International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-11 Issue-7, June 2022

End-to-End Machine Learning Pipeline for Real-Time Network Traffic Classification and Monitoring in Android Automotive

Sriram M, A Susmithaa Raam, B Vignesh, Balasubramanian V

Abstract*: The aim of this work is to build a network traffic monitoring application that is capable of categorizing network data traffic based on their application usage into 7 types: Browsing, Chat, Email, File Transfer, Streaming, VoIP and P2P. Flow-wise data is analyzed after the traffic stream is fed into the CICFlowmeter. Live traffic flow is fed to various ML models and algorithms such as K-Means Clustering algorithm, Agglomerative Clustering, Mean-shift algorithm, Random Forest Classifier, Adaptive Boosting algorithm, Gradient Boosting algorithm, Linear Discriminant analysis, Naive Bayes classifier, Classification and regression trees and the Support Vector Machine model. K-fold cross validation test is conducted, which derived results depicting the best of the models to be the Random Forest Classifier. We used 23 features for model training based on their importances. Model evaluation is done using the confusion matrix. Class imbalances are handled effectively with a comparative study of both under-sampling and oversampling of the dataset. Oversampling using SMOTE produces better results. The important timebased features in classification is recorded for further studies. The model used was fast enough to classify the flows in real time and display the analytics in the dashboard. The Flask framework is used to build a live dashboard to display the network traffic classified along with the several important features. We were able to prove that network traffic classification cam be done using time-based features which does not violate data protection laws. Network traffic classification using Random forest algorithm on oversampled dataset gave an overall accuracy of 0.92 was achieved.

***Keywords*: Machine Learning, Android Automotive, CICFlowmeter, Network Flow Classifier**

I. INTRODUCTION

Android automotive is a full-stack, open source platform and operating system that runs directly on the in-vehicle hardware. It supports a myriad of applications that were built for Android as well as those built for Android Auto, thus aggrandizing the infotainment experience in cars. The multiple applications that run on the platform are always sending and receiving multiple requests/data packets from and to the internet. With the advancement of methods that networks use to regulate traffic, it has become difficult to analyze the network traffic analysis.

This work aims to build an intelligent system that dynamically classifies network traffic into classes that define the logical purpose/usage of the network packets, thus bringing in data transparency. After a systematic perusal and research into the various methods that can be used to fulfill our goal, ta conclusion to devise a system that will classify network data into classes based on the human purpose for the applications that send/receive the data packets into the network, was derived.

The project is aimed at enabling data transparency by displaying data traffic details for on a live dashboard, for the concerned user. The CICFlowmeter is a tool that is primarily used to generate the essential data attributes for traffic so that it may be classified with high accuracy. The Canadian Institute of Cybersecurity distributes CICFlowMeter, a network traffic flow generator that generates 84 network traffic properties. It reads PCAP files and produces a report with the extracted features, as well as a csv file of the report. It's also capable of recording real-time network traffic.

Manuscript received on 23 May 2022.
Revised Manuscript received on 30 May 2022.
Manuscript published on 30 June 2022.

* Correspondence Author

Sriram M, UG Student, Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai (Tamil Nadu), India. sriram18168@cse.ssn.edu.in

Susmithaa Raam A, UG Student, Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai (Tamil Nadu), India. Email: susmithaa18181@cse.ssn.edu.in

Vignesh B, UG Student, Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai (Tamil Nadu), India. Email: vignesh18193@cse.ssn.edu.in

Dr. Balasubramanian V*, Associate Professor, Department of Computer Science and Engineering, Sri Sivasubramaniya Nadar College of Engineering, Chennai (Tamil Nadu), India. Email: balasubramanianv@ssn.edu.in

II. PROBLEM STATEMENT

Data transparency plays a major part in the authenticity of a system that deals with the user's private information. Android automotive has access to the car owner's private data and the ease with which information is accessed on edge devices, has consumers concerned. Data packets containing sensitive information have potential to be accessed by parties with malicious intent. This risk of the loss of data privacy can be mitigated by bringing in data transparency. This work aims at building an end-to-end ML pipeline that dynamically classifies network traffic into classes that identify the logical purpose/usage of the application that deals with said network packets, thus bringing in data transparency. Data transparency is ensured with the help of a live dashboard that was built with the goal of keeping the user data aware 24*7.

The problem is broken down into a sequence of subtasks:

- Data Preprocessing
- Construction of Multiclass Classifier
- Model Selection
- Live capture of network flows
- Backend development of web app
- Integrating model into the application
- Building a live GUI dashboard

III. LITERATURE SURVEY

Lashkari et al. presented a time analysis on Tor traffic flows, captured between the client and the entry node[1]. Two scenarios were defined where one was to detect the application type: Browsing, Chat, Streaming, Mail, VoIP, P2P, or File Transfer and other was to detect Tor traffic flows. The flow was defined as a sequence of packets with the same values for {Source Port, Destination Port, Source IP, Destination IP, and Protocol (TCP or UDP)}. For this experiment, they have used a application, the CICFlowmeter (ISCXFlowMeter, 2016) in order to generate the flows and calculate the required parameters. They tested the results obtained with various algorithms:C4.5, Random Forests and KNN, and 10-fold cross-validation is applied and the accuracy obtained is less than 90%. They grouped the classes in two sets, depending on the results obtained. In the first group, they have the classes with good precision and recall results: AUDIO, FT, VOIP, P2P, and VIDEO. In yet another group, they have the classes their classifier fails to obtain good results: CHAT, BROWSING and E-MAIL. In the paper, Characterization of Encrypted and VPN Traffic using Time-related Features[2], the efficacy of flow and time-related features to detect VPN traffic and to characterize encrypted traffic into different categories was studied. KNN and C4.5 algorithms are used to test the accuracy and precision of features. Abuthawabeh et al.[3] proposed to remove instances or features that have the following issues: a) Remove un-useful instances like intranet ones. For eg, i) 0.0.0.0:Internal routing ii) 8.8.8.8: Google DNS IP b) Removing unique flow features such as source IP and Destination IP. Ensemble learning techniques were used to select most important features. Extra-trees classifier has a accuracy of 79.97\% which is higher compared to other studies. In the paper, A framework for Android Malware detection and classification [4], they used app specific features. The planned methodology showed the common accuracy 90% for 5 classifiers which were Decision Tree, Random Tree, Random Forest, K-Nearest Neighbor, and Regression etc. Random forest was one of the best performing algorithms. In the paper, Application identification via network traffic classification [5], they explored ML methods for finding application via network traffic classification. Unlike the existing categories like FTP, IM, etc. used for classification, they used apps such as Facebook, Twitter, GMail etc. A stacked sparse autoencoder (SSAE) based semi-supervised classification model for network traffic classification. It makes use of labeled and unlabeled data. An unsupervised algorithm based on the SSAE is first pre-trained to obtain layer-by-layer initialization parameters. Then, a supervised neural network classifier is linked to the code layer of the SSAE and is fine-tuned by [7]. A method of building deep and parallel network-in-network (NIN) models for encrypted network traffic classification performs better than CNN[8]. jRip algorithm is one of the best algorithms with precision upto 1 for network traffic classification[9]. Deep Learning removes the need for data preprocessing, and they perform well on malware detection problems [10][15][16]. In the paper, Inferring Application Type Information from Tor Encrypted Traffic [11], they use a method based on HMM (Hidden Markov Models) to classify Tor traffic in 4 categories: P2P, FTP, IM, and Web with accuracy of 90%. A new method for assigning appropriate labels to create a 28 Standard Android Botnet Dataset (28-SABD) [12], ensembled KNN was used to improve accuracy. Due to the complexity of the learning models, Shapley Additive explanations (SHAP), an explainable AI methodology, has been adopted to explain and interpret the classification decisions of ML models[13].Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification[14] shows that K-Means have higher accuracy than Expectation-Maximization.

IV. PROPOSED SYSTEM

The System Architecture is shown in Fig. 1. The system comprises of 4 layers: Network Layer, ML Layer, Application Layer, User Layer. Each of these layers are explained in detail below.

A.Network Layer

Traffic flows are retrieved from the Network Interface Card (NIC), stored, and pre-processed in real time at the Network Layer. The traffic stream is then sent into CICFlowMeter, an open-source tool for extracting time-related data from network flows in order to train machine learning models. For both TCP and UDP protocols, CICFlowMeter can produce these features from bidirectional flows, filter properties from an existing feature set, and manage the duration of flow timeout.

CICFlowMeter is a traffic flow generator and analyzer for networks. It can be used to create bidirectional flows, with the first packet determining the forward (source to destination) and backward (destination to source) directions. As a result, more than 80 statistical network traffic features can be calculated separately in the forward and backward directions, such as Duration, Number of packets, Number of bytes, Length of packets, and so on. Selecting features from a list of current features, adding new features, and adjusting the duration of flow timeout are all additional functionalities. The application's output is a CSV file containing more than 80 network traffic analysis elements and six columns for each flow (FlowID, SourceIP, DestinationIP, SourcePort, DestinationPort, and Protocol). TCP flows are important to keep in mind.

B. Machine Learning (ML) Layer

The machine learning models employed are described in the ML Layer. It also depicts the machine learning process that operates beneath the ML model. The pipeline streamlines the machine learning process by allowing live traffic flows to be converted and coupled into each ML model to reach the necessary categorization results.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

It also serves as a link between the Flask GUI and the machine learning models. Our Machine Learning procedures were turned into separate, reusable, modular components that could then be pipelined together to create a more efficient and streamlined real-time traffic classifier using the ML pipeline. Dataset Description: The dataset contains 7 types of traffic (browsing, chat, Streaming, mail, VOIP, P2P, and File Transfer) from more than 18 representative applications (e.g., Facebook, Skype, Spotify, Gmail, etc.).

In our work, we make use of a combination of 2 datasets: captured with all the flow timeout values (15s, 30s, 60s, 120s). The Tor and VPN specific labels present in the dataset are normalized to the 7 classes as mentioned above. We use features that are common to all the datasets. The dataset consists of 23 features and a label field.

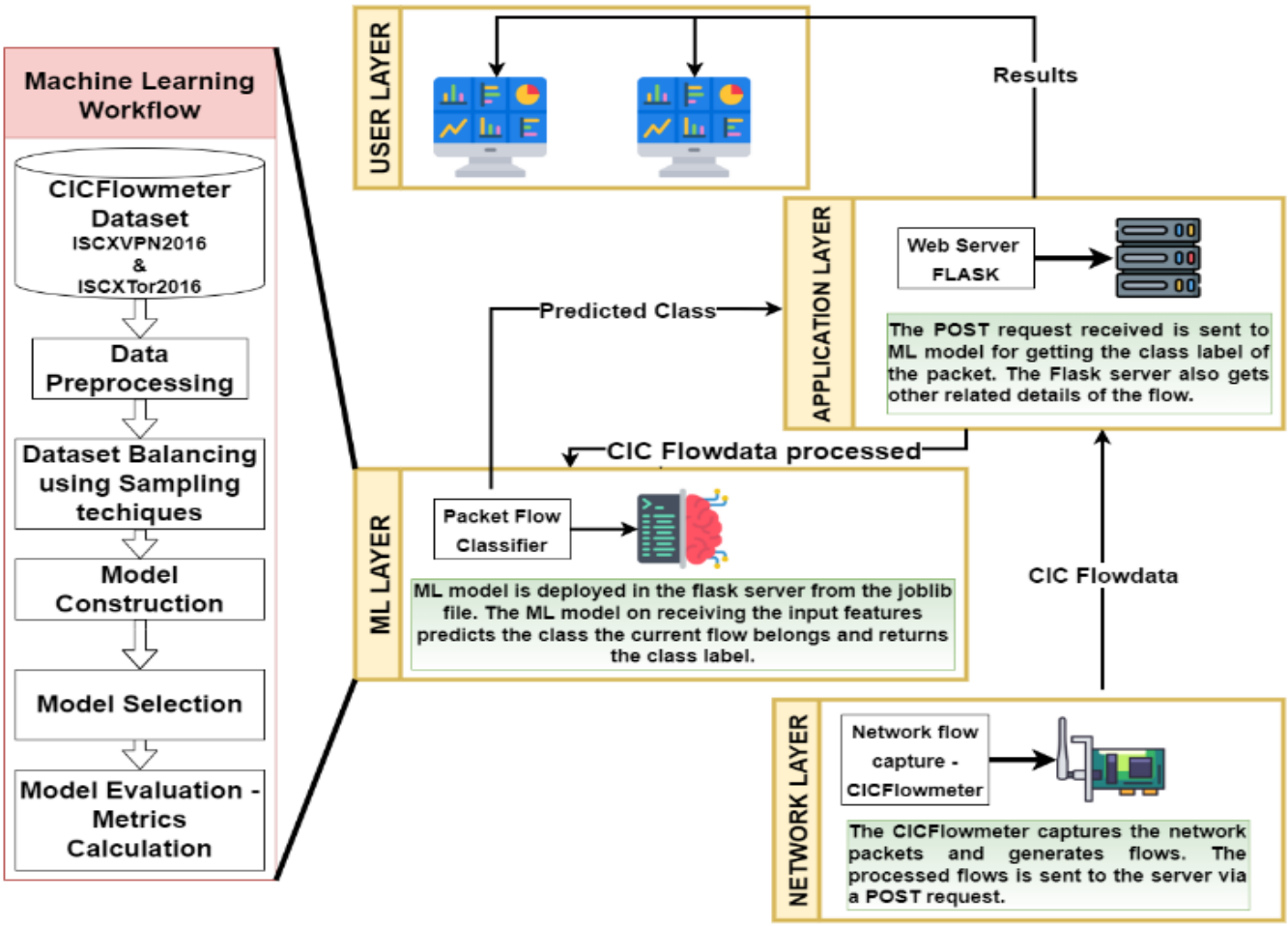

Figure 1: Block diagram of the network traffic classification pipeline

The dataset consists of 79967 flow data. The distribution of classes in the dataset is shown in below Fig. 2. It is evident that BROWSING class has the maximum count and EMAIL has the minimum count. From the distribution, it is evident that the dataset is not balanced.

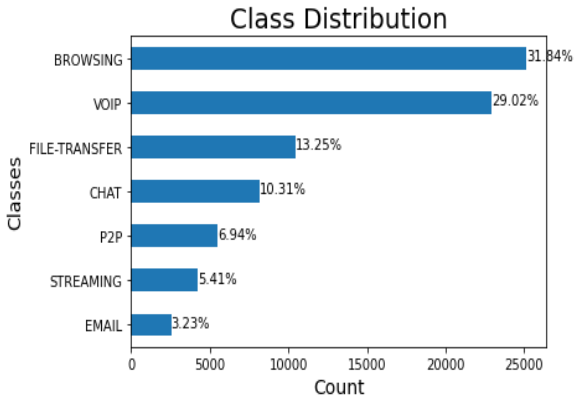

Figure 2: A Bar graph showing the distribution of 7 classes in the entire dataset.

1) Definition of Features: A flow is a collection of packets sent in both the forward (source to destination) and backward (destination to source) directions between two points. Flow duration is the time it takes for a flow to complete in microseconds. The forward inter-arrival time (Fwd IAT) is the delay between two packets arriving in the forward direction. The forward inter-arrival time (Fwd IAT) is the time that passes between two packets sent in the forward direction. The forward inter-arrival time (Bwd IAT) is the time between two packets sent in the opposite direction. The active time of a flow indicates how long it was active before becoming idle. The idle period of a flow indicates how long it was inactive before it became active. Flow duration, Flow IAT Mean, Flow IAT Max, Flow IAT Min, Fwd IAT Min, Fwd IAT Max, Fwd IAT Mean, Fwd IAT Total, Bwd IAT Min, Bwd IAT Max, Bwd IAT Mean, Bwd IAT Total, Active Min, Active Mean,

Active Max, Active Std, Idle Min, Idle Mean, Idle Max, Idle Std, Flow Bytes per second, Flow Packets per second and Flow IAT Standard Deviation were used in our analysis.

2) Sampling Methods: A clear class imbalance was noticed after the exploratory data analysis as shown in Fig. 1. Thus, we had to adopt techniques to remove the class imbalance. We experimented with both oversampling and undersampling techniques. Even the oversampling methods gave better results than the undersampling methods, it was noted that whilst oversampling, several data points were created that could not have existed in real-world scenarios. This could contribute to unnecessary biases while training the model. Thus, we restricted our study with the under sampled dataset.

SMOTE (Synthetic Minority Oversampling Strategy) is a statistical technique for evenly increasing the number of cases in your dataset. The component generates new instances based on existing minority situations you provide as input. The number of majority cases does not change as a result of SMOTE adoption. SMOTE generated an oversampled dataset with a total of 17608 classes.

Random Undersampling Techniques is an undersampling technique that randomly selects samples with or without replacement to undersample the majority class(es). Undersampling resulted in a dataset with 1758 classes.

Several machine learning models were tested, and the most optimal model was chosen based on the k-fold analysis test. The machine learning models tested out are KMeans Clustering algorithm, Agglomerative Clustering, Mean-shift algorithm, Random Forest Classifier, Adaptive Boosting algorithm, Gradient Boosting algorithm, Linear Discriminant analysis, Naive Bayes classifier, Classification and regression trees and Support Vector Machine model. The optimal model chosen was the Random Forest Classifier. The Random Forest classifier was trained on the sampled datasets and the results were recorded.

A Random Forest Classifier is a meta estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting a number of decision tree classifiers on various sub-samples of the dataset.

C.Presentation layer - Application and User layer

The web-based interface operating on a flask server is presented in the Application Layer. A user-friendly web application that allows users to interact with the system using the Python Flask Framework was created.

To improve the user's experience with the system, the user interface is designed as a responsive and user-friendly online application. We used the Python Flask Framework, HTML5, CSS3, JavaScript, and JSON to create the web application. All web pages are meant to be device-agnostic, meaning they may be seen on mobile devices as well as desktop computers. We had to encapsulate models created on scikit learn as a Representational State Transfer (REST) API using the Flask web framework to execute the web application on top of the Random Forest model. REST is a software architecture approach that allows heterogeneous computer systems connected via the internet to communicate with one another. The REST API is used to coordinate all interactions between the model and Flask. When the user captures fresh traffic flow, Flask app receives the flow from the POST method implemented in CICFlowmeter. The traffic flow data from the app is sent to the Random Forest Classifier model. The model predicts the class the current flow belongs to and this result is sent to the User Layer i.e., on-board GUI.

V. JUSTIFICATION FOR MODEL SELECTION

The various algorithms which try to analyze for our project are Random Forest Classifier, AdaBoost, Gradient Boosting, Linear Discriminant Analysis, K Nearest Neighbors, Naive Bayes, Support vector machines, and Decision Trees.

Interpretability, Accuracy, Rate of Convergence, Model assumptions, and the Nature of data itself, are the key factors into account when deciding the learning algorithm for classification.

Algorithms like Decision Trees and Random Forests generate models that are easily interpretable. In certain situations, we find that we want to know what our model is doing to the data in order to make better decisions. Logistic regression, SVM, Neural Nets, etc. lag in this regard since they are like a “black box” for the user.

Each algorithm provides different accuracy for different scenarios. Most often, the aim is to maximize accuracy. But when presented with datasets like ours containing highly unbalanced classes, often the aim is to maximize the correct classification of the less-occurring class. In such cases, overall accuracy is not the topmost priority. Hence, algorithms like logistic regression that tend to maximize the overall accuracy cannot be used in such circumstances, while decision trees can be due to their ability to incorporate class bias.

Algorithms like Logistic regression and Gradient Boosting assume that the data is linearly separable. Decision Trees do not assume data to be linearly separable, but instead, assume those decision boundaries lie parallel to the coordinate axes. Random forests assume that averaging outperformance on multiple decision trees is a safer bet. The power of numerous decision trees is combined in a random forest. It does not reliant on a single decision tree's feature relevance. A set of features is given a lot of weight in the decision tree model. During the training phase, however, the random forest selects features at random. As a result, it is not overly reliant on any particular set of characteristics. RFC uses the wisdom of the crowd principle, which means that if a small group of trees makes a bad judgement, it has no bearing on the main conclusion. Because one tree is largely uncorrelated with another, an error in one tree does not propagate to other trees, and other trees are unaffected by this choice. As a result, the random forest can better generalize over the data. A random forest is substantially more accurate than a decision tree because of the randomized feature selection.

Certain algorithms like SVMs, Neural Nets, and Random Forests have a slower rate of convergence compared to other algorithms like Random Forest and Decision Trees. Thus, the Random Forest classifier is comparatively faster during training on large datasets.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

The nature of data is the most important factor in deciding on which algorithm to deploy. Since most of the features in our data are categorical in nature, algorithms such as trees are often considered a good choice since they provide for easily interpretable models. Since no distinct linear separation of classes could be concluded, we can rule out algorithms such as SVM, Logistic Regression, KNN etc.. When we have a lot of features in our data compared to the number of data points (this is often the case in text analytics), trees are often not a good choice. We will simply not be able to generalize well with trees in that case. Other algorithms like logistic regression and SVM will generalize much better in this case. In our case, we have a reduced feature space and there are no linear separations of classes in our dataset. Hence, Random forest is one of the best algorithms to be considered for our purpose.

VI. RESULTS AND DISCUSSIONS

A. Classification Model Selection for ML Layer

The network flows with the input features as defined in the above section was fed into various machine learning models. A 10-folds cross validation test was conducted to identify the best machine learning model. From the results as shown in Table. 1, it was evident that Random Forest Classifier performed the best with the highest mean accuracy and lowest Std Deviation in the accuracy making it the most reliable and accurate model from the lot. Random Forest classifier produced an accuracy of 92%.

Table I. Summary of Accuracies of Ml Algorithms

ML Algorithm	Mean Accuracy	Standard Deviation of Accuracy
Random Forest	0.919560	0.002414
Adaboost	0.650735	0.018146
Gradient Boosting	0.848444	0.004722
Linear discriminant analysis	0.452446	0.007493
Classification and regression trees	0.893477	0.003679
Naive Bayes	0.198242	0.014723
Support Vector Machine	0.534864	0.006807
K-Nearest Neighbour	0.775448	0.005420

B. Results of Random Forest Classifier

From the Table. 2, it is evident that BROWSING, FILE-TRANSFER, P2P and VOIP have higher recall compared to other classes. CHAT, EMAIL and STREAMING are the classes that comparatively fail to get classify correctly. For RFC the overall accuracy obtained is 0.92 on the entire dataset, 0.92 on the oversampled dataset and 0.87 on the under sampled dataset. Thus, SMOTE produces better results and can used for classification application in real-time.

C. Feature Importance

Feature importance was computed for this dataset using the Random Forest Classifier as shown in Fig. 3. It is easy to understand the impact of each of the feature in classification. Features like flow bytes per second, duration of flow, flow packets per second have more importance compared to other features.

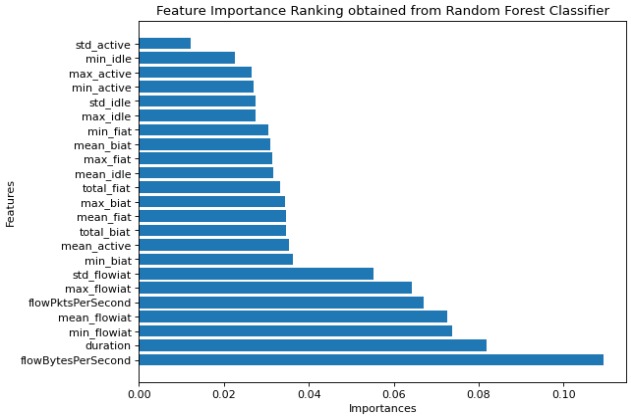

Figure 3: Bar Graph showing feature importance of Random Forest Classifier

D. Comparison with existing works

In the paper by Lashkari et al.[1], C4.5 and KNN were able to produce an f1-score around 80\%. In the paper by Gil et al.[2], 0.843,0.788,0.705 were the accuracy got for the RFC, C4.5 and KNN respectively. We need to take into the fact that different features were used for the above works. In the paper by Hu et al.[6], they were able to classify into 4 classes using RFC at an accuracy of 90\% by using 26 features.

TABLE II. Results of Random Forest Classifier on Sampled Datasets

Class Name	Original dataset			Oversampled			Undersampled		
	Precision	Recall	f1-score	Precision	Recall	f1-score	Precision	Recall	f1-score
BROWSING	90	96	93	94	91	92	93	83	88
CHAT	85	81	83	77	85	81	66	81	73
EMAIL	87	82	84	77	84	80	63	89	74
FILE-TRANSFER	90	85	87	89	84	86	84	7	78
P2P	94	91	92	92	92	92	82	92	87
STREAMING	85	84	85	82	88	85	70	87	78
VOIP	99	99	99	99	98	99	99	97	98

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

The study also reveals that when the feature extraction can accurately describe traffic characteristics, the DL approach does not perform well. We make use of 23 features and are able to 92\% as average accuracy when we apply 10-fold cross validation. The feature-set which we have used for training the model hasn't been been used earlier in any other research. Our model is able to predict all the classes with a accuracy greater that 80\% in contrast to the earlier papers where the individual class accuracy was ranging between 55\% to 90\%. Hence, it is clearly evident that our model with the given feature-set outperforms the existing models.

E. Justification of the results

We choose Random Forest classifier for our project as it provides higher accuracy and f1-score through cross-validation. In the model, RFC avoids overfitting trees. It's also capable of handling huge data sets with increased dimensionality. RFC models have the added benefit of not requiring data to be rescaled or modified.

There isn't much that needs to be done in terms of pre-processing. It is capable of handling binary, category, and numerical features. The Random Forest classifier is parallelizable, which means we can run it on several machines. As a result, the computation time is reduced. Boosted models, on the other hand, are sequential and would take longer to compute.

Outliers are handled by Random Forest by effectively binning them. It is also unconcerned about non-linear characteristics. Because it uses many decision trees to arrive at its output, this method is also quite robust. It includes methods for balancing errors in uneven data sets' class populations. When we have an uneven data set, random forest aims to reduce the overall error rate, therefore the larger class will have a low error rate while the smaller class will have a higher error rate. The variation of each decision tree is high, but the bias is modest. However, because we average all of the trees in a random forest, we also average the variance, resulting in a model with low bias and moderate variation. Random forest makes evaluating the importance of variables, or their contribution to the model, simple. There are a few methods for determining the relevance of a feature. When a feature is removed from a model, the Gini importance and mean drop in impurity (MDI) are commonly used to determine how much the model's accuracy falls. Because we are only working with a subset of features in our model, it is faster to train than decision trees. We can easily work with hundreds of features. Because we can save created forests for future applications, prediction speed is substantially faster than training speed. Prediction speed is significantly faster than training speed because we can save generated forests for future uses. Hence RFC is the best model based on the results and findings.

VII. CONCLUSION

This End-to-End Machine Learning Pipeline for real-time network traffic classification and monitoring in Android Automotive classifies network flows only using time-based features. This is in compliance to General Data Protection Regulation and does not use any personal data. We use CICFlowmeter to capture flows. We exhaustively compare the performance of 8 different machine learning models on the CICFlowmeter dataset using K-Folds test. Random Forest classifier performs the best. In order to overcome the problem of the unbalanced dataset, we used SMOTE and Random undersampling techniques and the former one produced better result. Thus, traffic classification based on flows provides better insights to the users and can be extended to all devices.

ACKNOWLEDGEMENT

We would like to extend our gratitude to Datator for sharing their inputs and thoughts which helped us in aligning this project to satisfy the industry needs.

REFERENCES

1. Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun and Ali A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features", In the proceeding of the 3rd International Conference on Information System Security and Privacy, SCITEPRESS, Porto, Portugal, 2017 [CrossRef]
2. Gerard Drapper Gil, Arash Habibi Lashkari, Mohammad Mamun, Ali A. Ghorbani, Characterization of Encrypted and VPN Traffic Using Time-Related Features", In Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), pages 407-414, Italy, 2016 [CrossRef]
3. M. K. A. Abuthawabeh and K. W. Mahmoud, "Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features," 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 42-47, doi: 10.1109/ACIT47987.2019.8991114. [CrossRef]
4. M. Murtaz, H. Azwar, S. B. Ali and S. Rehman, "A framework for Android Malware detection and classification," 2018 IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS), 2018, pp. 1-5, doi: 10.1109/ICETAS.2018.8629270. [CrossRef]
5. B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz and M. E. Karsligil, "Application identification via network traffic classification," 2017 International Conference on Computing, Networking and Communications (ICNC), 2017, pp. 843-848, doi: 10.1109/ICCNC.2017.7876241. [CrossRef]
6. Y. Hu, F. Zou, L. Li and P. Yi, "Traffic Classification of User Behaviors in Tor, I2P, ZeroNet, Freenet," 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2020, pp. 418-424, doi: 10.1109/TrustCom50675.2020.00064. [CrossRef]
7. Aouedi, O., Piamrat, K., and Bagadthey, D. (2020) 'A semi-supervised stacked autoencoder approach for network traffic classification', In proc. of 28th IEEE International Conference on Network Protocols (ICNP), pp. 1–6. [CrossRef]
8. Bu, Z., Zhou, B., Cheng, P., Zhang, K., and Ling, Z.-H. (2020), 'Encrypted network traffic classification using deep and parallel network-in-network models', IEEE Access, Vol. 8, pp. 132950-132959. [CrossRef]
9. Cuzzocrea, A., Martinelli, F., Mercaldo, F., and Vercelli, G. (2017), 'Tor traffic analysis and detection via machine learning techniques', In proc. of IEEE International Conference on Big Data (Big Data), pp. 4474–4480. [CrossRef]
10. Gohari, M., Hashemi, S., and Abdi, L. (2021), 'Android malware detection and classification based on network traffic using deep learning', In proc. of IEEE 7th International Conference on Web Research (ICWR), pp. 71–77. [CrossRef]
11. He, G., Yang, M., Luo, J., and Gu, X. (2014), 'Inferring application type information from tor encrypted traffic', In proc. of IEEE Second International Conference on Advanced Cloud and Big Data, pp. 220–227. [CrossRef]
12. Moodi, M. and Ghazvini, M. (2019), 'A new method for assigning appropriate labels to create a 28 standard android botnet dataset (28-sabd)', Journal of Ambient Intelligence and Humanized Computing, Vol. 10, pp. 4579–4593. [CrossRef]

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

13. Sarhan, M., Layeghy, S., and Portmann, M. (2021), 'Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection', arXiv Computing Research Repository, pp.1-12.
14. Singh, H. (2015), 'Performance analysis of unsupervised machine learning techniques for network traffic classification', In proc. of IEEE Fifth International Conference on Advanced Computing Communication Technologies, pp.401–404. [CrossRef]
15. Wan, J., Wu, L., Xia, Y., Hu, J., Xia, Z., Zhang, R., and Wang, M. (2019), 'Classification method of encrypted traffic based on deep neural network', In proc. of Springer International Conference of Pioneering Computer Scientists, Engineers and Educators, pp. 528–544. [CrossRef]
16. Zhao, S., Ye, K., and Xu, C.-Z. (2019), 'Traffic classification and application identification based on machine learning in large-scale supercomputing center', In proc. of IEEE 21st International Conference on High Performance Computing and Communications, pp. 2299–2304. [CrossRef]

AUTHORS PROFILE

Sriram M., Final year student currently pursuing an undergraduate degree in Computer Science and Engineering from SSN College of Engineering. Areas of research include Machine Learning, Deep Learning and Cyber Security.

A Susmithaa Raam, Final year student currently pursuing an undergraduate degree in Computer Science and Engineering from SSN College of Engineering. Areas of interest include Machine learning, software development and Internet of Things

Vignesh B, Final year student currently pursuing an undergraduate degree in Computer Science and Engineering from SSN College of Engineering. Areas of interest include Software engineering, data science and Machine learning.

Dr. V. Balasubramanian, Associate Professor in the Department of Computer Science and Engineering and has 23 years of teaching and research experience. He received his B.E. in Electronics and Communication from University of Madras, in the year 1997 and M.E. degree in Computer Science and Engineering from College of Engineering, Guindy, Anna University, Chennai, in the year 2002. He received his Ph.D from College of Engineering, Guindy, Anna University. He has published 20 papers in National, International Conferences and Journals. He has authored five books: Theory of Computation, Computer Networks, Computer Architecture, Design and Analysis of Algorithms and Artificial Intelligence. He is a Senior member of IEEE and ACM and also a Life member of Computer Society of India (CSI), and Indian Society for Technical Education (ISTE).

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)