# The Same, Only Different: Contrasting Mobile Operator Behavior from CrowdSourced Dataset

Konstantinos Kousias
Simula Research Laboratory
Norway

Cise Midoglu
Simula Research Laboratory
Norway

Özgü Alay
Simula Research Laboratory
Norway

Andra Lutu
Simula Research Laboratory
Norway

Antonios Argyriou
University of Thessaly
Greece

Michael Riegler
Simula Research Laboratory
Norway

***Abstract*—Crowdsourcing mobile network performance evaluation is rapidly gaining popularity, with new applications aiming to deliver more accurate and reliable results every day. From the perspective of end-users, these utilities help them estimate the performance of their service provider in terms of throughput, latency and other key performance indicators of the network. In this paper, we build a machine learning based framework, ORCA, to determine and characterize the behavior of Mobile Network Operators (MNOs) from crowdsourced dataset. We investigate whether one can differentiate MNOs by using crowdsourced end-to-end network measurements. We consider different performance metrics (e.g. dowload/upload rate, latency, signal strength) and study the impact of them individually but also collectively on differentiating MNOs. We use *RTR Open Data*, an open dataset of broadband measurements provided by the Austrian Regulatory Authority for Broadcasting and Telecommunications (RTR), to characterize the three major mobile operators and two virtual operators in Austria. Our results show that ORCA can be used to identifying patterns between various mobile systems and disclosing their differences from the end user perspective.**

## I. Introduction and Motivation

The use of Mobile Broadband (MBB) networks has exploded over the last few years due to the immense popularity of mobile devices such as smart phones and tablets, combined with the availability of high-capacity 3G/4G mobile networks. Therefore, understanding the underlying mechanisms that dictate user performance and reliability of MBB networks is of great importance towards smooth operation and future improvements.

One challenge with understanding MBB networks is that they consist of several heterogeneous and complex components that are intertwined into a system. A simplified 4G Long-Term Evolution (LTE) network architecture has two main building blocks: a Radio Access Network (RAN) and a Core Network (CN). The RAN is composed of radio base stations whose primary responsibility is to mediate access to the provisioned radio channel and transport the data packets to and from the user's device. The CN connects the radio network to the public Internet and is responsible for the data routing, accounting, and policy management. Although these same components exist in every operator's network, users of each network typically experience considerably different performance due to the different deployment of base stations, the number of users each network serves, different network configurations, and different traffic policies across operators. In addition, the environment is highly dynamic in terms of physical channel conditions, applications, and users. The first question we are trying to answer in this paper is, how visible these differences are to the end users. In other words, can we differentiate the operators and characterize the operator behavior by just looking into the end user measurements?

In mobile networks, to assess the quality experienced by end users, certain network performance metrics are collected typically via end-to-end network measurements [13], [9], [11], [8]. One popular approach for performing end-to-end measurements of MBB performance has been to rely on end users to run performance tests by using a special measurement application. Such a crowdsource approach is accepted as norm today since it can collect millions of measurements from different regions, networks and user equipment. For example, Okla's speedtest [3]) evaluates the operator performance and provides awards to the fastest operator in a country [5]. Such results are then commonly used by the operators in their marketing announcements. Similarly, regulators use the results of crowdsourced mobile applications such as RTR Nettest [4]) to evaluate whether the operators fulfill their obligations. In all these applications, the most popular parameter evaluated is the *downlink data rate*. Operators are benchmarked against each other using Download (DL) data rate, therefore it is the parameter that each operator aims to differentiate itself from the other operators. The second question we are trying to answer in this paper is, how to determine the fundamental network parameters that differentiate operators. Is it DL data rate? Or is it Upload (UL) data rate? Or maybe latency or wireless signal strength? Is one parameter dominant or do we need combination of certain parameters?

Our goal in this paper is to understand whether different popular performance parameters can be used to differentiate the performance of mobile operators using crowdsourced dataset. To tackle this problem, we propose a methodology that captures and discloses the unique behavior of the operators by identifying patterns, namely *ORCA: Operator Classifier*.

ORCA is designed to leverage large crowdsourced datasets. Such datasets are composed of a rich set of features such as end-to-end measurements of latency, DL/UL data rate, together with network-level characteristics such as wireless signal strength, from a large number of measurement vantage points spread throughout different regions, numerous networks and types of user equipment. However, crowdsourced datasets are known to be noisy and naturally unbalanced [14] due to experiments running at users' own will. To eliminate this noise and imbalance, ORCA introduces a set of pre-processing and filtering steps. Then, it follows a learning process based on a training-validation-testing format and builds a classification model by choosing the best features that are representative of selected operators. The resulting model is then used to investigate whether one can differentiate operator behavior using a crowdsourced dataset.

Our results reveal that DL data rate alone is not enough to differentiate operators, while the latency often is the best parameter to do so. In all cases, the combination of different performance parameters allow us to better distinguish operators. More importantly, ORCA reveals the operating regime for operators in which such differentiation can occur. Therefore, we can answer more detailed questions of the form: what is the range of the average latency that is representative of a certain operator? The end result is that ORCA can differentiate native operators (i.e. operators that own their network infrastructure) with high accuracy. Further analysis using ORCA illustrates how virtual operators are treated compared to the infrastructure owners: while native operators prioritize their own users, they do not differentiate between virtual operators. The basic framework of ORCA is relevant for many potential applications that could be explored as part of our future work: Improving the robustness of ORCA would allow a more accurate match between pricing and claimed offered QoS, hints regarding the different CN and RAN structures, detection of performance bottlenecks and network problems.

## II. Dataset

There is an increasing amount of attention from both academia and industry towards crowdsourced approaches for measuring MBB performance via end-user devices. Such efforts include *Speedtest* [3], *OpenSignal* [1], *MobiPerf* [2], and *RTR-Nettest* [4]. Among available platforms, *RTR-Nettest* is the only one that provides its source code together with the complete open dataset, called *RTR Open Data*. We use this dataset for the analysis and exploration of operators' characteristics as captured from the end-user perspective.

*RTR-Nettest* is a measurement platform launched by the RTR in 2013. It measures Quality of Service (QoS) parameters such as data rate and latency, as well as signal strength, geolocation, network and device type, with a timestamp for each measurement. A measurement in the *RTR-Nettest* platform consists of 6 stages: *Initialization*, *Pre-Test Download*, *Ping Test*, *Download Test*, *Pre-Test Upload* and *Upload Test*. *Initialization* consists of the client connecting to the Control Server and undertaking necessary authentication procedures before making a measurement request, which, when granted, starts the communication between the client and the Measurement Server. This exchange is very brief and consists of an almost-constant number of packets. Once the client establishes a connection with the server, the Pre-Test Download phase follows. Both of the Pre-Test phases are undertaken with the same purpose: to ensure that the Internet connection is in an "active" state, i.e. that dedicated radio resources are available. During this phase, the client requests and the server sends a data chunk in each active thread. While the duration of the phase has not exceeded its nominal value, the client requests a data block of double size compared to the last iteration step. The transfer of the last data block will be finished even if the duration has already exceeded the nominal value. The Pre-Test Upload phase works analogously to the Pre-Test Download phase, but with the client as the sender and the server as the receiver.

The Ping Test consists of the client sending a certain number of Transmission Control Protocol (TCP) "ping"s in short intervals to the server to test the latency of the connection. This exchange is also very brief and consists of an almost-constant number of packets. The Download Test and Upload Test are the main components of the measurement where multiple TCP threads are opened and within each of these, the receiver side simultaneously requests and the sender side continuously sends data streams consisting of fixed-size chunks. After the nominal duration, the sender stops sending further chunks on all connections, the last chunk per each thread is allowed to transmit completely, and the DL/UL data rate of the connection is estimated. We refer the reader to the *RTR Open Data Interface Specification* [4] for a complete list of available parameters and their descriptions.

**RTR Fields:** *RTR Open Data* currently provides up to 67 features which are grouped in 6 categories: Test, location, device, network, coverage and performance related features. For the scope of this paper we use 11 fields from the dataset. `time_utc` indicates the date and time (in UTC) for the measurement. Other test-specific parameters like identifiers of the test and relative start timers do not contribute to our model and therefore are discarded from the dataset. In our analysis, we do not consider geo-location characteristics (e.g the latitude or the longitude of client's position, country name, the distance a client moved during the measurement, etc), hence, no location related parameter is used. `platform` indicates the device platform (Android/iOS for mobiles), and `model` indicates the device model name. We do not consider the software version of the client (e.g for Android 2.1.1, 2.1.2, etc). For identifying networks, we use `network_type` which indicates the technology (e.g UMTS, GSM, 3G, LTE, etc.), and a combination of `sim_mcc_mnc` and `network_mcc_mnc` which indicate the Mobile Country Code (MCC) and Mobile Network Code (MNC) as read from the Subscriber Identification Module (SIM) card (i.e. home network), and the network that is currently used (i.e. access network), respectively. With this information we can identify cases of roaming. We use LTE signal strength information in the form of Reference

TABLE I: Model features.

| Idx | Feature | Description | IG |
|---|---|---|---|
| 1 | ping_ms | Latency ($ms$) | 0.21 |
| 2 | upload_kbit | UL data rate ($Kbps$) | 0.16 |
| 3 | download_kbit | DL data rate ($Kbps$) | 0.11 |
| 4 | lte_rsrp | LTE signal strength ($dBm$) | 0.06 |
| 5 | lte_rsrq | LTE signal quality ($dB$) | 0.05 |
| 6 | hour | Hour of the day | 0.02 |
| 7 | weekend | Weekend indicator | 0.01 |

Signal Received Power (RSRP) from the field lte_rsrp, and Reference Signal Received Quality (RSRQ) from the field lte_rsrq. signal_strength is not an LTE related feature and is dropped accordingly. Similar to several other datasets, *RTR Open Data* includes a series of QoS-related parameters, namely download_kbit, upload_kbit and ping_ms. Interface specific related performance parameters are out of the scope of this paper and are not considered in the model.

Based on the above-mentioned fields, we select our model features which are listed in Table I. A short description of each feature is also apposed. Features $1-5$ are used directly from the dataset, where $6-7$ are derived. We use time_utc to derive the hour of the day (hour) and add a weekend indicator (weekend, 1 if the measurement conducted in the weekend, 0 otherwise) to investigate temporal effects. Overall, we focus on network related features that are available in every crowdsourced dataset.

**Dataset Statistics:** The total number of samples in the *RTR Open Data* between 2013 and the first two months of 2017 is 4.1 millions. In this paper, we use a part of the dataset corresponding to the most recent 6 months of measurements (September 2016 - February 2017). During this period, we observe an average of $22,662$ samples per month in LTE, among which $14,050$ samples are collected from Android devices. There are 22 distinct SIM networks, including native operators, i.e. MNOs who manage their own infrastructure, virtual operators, i.e. Mobile Virtual Network Operatorss (MVNOs) who rely on others' infrastructure to operate via national roaming agreements, and operators of other countries who are roaming internationally in Austria. Measurements are collected from 628 different device models.

**Exploration and Filtering:** As mentioned before, crowdsourced datasets are noisy and unbalanced due to the voluntary participatory initiation of measurements by end users. In the *RTR Open Data*, for instance, we observe large variations in the number of samples per operator. Furthermore, there is a significant imbalance in the distribution of devices per operator (potentially due to joint smartphone and subscription deals), which implies that higher category devices might pull the average data rate up for a given operator. To overcome this bias, we first select a representative distribution of device categories. Namely, we select LTE Category 4, Category 6, Category 9, and Category 12-13 to label our samples. We then randomly select equal number of samples from each operator per device category. We account only for Android devices with LTE support.

TABLE II: Original and balanced dataset used for Austria's top 3 MNOs (3 AT, A1, T-Mobile (TMA)) in LTE on Android platform (only native) collected during September 2016 - January 2017.

| | A1 | TMA | 3AT | Total |
|---|---|---|---|---|
| Original Dataset | $15,795$ | $19,210$ | $13,834$ | $48,839$ |
| Balanced Dataset | $4,765$ | $4,765$ | $4,765$ | $14,295$ |

We select a subset of the dataset described in Section III. This dataset has a balanced distribution where each operator has equal number of samples (see Table II). In Figure 1, we illustrate the characteristics of the balanced dataset in terms of network related parameters such as latency, DL and UL data rate, and LTE signal strength (RSRP and RSRQ). We use violin plots to show the range of each parameter for each operator. An important observation from this figure is that the range and the density of the parameters do not vary greatly from one operator to another, making it hard to find a single parameter that clearly differentiates operators. In addition, defining thresholds on statistical descriptors of the parameter distributions does not suffice to capture the interplay of all these metrics. Hence, they fall short in contrasting operator behavior across multiple dimensions. This motivates our decision to consider all available parameters and leverage machine learning to build an operator classifier.

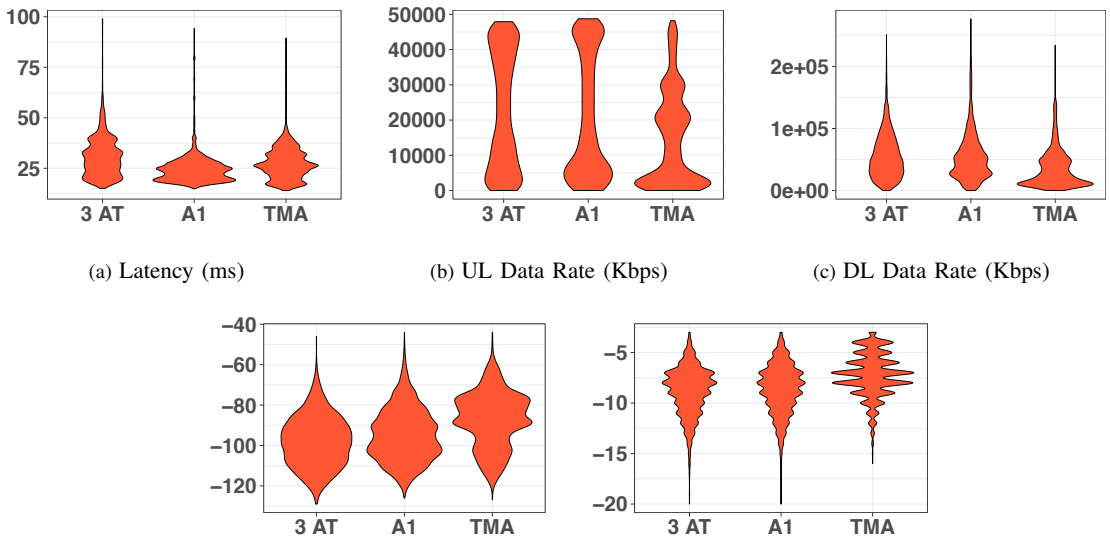
## III. ORCA: Operator Classifier

In this section, we expand on the design and methodology behind ORCA. The flowchart depicted in Figure 2 contains three main building blocks: Dataset, Study Design and Decision Tree Induction.

### A. Study Design

The learning process is designed with a methodology that uses sequentially training, validation, and testing. In particular, we first train and tweak the classification model (training and validation) and then we measure how it behaves on an independent never-before-seen dataset (testing). This approach implies splitting the data into *known data*, which we use for training and validation, and *unknown data*, also known as *hold-out test data*. For a robust evaluation we perform K-Fold cross-validation for the training and validation phase. Cross-validation splits the known data into disjoint training and validation sub-sets, in order to estimate the average accuracy of the model. To further improve the performance of the decision model we evaluate a set of error measures during the cross-validation. We repeat the training and validation for 10 splits of the known data. In the following we explain in detail the data structure and error metrics we used for ORCA.

**Data Structure:** When splitting the dataset into training, validation and hold-out test data, we need to ensure that the three datasets are perfectly disjoint. We first isolate 5 months of the data (September 2016 - January 2017) to create the training dataset. For the hold-out dataset we use one month of




(a) Latency (ms) (b) UL Data Rate (Kbps) (c) DL Data Rate (Kbps)

(d) RSRP (dBm) (e) RSRQ (dB)

Fig. 1: Characteristics of the Training Dataset.

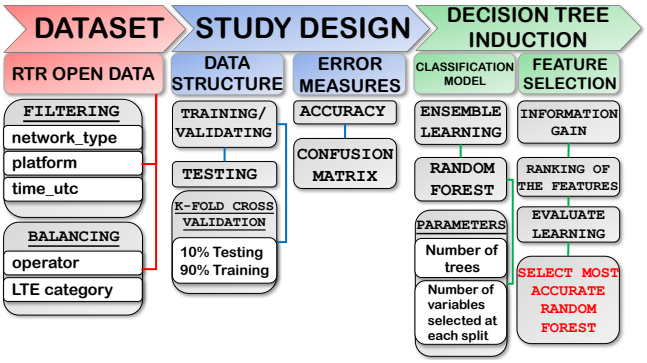



Fig. 2: ORCA flowchart.

the measurement data (February 2017). We then filter the data to ensure a balanced distribution (Table II). With 10-fold cross validation, we train our algorithm and evaluate its performance under 10 different splits of the training data. In this case, the training dataset has 90% of the total learning dataset, and the validation set, the 10% left. At each repetition (i.e., fold), we do not reuse the 10% of training data we use for validation.

**Error Measures:** To determine the model with the best results, we evaluate a set of different error measures. The *accuracy* of a classifier is, by definition, the percentage of instances that are classified correctly when a set of unknown data is tested. Accuracy is commonly used as a performance metric in binary classification problems. However, in a multi-classification problem, accuracy is not enough to reflect the performance and the efficiency of the classifier. To further evaluate the performance of the model, we build a confusion matrix that maps predicted operator labels on the rows direction with the real ground-truth operator labels on the column directions.

### *B. Decision Tree Induction*

In this section, we discuss the learning method we select and the steps we take to select the most important features for building the decision model.

**The Random Forests Learning Method:** We derive and fine-tune a decision tree model based on the popular machine learning method of Classification and Regression Trees (CART) [6]. Tree-based learning methods rely on iteratively partitioning the data into smaller groups of similar elements [12]. The key idea is to chose the splits which maximize the group homogeneity (i.e., how similar are the elements within the same group), or until the small groups are sufficiently *pure*. Choosing the right number of splits is a challenge, since the model can easily overfit by considering splits that are very specific to the training data, or, contrarily, underfit it by considering shallow general splits. Finding the correct balance is conditioned by finding the optimal set of features used to partition the data.

The next step is to adopt *ensemble learning*, that is, generate many classifiers and aggregate their results. For this purpose the Random Forests [7] algorithm is selected. The used bagging approach builds independent decision trees (i.e., with each new tree the algorithm growth does not depend on the trees grown earlier) using a bootstrap sample of the training dataset. In the end, a simple majority vote is taken for finding a prediction. Random Forests add randomness to the bagging approach. In Random Forests, each split uses a subset of features randomly chosen at each repetition. This algorithm is known to outperform many other algorithms, including discriminant analysis, support vector machines and neural networks, and is robust against overfitting [7].

The number of trees in Random Forests is an important

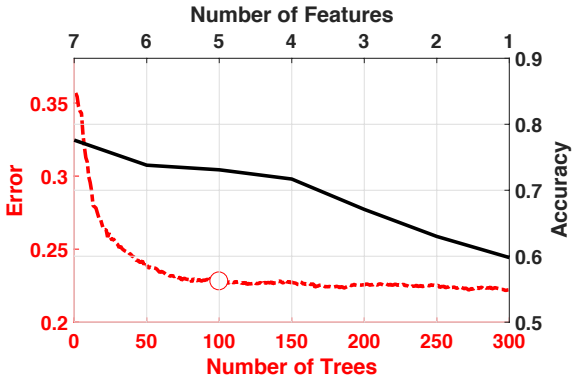
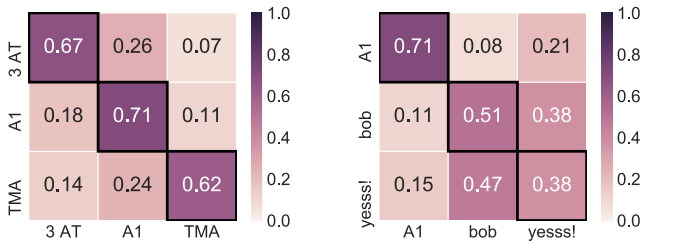



Fig. 3: (i) **Left and bottom axes -dotted line-** Classifier error as a function of the number of trees in Random Forests. Based on these results, we select to use 100 decision trees in the forest. (ii) **Top and right axes -solid line-** The impact of total number of features used for model building on the accuracy of the classifier.




(a) Native MNO scenario.

(b) MVNOs compared to their native operator.

Fig. 4: Confusion matrices illustrated by heatmaps. The gradient encodes the accuracy for each block of the confusion matrix. Rows and column direction indicate the groundtruth and the predicted operator respectively.

parameter that dictates the performance and the computational complexity of the classifier. We need to select a number of trees that provides a good compromise between the accuracy of the classifier, the computational complexity, and the probability of overfitting to the training dataset [15]. We vary the number of trees between 2 and 300 and evaluate our classifiers using 10-fold cross-validation on the training dataset. Figure 3 (Left and bottom axes -dotted line-) presents the Out-of-bag (OOB) error (i.e. the mean error) of the classifier against the number of trees. Based on these results, we select a forest that consists of 100 decision trees. Another important parameter that dictates the performance of Random Forests is the maximum number of features that are randomly sampled in each individual tree. We set this parameter to its default value which can be defined as the square root of the total number of features.

**Feature Selection:** We proceed with feature selection and the metric of our choice is Information Gain (IG). IG (also called *entropy*) is a widely accepted method [10] for evaluating the contribution of a feature in distinguishing between instances of different classes. It varies between 0 and 1 with the latter one representing maximum information. We use a ranking approach to sort the features based on the scoring assigned by IG. The decreasing order of the features and the associated IG value is being reflected on Table I. We observe that latency dominates the pool of features in the dataset along with upload and download data rate following close behind. In the contrary, entropy of `weekend` is close to zero which means that it hardly provides useful information to the classifier.

In order to select the subset of features that ensures the optimal performance of Random Forests, we adopt a progressive approach. We first train Random Forests with all the available features using 10-fold cross-validation and estimate its performance by generating the confusion matrix. Subsequently, we eliminate the feature with the lowest IG (i.e., weekend), retrain the model and calculate yet again the confusion matrix. With this approach, we further iterate through all the remaining features in an increasing order of IG and, at each iteration, the feature with the lowest contribution is eliminated. This is invaluable to the main goal of this paper, that is to estimate whether a subset of the selected features confuse Random Forest instead of helping the algorithm to produce a higher-performance classifier. Figure 3 (Top and right axes -solid line-) depicts the overall accuracy of each classifier against the number of features used according to the predefined approach. We observe that the generated classifier with all seven features is the best performing one.

## IV. Performance Evaluation

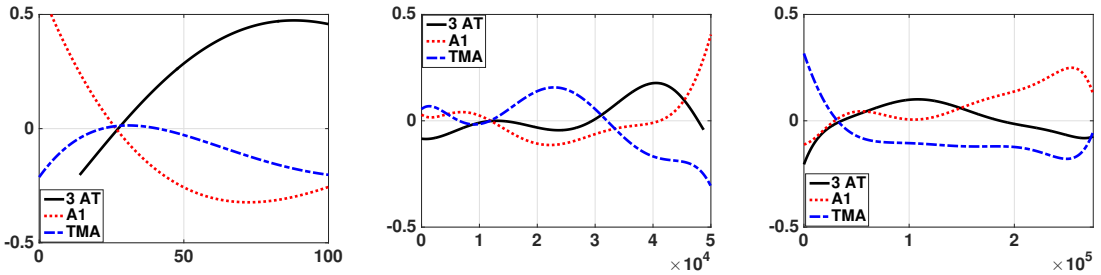
Describing the performance of a multi-classification algorithm with a single number is often not enough to describe the overall behavior of the classifier. Next, we evaluate the performance of ORCA by leveraging visualization tools such as heatmaps to illustrate better the confusion matrices.

### A. Classification of Native Operators

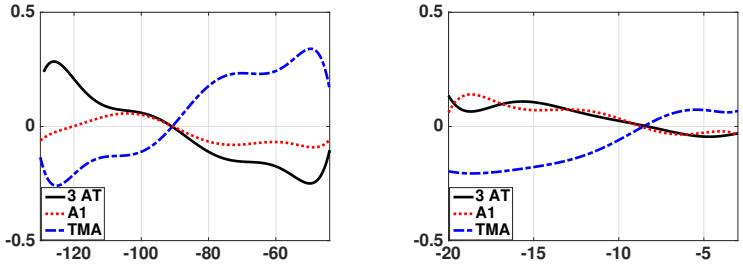
For the classification of native operators (i.e. no roaming case), we use the balanced training dataset described in Section II (September 2016 - January 2017) and evaluate the performance of ORCA on the hold-out test dataset (February 2017) which consists of 3840 samples. Figure 4a shows the heatmap of the confusion matrix where the correctly classified instances are located in the main diagonal of the matrix.

We observe that ORCA can identify MNOs with an accuracy of 67%, 71% and 62% respectively. This is a rather good result considering that a random guess has a 33% probability to be successful. Notice that, there exists a slight confusion between 3 AT - A1 and TMA - A1 while A1 is equally likely to be confused with either one. It is important to point out that it was difficult to differentiate the operators by using statistical representations (violin or box plots) of single features. However, with ORCA, the operators can be classified quite accurately indicating that each operator has a certain pattern that is uniquely identifiable.

To understand the contribution of each feature to the classification performance of ORCA, we use the *forest floor* approach




(a) Latency (ms) (b) Upload Data Rate (Kbps) (c) Download Data Rate (Kbps)




(d) RSRP (dBm) (e) RSRQ (dB)

Fig. 5: Feature contributions for the training dataset, for each feature and for each operator. Y- axis depicts the change in the predicted probability. X-axis is the value range for each feature. We fit polynomial curves to ease readability of the plots.

described in [16]. Recall that, by using a pluralistic vote mechanism, Random Forests serves a probabilistic prediction for each class. The connections between samples are described by the change in the predicted probability for each operator and they add to zero. Therefore, feature contributions can be defined as the sum of these changes over trees for each sample. Figure 5 depicts the change in the predicted probability versus the value range of each feature, per operator. Sub-plots corresponding to the most important 5 features are sorted in decreasing order of importance according to the IG. We notice that observations with latency higher than $30ms$ are more likely to be classified as 3 AT. Moreover, RSRP higher than $-90dBm$ imply a higher possibility to be labeled as TMA. In addition, we observe that UL and DL data rate can identify the correct operator with an adequate likelihood within different intervals. In summary, operators have different likelihood for the value range of each feature and ORCA exploits this to differentiate the operators.

### B. Contrasting MNOs Using a single Feature

In this subsection, we assume there is only one available feature in the dataset and investigate what the dominant feature is for contrasting the operators network. Figure 6 depicts the accuracy per MNO for 5 different features that are ordered according to the IG ranking presented in Section III. The foreground bars illustrate the accuracy when only a single feature is used and the background bars represent the accuracy when all seven features are available and used. We observe that for the RTR dataset, we can identify MNOs with an average accuracy of 60% by using only latency. Note that this is only 7% lower than the native MNO scenario and clearly indicates

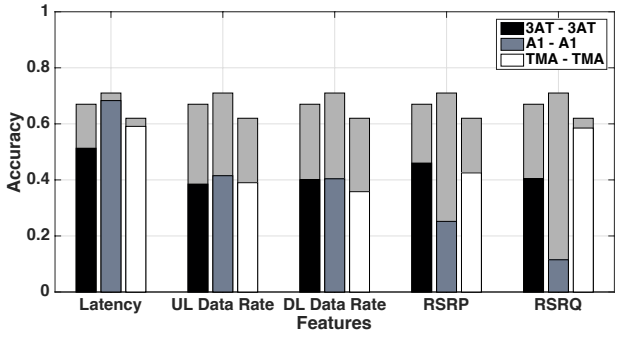



Fig. 6: Accuracy per MNO using a single feature. The background bars indicate the accuracy of the native MNO scenario.

that the latency is the most important feature contributing greatly to differentiating operators. Furthermore, we observe that DL and UL data rate are not good for differentiating operators, providing a 38% accuracy, which is slightly better than choosing a mobile operator randomly. This shows that DL and UL data rates are very similar for all operators. The same results apply for the LTE power-related parameters. However, the distribution of accuracy between the MNOs is different. RSRP identifies 3 AT and TMA with an average accuracy of 44%, while for A1, this number goes down to 25%. Finally, while RSRQ is a good identifier for TMA, it totally fails when it comes to A1, where accuracy reaches 11%. In summary, our results show that latency is the most important feature for operator classification, however, using all available features clearly increases the accuracy of operator classifier.

### C. Can ORCA Identify MVNOs?

We further investigate the roaming scenario, where a national agreement between MVNOs and a native operator exists. MVNOs are allowed to utilise the physical infrastructure of their service provider while applying different tariffs or QoS to their customers. Our aim is to understand if the customers of both sides are "treated" in the same manner. We use the same datasets in time, and consider A1 and two of its MVNOs: bob and yesss!. Figure 4b shows the heatmap of the confusion matrix for A1, bob and yesss!. We observe a clear differentiation between A1 and the MVNOs, most likely due to resource constraints enforced by A1 to the MVNOs. Moreover, *upload_kbit* is among the features with the largest entropy according to the IG analysis, which means that MVNO customers are not allowed to exploit all the available bandwidth when uploading. On the other hand, ORCA suffers more when it comes to bob and yesss!, showing an accuracy of 49% and 40% respectively. This indicates that these MVNOs are treated similarly and ORCA cannot distinguish any differences between them.

## V. Conclusions and Future Work

In this paper we introduced *ORCA: Operator Classifier* for identifying patterns and disclosing exclusive aspects of MNOs using noisy crowdsourced datasets. As opposed to the majority of existing works in literature, which focus on operator performance with respect to a single parameter, ORCA leverages the power of the rich set of features commonly found in crowdsourced datasets and jointly considers multiple features. It includes a fundamental pre-processing and filtering step where the data is de-noised and re-sampled, and uses a learning process to build a classifier for identifying operators. Random Forests is used for capturing this characterization. Our results show that latency is the most important feature to differentiate MNOs behavior. However, using all available features clearly increases the accuracy of operator classifier. Furthermore, virtual operators are treated differently compared to their native operator while they have similar behaviour among themselves and are not easily detectable.

The basic framework of ORCA is relevant for many potential applications that could be explored as part of our future work. For example, improving the robustness of ORCA would allow a more accurate match between pricing and claimed offered QoS, hints regarding the different CN and RAN structures, detection of performance bottlenecks and network problems. Part of our future work, we also plan to carry on further analysis on benchmarking of different classification algorithms and exploration of similar crowdsourced datasets.

## References


[1] https://opensignal.com.
[2] https://sites.google.com/site/mobiperfdev/.
[3] https://speedtest.net.
[4] https://www.netztest.at/en/.
[5] http://www.speedtest.net/awards.
[6] Breiman et al. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.
[7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
[8] Cainey et al. Modelling download throughput of LTE networks. *IEEE LCN Workshops*, 2014.
[9] Chen et al. Understanding the Complexity of 3G UMTS Network Performance. In *IFIP Networking Conference*, 2013.
[10] R. L. De Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine learning*, 6(1):81–92, 1991.
[11] Ferlin et al. Measuring the QoS characteristics of operational 3g mobile broadband networks. In *WAINA*, pages 753–758. IEEE, 2014.
[12] J. Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.
[13] Huang et al. An In-depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance. In *Proc. of SIGCOMM*, 2013.
[14] Midoglu et al. Opportunities and challenges of using crowdsourced measurements for mobile network benchmarking a case study on RTR Open Data. *SAI Computing*, pages 996 – 1005, 2016.
[15] Oshiro et al. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.
[16] Welling et al. Forest floor visualizations of random forests. *arXiv preprint:1605.09196*, 2016.