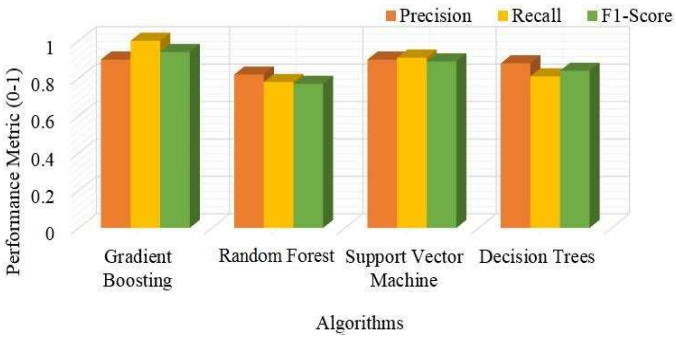

Fig. 3. Metrics comparison for classification Stage 1

In Fig. 4, the gradient boosting gives the highest accuracy i.e., 99% with a significant high MCC value of 0.965671. The accuracy for SVM is 91% with 0.88 MCC and the accuracy for decision tree is 89% with 0.79 MCC. Once more, the random forest presents the worst performance with an accuracy of 0.77 and an MCC of 0.72. Further analysis showed that there are 5 classes out of 21 with incorrectly classified for the random forest and as the accuracy is the ratio of these numbers, we corroborate the poor performance of random forest as shown in Fig. 4.

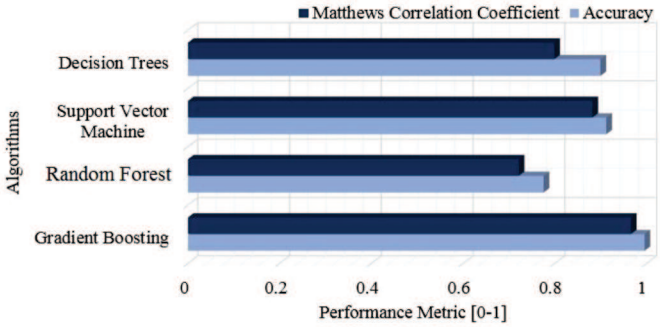

Fig. 4. Accuracy vs MCC for various classification algorithms

Next, we have plotted the performance metrics per device for Stage 1 in Fig. 5. Some devices such as dropCam, iPhone, netatmo weather, smartcam and TP link camera presents the highest performance i.e., recall=1; precision=1 and F1-score=1, all aggregated to 3. However, for TP link plug the aggregated value is 1.31 because the F1 score is 0.2, the recall is 0.11 but the precision is significantly high i.e., 100%. For the smart scale, the aggregated value is 0.94 as precision is reasonably good i.e., 0.67 but recall and F1 score is relatively low i.e., 0.17 and 0.1.

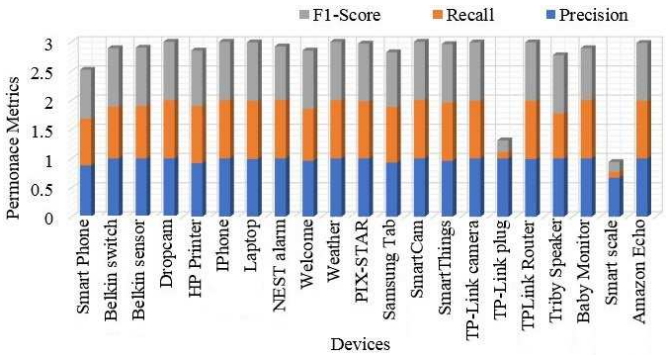

Fig. 5. Performance comparison per device category for Stage 1

3) Overall Performance: In Fig. 6 the row entries of a confusion matrix depicts the actual values and the column entries depicts the predicted values for 21 classes. All the diagonal entries corresponds to correct classification whereas entries above diagonal are all Type I error (also called False Positive Rate (FPR)) and entries below are Type II error (also called False Negative Rate (FNR)). The goal is to minimize the Type I and Type II errors close or equal to zero. At the main diagonal there are three exception cases: (i) for the nest alarm we have the worst classification of 0 since all 19 instances of the particular device were classified as Laptop, thus depicting 100% FNR, (ii) for the withings scale the classification value is 0.29 with 71% FNR, as it was misclassified as Samsung tab, and (iii) for the HP printer, we notice 39% of misclassifications as smart phone (Type II error), and 5% of misclassifications

as TP Link Router (Type I error), while only 56% instances were correctly classified. This is because of the following reasons: (i) there are 21358 instances of laptop compared to 19 nest alarm instances; 2492 Samsung tab instances compared to 20 instances of withings scale; 902 smart phone instances and 42389 TP Link Router instances compared to 362 HP printer instances. Thus, the value of $F_m(x)$ (represented in Eq. 13) for laptops, Samsung tabs, smart phones and TP link routers is high as compared to nest alarm, withings scale and HP printer; (ii) the traffic rate values (feature s_3) were very low for nest alarm and withings scale as compared to the laptop and Samsung tab. Finally, there are 38% server-side ports which are empty for HP printer instances as compared to the smart phone and TP link router, because the HP printer communicates with local devices instead of internet end points.

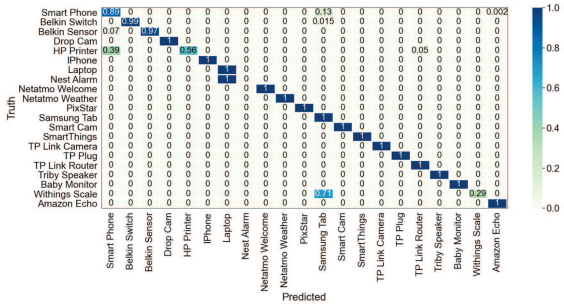

Fig. 6. Confusion Matrix of IoT device classification

VII. Conclusion

In this paper, we have presented a two-stage classification technique that utilizes the network and statistical features to perform the IoT device and traffic classification in the context of a smart city. After data preprocessing, the stage 0 classifier employs a multiclass logistic regression algorithm to classify traffic using the nominal and multivalued attributes as port numbers, IP protocols and MAC addresses. The result of stage 0 is then fed into the stage 1 classifier, which uses a gradient boosting algorithm that takes a second subset of features consisting of IP addresses, TTL, inter-arrival time, traffic flow rate, burstiness rate and packet length for 21 types of IoT devices. Finally, the comparison is done with existing works for both stages in term of recall, precision, F1-score, accuracy, Mathews Correlation coefficient and confusion matrix. The results indicate that multiclass logistic regression in stage 0 and gradient boosting in stage 1 performed better than all other algorithms while achieving a 99% accuracy and a significant MCC of 0.96. Future direction of this work includes incorporation of other machine learning techniques such as K-means clustering along with the classification.

References

[1] N. Ivanov, "Unleashing the Internet of Things with in-memory computing - IoT Now - How to run an IoT enabled business", IoT Now - How to run an IoT enabled business, 2019. [Online]. Available: https://www.iot-now.com/2019/01/17/92200-unleashing-internet-things-memory-computing. [Accessed: 18- Aug- 2020].

[2] M. Laner, P. Svoboda, N. Nikaein and M. Rupp, "Traffic Models for Machine Type Communications," ISWCS 2013; The Tenth International Symposium on Wireless Communication Systems, Ilmenau, Germany, 2013, pp. 1-5.

[3] M. Laner, N. Nikaein, P. Svoboda, M. Popovic, D. Drajic and S. Krco, "Traffic models for machine-to-machine (M2M) communications: types and applications", in Machine-to-machine (M2M) Communications Architecture, Performance and Applications, C. Antón-Haro and M. Dohler, Ed. Woodhead Publishing, 2020, pp. 133-154.

[4] A. Orrevad, "M2M Traffic Characteristics: When Machines Participate in Communication", Ph.D. Thesis, KTH Information and Communication Technology, 2009.

[5] M. Miettinen et al., "IoT Sentinel Demo: Automated Device-Type Identification for Security Enforcement in IoT," 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, 2017, pp. 2511-2514, doi: 10.1109/ICDCS.2017.284.

[6] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray and I. Ray, "Behavioral Fingerprinting of IoT Devices", in ASHES '18: Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, 2018, pp. 41-50.

[7] Y. Meidan, et al., "Detection of Unauthorized IoT Devices Using Machine Learning Techniques", Tech. Report, ArXiv, [Online]. Available: http://arxiv.org/ abs/1709.04647. [Accessed: 20- May- 2020]

[8] N. Apthorpe, D. Reisman, and N. Feamster, "A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic", Tech. Report, ArXiv, [Online]. Available: https://arxiv.org/abs/1705.06805. [Accessed: 20- May- 2020]

[9] R. Lippmann, D. Fried, K. Piwowarski and W. Streilein, "Passive Operating System Identification From TCP/IP Packet Headers", in ICDM Workshop on Data Mining for Computer Security (DMSEC), 2003, pp. 1-10.

[10] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things," in IEEE Access, vol. 5, pp. 18042-18050, 2017, doi: 10.1109/ACCESS.2017.2747560.

[11] Y. Meidan et al., "ProfilIoT: a machine learning approach for IoT device identification based on network traffic analysis Share on", in SAC '17: Proceedings of the Symposium on Applied Computing, Marrakech, Morocco, 2017, pp. 506-509.

[12] A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics", IEEE Transactions on Mobile Computing, vol. 18, no. 8, pp. 1745-1759, 2019. Available: 10.1109/tmc.2018.2866249.

[13] University of New Souths Wales, "IoT Traffic Traces", [Online]. Available: https://iotanalytics.unsw.edu.au/iottraces. [Accessed: 20- May- 2020].

[14] Q. Wang, J. Xu, H. Chen and B. He, "Two improved continuous bag-of-word models," 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, 2017, pp. 2851-2856, doi: 10.1109/IJCNN.2017.7966208.

[15] N. Balakrishnan and A. P. Basu, "Exponential Distribution: Theory, Methods and Applications", CRC Press Taylor & Francis Group, 2018.

[16] D. W. Hosmer, J. Stanley Lemeshow and R. X. Sturdivant, "Applied Logistic Regression", 1st ed. John Wiley & Sons, 2013.

[17] P. Bahad and P. Saxena, "Study of AdaBoost and Gradient Boosting Algorithms for Predictive Analytics", in International Conference on Intelligent Computing and Smart Communication, 2019, pp. 235-244.