Estimating Downlink Throughput from End-User Measurements in Mobile Broadband Networks

Konstantinos Kousias*, Özgü Alay†, Antonios Argyriou‡, Andra Lutu§ and Michael Riegler†
*Simula Research Laboratory, Oslo, Norway
†Simula Metropolitan Center of Digital Engineering, Oslo, Norway
‡University of Thessaly, Volos, Greece
§Telefonica Research, Barcelona, Spain

***Abstract*—In recent years, *Downlink* (DL) throughput estimation in *Mobile Broadband* (MBB) networks has gained immense popularity and it is expected to become a vital component of the upcoming *fifth generation* (5G) systems. Plentiful adaptive video streaming algorithms greatly rely on accurate DL throughput predictions to adapt their mechanisms and ensure high *Quality of Service* (QoS) to the end-users. Thus far, conventional DL throughput estimation approaches, also known as *speed tests*, require an extensive exchange of TCP traffic over the network for an allocated time duration. While such tools appear to deliver trustworthy results, they turn out to be inefficient when mobile subscriptions with limited data plans are engaged.**

In this paper, we propose a supervised *Machine Learning* (ML) solution for DL throughput estimation that aims at delivering highly accurate predictions while significantly limiting the over-the-air data consumption. We capture the network performance metrics by exploring both crowdsourced and controlled testing methodologies. We leverage `RTR-NetTest`, a platform of broadband measurements provided by the *Austrian Regulatory Authority for Broadcasting and Telecommunications* (RTR), and `MONROE-NetTest`, its counterpart wrapper built as an *Experiment as a Service* (EaaS) on top of *Measuring Mobile Broadband Networks in Europe* (MONROE). Results reveal that our solution can achieve a 39.7% reduction in terms of data consumption while delivering a *Median Absolute Percentage Error* (MdAPE) of 5.55%. We further show that accuracy can be traded-off, for example, a significant data consumption reduction of 95.15% can be achieved for a MdAPE of 20%.

***Index Terms*—Downlink (DL) Throughput Estimation, Mobile Broadband (MBB) Networks, Machine Learning (ML), Multiple Linear Regression (MLR), Support Vector Regression (SVR), Random Forests (RF)**

I. Introduction

Mobile Broadband (MBB) networks underpin numerous vital operations of the modern society and are arguably becoming the most important piece of the modern communications infrastructure in the world. The use of 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 networks. According to the Cisco Global Mobile Data Traffic Forecast[1], there will be 11.6 billion mobile-connected devices by 2021, exceeding the world's projected population at that time. Moreover, by 2021, the emerging fifth generation (5G) mobile connections will generate 4.7 times more traffic than the average fourth generation (4G) connections. Among all mobile content types, more than three-fourths of the world's mobile data traffic will be video. These facts motivate researchers to further enhance the capabilities of MBB networks to cater for plethora of new applications and services (e.g. user-generated live streaming services, IP telephony, video chat applications, etc.) and a wide variety of mobile devices. Therefore, it is essential to better understand the fundamental characteristics of MBB networks and how to characterize network performance. This is crucial not only for improving the user's experience for the services that are running on the current MBB infrastructure, but also for providing feedback to the design of the upcoming 5G technologies.

In recent years, the term *mobile speed* (commonly represented by Downlink (DL) throughput) has become synonymous with the performance of Mobile Network Operators (MNOs). A good estimate of the DL throughput is essential for myriad tasks that include network performance optimization, traffic management, application provisioning and crowdsourced reporting. Service providers, whose business depends on MBB networks, can use these estimates to optimize the performance of their services. Application developers can also exploit these results to adapt their applications towards enhancing Quality of Service (QoS). For example, Youtube and Netflix heavily depend on bandwidth estimation techniques for video rate adaptation. They leverage accurate bandwidth predictions and video performance models to provide a better service to their subscribers. Therefore, it is of utmost importance to reveal the key factors that are required to accurately characterize and predict the mobile speed in MBB networks.

Current crowdsourced speed test tools provide estimates of throughput by initiating parallel TCP connections over the network for an allocated time duration. When no data limitations exist, these approaches seem to work sufficiently well, delivering reliable results. However, when mobile subscriptions with limited data plans are involved, a solution to restrain data usage is imperative.

In this paper, we target to characterize DL throughput and build an empirical supervised model that exploits the potential of Machine Learning (ML), provides highly accurate predictions and reduces the over-the-air data consumption. The main research questions that we answer in this paper

[1]Cisco Visual Networking Index: Global Mobile Data Forecast Update, 2016-2021, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

are the following: What is the minimum set of network features that can characterize DL throughput accurately? Is the metadata information we extract from the end devices adequate to capture and predict DL throughput? If not, which are the additional parameters that we need to measure? Can we extract these parameters via passive measurements, or active ones are required? And how much data from active monitoring do we need to increase DL throughput estimation accuracy? Considering that there may be a leeway regarding the accuracy of the DL throughput estimate, we investigate scenarios where accuracy can be traded for a lower amount of data consumption. For example, a video streaming application might be fully operational with a coarse estimate of the end-to-end DL throughput, while high accuracy is usually desired for monitoring performance.

We answer these questions by leveraging the power of ML to extract information from a data-rich environment while analyzing large amounts of controlled and crowdsourced measurements from operational MBB networks. Assuming a non-cooperative environment without access to network-side information, we present a supervised ML solution that captures the relationship between network performance and the underlying factors that affect it, as perceived from the end-user plane.

Our main contributions in this paper are twofold:

1) We introduce a feature engineering process that aims to derive features from raw parameters towards enhancing the performance of ML models.
2) We propose a supervised ML approach that leverages historical network data to estimate DL throughput with high accuracy while reducing excessive amounts of active measurements. We show that by leveraging our approach we can achieve a 39.7% reduction in the amount of data an end-user would normally transfer to actively measure DL throughput with a conventional speed test tool and only minimally reduce the measurement accuracy, providing a Median Absolute Percentage Error (MdAPE) of 5.55%. Moreover, we show that we can trade-off accuracy, depending on the application's requirements, where a significant 95.15% data consumption can be achieved for a MdAPE of 20%.

II. Experimental Setup

There exists a plethora of methodologies adopted to capture performance metrics for MBB networks that can be clustered in two categories. The first is *crowdsourced testing*, where the assistance of a group of people is required to extract network related characteristics. The second is *controlled testing*, where experimentation takes place in a more discipline manner where less factors can influence the outcome (e.g. hardware testbeds, drive tests with benchmarking equipment, etc.).

A. NetTest Measurement Methodology

In this paper, we rely on the NetTest measurement methodology to collect data from operational MBB networks. First, we retrieve the measurements results from running NetTest with a crowdsourced approach (as offered by RTR, see Section II-B). Then, we retrieve similar results from running NetTest with a controlled approach (as offered by Measuring Mobile Broadband Networks in Europe (MONROE), see Section II-C). We highlight the six phases that a NetTest speed test undertakes in the following order: I) *Initialization*, II) *Pre-Test Download*, III) *Ping Test*, IV) *Download Test*, V) *Pre-Test Upload* and VI) *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 with the measurement server. The above exchange is very brief and consists of an almost constant number of packets. Once the connection is established, *Pre-Test Download* begins. Both of the *Pre-Test* phases are undertaken to ensure that the connection is in an *active state*. During *Pre-Test Download*, the client requests and the server sends a data packet in each active thread. While the duration of the phase has not exceeded its nominal value, the client requests a data block of double the size compared to the last iteration step. The transfer of the last data block will be completed even if the duration has already exceeded its nominal value. *Pre-Test Upload* works analogously to *Pre-Test Download*, but with the client as the sender and the server as the receiver.

Ping Test consists of the client sending a fixed number of TCP pings to the server to test the latency of the connection. *Download Test* and *Upload Test* are the last components of a speed test, where multiple TCP threads are opened, and within each of these, the receiver requests and the sender sends data streams that consist of fixed size packets. After the nominal duration, the sender stops sending packets on all connections. The last packet for each thread is transmitted at its entirety where DL and Uplink (UL) throughput are estimated. The default duration value as provided by RTR is set to 7000 ms while in MONROE it acts as a hyperparameter with its default value set to 10000 ms. To maintain consistency between the two platforms, we alter the duration value of the latter to match the formers' one.

B. RTR Crowdsourced Testing

The elemental notion behind crowdsourcing is to encourage people to utilize their end devices and run specially designed application tests, widely known as *speed tests*, to generate and collect explicit information regarding their connection. From end-users' perspective, speed tests come at a benefit, thus justifying the (sometimes costly) data consumption. For example, end-users find speed tests convenient when assessing the performance of their own network connection in terms of throughput and latency. Examples of crowdsourced platforms include Speedtest[2], OpenSignal[3], MobiPerf[4] and RTR-NetTest[5]. Among the available tools, RTR-NetTest is the only

[2]www.speedtest.net
[3]www.opensignal.com
[4]www.sites.google.com/site/mobiperfdev
[5]www.netztest.at

one that openly provides via its webpage the source code together with an open dataset, called *RTR Open Data*.

Data Access. The results of RTR-NetTest are available as CSV and JSON files. The CSV files are organized in monthly bins, and can be downloaded as zipped archives. The JSON files are linked to single measurements, hold a wider space of features and can be fetched by using a *RESTful API*[6].

C. MONROE Controlled Testing

Unlike crowdsourcing, experimentation in a controlled environment implies that certain aspects of the experimental setup can be controlled to eliminate certain bias. An example of such testbed is MONROE, the first open access, European-scale platform that allows for research on operational MBB networks [1], [2]. It comprises of 450 stationary and mobile nodes widely distributed across Norway, Sweden, Italy and Spain while it is continuously expanding. The architecture of a MONROE node includes an Accelerated Processing Unit with AMD 1GHz dual core 64 bit processor and 4GB DRAM. Nodes are multi-homed to three MBB networks using commercial grade subscriptions. The three 3G/4G MC7455 miniPCI modems support LTE Category 6 while WiFi connectivity is available through a built-in dual band AC WiFi card. The operating system is based on a Debian Linux/GNU distribution. A GPS system is utilized for tracking their location in the case of mobile nodes. To ensure reliability and provide agile reconfiguration, MONROE runs experiments inside *Docker* containers. The authors in [3] package RTR-NetTest as an Experiment as a Service (EaaS) for MONROE (under the MONROE-NetTest tag). We will use the measurement results of MONROE-NetTest which by design allow for compatibility at the methodology of the RTR-NetTest crowd campaign.

Data Access. MONROE-NetTest runs in MONROE as a Docker container[7] and produces four output JSON files covering the result fields, such as DL and UL throughput and the median TCP payload Round Trip Time (RTT), raw time series for each TCP flow, detailed TCP statistics and traceroute information. We access the openly available result files that are stored in the MONROE private database.

III. Datasets

In this section we describe in detail the NetTest raw parameters and the derived model features while we unveil statistical properties regarding the two datasets.

A. Statistical Properties

RTR-NetTest consists of 5038 observations collected between January '18 and October '18 (10 months timespan). MONROE-NetTest consists of 3174 observations collected from 6 MONROE testing nodes between October '17 and February '18 (5 months timespan). We split both datasets in training data (67%) and testing data (33%) by using systematic sampling to ensure that they are drawn from the same distribution. In ML, a good model should be able to capture the intrinsic information of the training data but also perform well on the never seen testing data, which basically means that it is generalizable. We remove invalid or false measurements that can affect the reliability of our results (e.g. samples with negative values for throughput or latency, Reference Signal Received Power (RSRP) or Reference Signal Received Quality (RSRQ) values outside the licit range, etc.) However, we do not treat extreme but rational values as outliers since they can explain signs of variation in the data.

B. Features

NetTest[8] parameters are divided in six clusters: test, location, device, network, coverage and performance. The CSV files consist of 39 parameters, while additional attributes bring the number up to 76 for the JSON files. Such examples include roaming information, amount of data volume transferred per interface, an extensive list of location related features, and many more[9]. The most high-profile parameter is called speed curve (`speed_curve`). It is a compound parameter and it is only available in the JSON files. A speed curve covers the historical *per-thread* time series evolution of the *Ping Test*, *Download Test* and *Upload Test*. However, and only for RTR-NetTest, an aggregated (between the available threads) estimation is returned instead due to several cases of missing data. As a result, for MONROE-NetTest, we average between the threads as a pre-processing step. Each sequence is unevenly spaced with time intervals that lie between 50 ms and 100 ms in average. The total number of bytes successfully transferred after the nominal period of a *Download Test* is represented as `downlink`. Upon conversion, `downlink` can be translated to the link capacity.

We split the space of attributes in two tables. Table I outlines the NetTest raw parameters while Table II the NetTest derived model features. We provide a short description alongside.

We filter the data to account only for Android devices (`platform`) with LTE connectivity (`network_type`, e.g. UMTS, GSM, LTE, etc. and `cat_technology`, e.g. 3G, 4G, etc.) and no national or international roaming (`roaming_type`). In addition, we filter by geo location source (`loc_src`), NAT/IP-version (`nat_type`) and test server (`server_name`, namely *RTR https 10G AT*) to eliminate unwanted bias. The number of active TCP threads (`num_threads`) is three for RTR-NetTest and five for MONROE-NetTest. The latter can be configured (if necessary) during the docker deployment phase.

The LTE category (`lte_cat`) characterizes an end device in terms of network capabilities, thus, we include it as a categorical feature. In MONROE, however, due to the same modems used in all the nodes, `lte_cat` is the same for all

[6]During an HTTP call, a unique test identifier (*open-test-uuid*) is required. An example of a `GET` request is as follows: data.netztest.at/opendata/opentests/⟨open-test-uuid⟩?⟨parameters⟩

[7]www.github.com/MONROE-PROJECT/Experiments/tree/master/experiments/nettest

[8]We refer to *NetTest* when the information following is common for both platforms except implied otherwise.

[9]We refer the reader to the *RTR-NetTest Open Data Interface Documentation* for a complete list of available parameters and their description.

TABLE I: NetTest raw parameters alongside a short description.

ID	Raw Parameter	Short Description
1	`speed_curve`	Historical evolution of speed test
2	`network_type`	Type of the network e.g. GSM, LTE
3	`cat_technology`	Technology category e.g. 3G, 4G
4	`platform`	Operating system
5	`roaming_type`	Roaming status
6	`model`	Device brand
7	`network_mcc_mnc`	Home network identification code
8	`sim_mcc_mnc`	Access network identification code
9	`num_threads`	Number of threads
10	`loc_src`	Source for the geo location-data
11	`nat_type`	Type of connection
12	`server_name`	Name of the test server
13	`land_cover`	Classification of the land cover
14	`lat`	Latitude
15	`long`	Longitude
16	`time_utc`	UTC date and time

TABLE II: NetTest model features alongside a short description. Features in bold are only available for RTR-NetTest. We represent the scalar response in *Italics*. The horizontal lines split the pool of features in three clusters based on how *pricey* they are to collect in terms of data consumption.

ID	Model Feature	Short Description
1	`ping_avg`	Latency average (ms)
2	`ping_std`	Latency standard deviation
3	`rsrp_avg`	Signal strength average (dBm)
4	`rsrq_avg`	Signal quality average (dB)
5	`time_of_day`	Hour of the day
6	`weekend`	Weekend indicator
7	`network`	MNO
8	**`move`**	**Distance covered (Km)**
9	**`lte_cat`**	**LTE category**
10	`UL_beta0`	Intercept of the UL regression line
11	`UL_beta1`	Slope of the UL regression line
12	`DL_beta0`	Intercept of the DL regression line
13	`DL_beta1`	Slope of the DL regression line
14	*`downlink`*	DL transferred bytes

measurements, and hence, it is not considered as a feature. At the time of writing, there are 19 available LTE categories, and from end-user's perspective, the primary difference lies on mobile speed. Consequently, higher category devices can pull the average throughput up and bias the reported results. Within the scope of this paper, we select the five most popular mobile devices, corresponding to two LTE categories that we extract by using the device brand (`model`). An alternative way would be to include `model` as feature, but we choose to use the category in order to focus more on network related parameters rather than individual brands. Table III summarizes the mobile brands, DL/UL throughput values and LTE categories. It should be noted that these are only theoretical speeds that can only be observed under ideal conditions (e.g. minimum interference, transmissions carried out in immediate proximity of the eNodeB, etc.).

RSRP (`rsrp_avg`) is the power of a reference received signal at the end-user side. It ranges between -140dBm and -44dBm. Likewise, RSRQ (`rsrq_avg`) is a metric of the wireless channel quality, that is the ratio of RSRP versus the total received signal power, including interference and noise, and it ranges between -19.5dB and -3dB. Both of the

TABLE III: List of mobile brands alongside their theoretical speeds (in Mbps) and LTE category. Order is based on popularity.

Mobile brand	DL	UL	LTE category
`Samsung S8 (SM-G950F)`	979	150.8	16/13
`Samsung Galaxy S7`	603	150.8	12/13
`Samsung SM-N950F`	979	150.8	16/13
`Samsung SM-G955F`	979	150.8	16/13
`LG G5`	603	150.8	12/13

features are not present in the MONROE-NetTest reported results. Instead, we collect them from the MONROE metadata with node ID, Integrated Circuit Card Identifier (ICCID) and timestamp treated as primary keys.

To account for temporal effects, we derive the hour of day (`time_of_day`) and a weekend indicator (`weekend`, 1 if the measurement took place during the weekend, 0 otherwise) from the absolute timestamp (`time_utc`, in POSIX time).

MNOs are in charge of network configurations, spatial deployment of eNodeBs and traffic policies applied over LTE. Furthermore, they regulate the QoS provided to their customers and they settle the different tariffs and subscription plans. We derive `network`, a three-level categorical feature that stands as an identifier for each MNO (i.e. *A1*, *T-Mobile A* and *3 AT* for RTR-NetTest, and *Telenor*, *Telia* and *ICE* for MONROE-NetTest). We only consider MNOs that own the physical network infrastructure by using a combination of `sim_mcc_mnc` and `network_mcc_mnc` which indicate the Mobile Country Code and Mobile Network Code as read from the SIM card (i.e. home network), and the network that is currently in use (i.e. access network), respectively.

The end device geographical position is determined with a combination of latitude (`lat`) and longitude (`long`) coordinates. Both parameters are available as time series attributes and we leverage them to estimate the distance covered throughout a speed test, denoted as `move` and measured in Km. In MONROE, only stationary nodes are considered, therefore, `move` is not used as a feature. We must point out that `move` declares the distance covered throughout the complete NetTest experiment (all six stages). Since the duration of the speed test is known, it can be interpreted to the end-user's speed rate. To further remove any location related bias, we consider the most prominent land cover areas (`land_cover`) in both countries (Austria and Norway).

In addition, a speed curve features the results of a *Ping Test*. As we mentioned earlier, a *Ping Test* takes place prior to the *Download Test* and *Upload Test* and consists of 10 (in most cases) individual measurements. We represent latency by average (`ping_avg`) and standard deviation (`ping_std`) statistics.

IV. Feature Engineering

One of the main contributions of this paper is the feature engineering methodology. We split this process in two phases. First, we introduce a linear interpolation method to guarantee that all speed tests follow a consistent time series structure. Second, we propose a curve fitting scheme to *compress* the historical data information and consequently cut down the data size and help fight overfitting during the training phase. Feature engineering aims to reshape and prepare the datasets in an appropriate manner before we move to the ML territory.

A. Linear Interpolation

Since the speed curve granularity can slightly differ between samples or threads, we apply an interpolation method to sub-sample the sequences in equal spaced *segments* and preserve consistency across the two datasets.

In Listing 1, we illustrate the interpolation algorithm using a pseudocode. The input parameters comprise of the dataset, a level of granularity (70 ms) and the number of segments that each sequence is split (100)[10]. A high level of granularity may end up as an issue due to missing data, while a low level of granularity may hurt the model accuracy. Therefore, its selection needs to be a compromise between the two. At every repetition, we compute the two interpolated coordinates, and the outcome of the linear interpolation formula yields the number of bytes transferred up to that point in time. We repeat the linear interpolation algorithm for UL.

Algorithm 1 Linear Interpolation Algorithm

```
Input: data, g_level, no_segments
Output: DL_v, UL_v
1: for $i = 1 \rightarrow length(\texttt{data})$ do
2:     for $j = 1 \rightarrow$ no_segments do
3:         $x \leftarrow [tDL1, DL1]$ ▷ Coordinates for data point A
4:         $y \leftarrow [tDL2, DL2]$ ▷ Coordinates for data point B
5:         $interval \leftarrow$ g_level $* j$
6:         $interp \leftarrow function(x, y, interval)$
7:         Store the outcome
8:         Repeat for Upload
9:     end for
10: end for
```

B. Curve Fitting with Simple Linear Regression

In Listing 2, we depict the pseudocode that translates each speed curve into coefficients by leveraging a simple linear regression scheme. After the linear interpolation algorithm, each sequence is divided in 100 evenly spaced segments. First, we cut the time dimension in batches of 350 ms (i.e. 5 segments[11]). Then, we fit a simple linear regression model with input data from the first batch to capture the relationship between the response variable (bytes_transferred) and the explanatory variable (time_elapsed). We store the regression coefficients DL_beta0 (intercept) and DL_beta1 (slope) in a new table. At the next iteration, we linearly concatenate the adjacent batch to form a larger curve (i.e. 700 ms). Likewise, we fit a linear regression model with a new pair of regression coefficients. We carry out the same process until the regression line fits the entire sequence. The new table length equals to length(data) $*$ no_segments/no_batches with the remainder of the features in Table II replicated row-wise[12]. We repeat the curve fitting algorithm for UL.

Let n represent the number of points in a batch, then the simple linear regression model is expressed as $Y_i = \beta_1 X_i + \beta_0 + \epsilon_i$, where Y_i, $i \in \mathbb{Z} : i \in [1, n]$ is the bytes_transferred and X_i is the time_elapsed. The regression coefficient (β_1) shows the magnitude of the effect that the explanatory variable has on the response variable given that the remainder of explanatory variables remain constant (if any). The sign signifies whether this effect is positive or negative. Last, β_0 stands for the intercept term while ϵ_i is the prediction error.

Algorithm 2 Curve Fitting Algorithm

```
Input: data, DL_v, UL_v, no_segments, no_batches
Output: Linear Regression coefficients
1: $t \leftarrow$ Split time in $k =$ no_segments equal segments
2: for $i = 1 \rightarrow length(\texttt{data})$ do
3:     for $j = 1 \rightarrow$ no_batches do
4:         $x \leftarrow$ DL_v$[1 : (5 * j)]$
5:         $y \leftarrow t[1 : (5 * j)]$
6:         $DLfit \leftarrow fit(x, y)$      ▷ Fit a regression model
7:         Store the coefficients
8:         Repeat for Upload
9:     end for
10: end for
```

V. Model Design

Next, we provide a high level description of the forecasting algorithms that we use during the DL throughput estimation phase. Moreover, we leverage a feature selection approach to obtain a subset of the NetTest model features and enhance the performance of the ML models.

A. Overview of the Forecasting ML Algorithms

To estimate the number of bytes transferred at each speed test duration scenario, we fit a supervised ML learning model.

1) Multiple Linear Regression (MLR): In statistical modeling, Multiple Linear Regression (MLR) is an elementary but rather efficient technique for capturing the relationship between a response (dependent) variable and two or more explanatory (independent) variables by fitting a line through a multi-dimensional space of data samples. Despite the fact that more complex approaches, such as principal component analysis and neural networks, outperform MLR in terms of accuracy, it delivers a good trade-off between computational complexity and performance [4]. Throughout the years, linear

[10]We calculate the number of segments from dividing the default speed test duration (i.e. 7000 ms) by the level of granularity.

[11]A basic assumption of a simple linear regression model is that at least two data points are present, while to observe evidence of departure from linearity three data points are required. Although we set the level of granularity to 70 ms to avoid dropping sequences with missing data, such cases may still exist. Therefore, we select batches larger than three segments.

[12]Besides the regression coefficients, distance covered (move) also changes across different batches (in case of mobility). Unfortunately, the available time series data for lat and long are low granular, therefore, we make the assumption that move has identical values for the entirety of scenarios.

regression analysis has found application in a wide variety of fields including business, economics and medicine [5]–[7].

To approximate the optimal solution, MLR leverages the linear least squares fitting approach that minimizes the sum of squares between the predicted and the ground-truth data [8]. The formal mathematical expression of a MLR model is a generalized form of the Equation in Section IV-B. Let f represent the number of available explanatory variables, then b_1 and X_i can be altered with b_i and $X_{j,i}$ respectively, where $j \in \mathbb{Z} : j \in [1, f]$ and $i \in \mathbb{Z} : i \in [1, n]$.

2) Support Vector Regression (SVR): The concept behind Support Vector Regression (SVR) is similar to this of Support Vector Machines but instead of classes, the output consists of real value predictions [9]. A prime advantage of SVR is that the output model does not depend on the distribution of the underlying data, unlike MLR, but instead relies on kernel functions. A kernel function transforms the data from a non-linear space to a linear space and makes it possible to perform the linear separation. A list of the most commonly used kernel functions include the *Linear*, *Polynomial* and *Radial Basis*.

3) Random Forests (RF): Ensemble learning is the process by which multiple algorithms are combined to solve a classification or regression related task. During the past decade, numerous ensemble learning methods have been developed with the main goal to improve the performance of previous baseline approaches. Random Forests (RF) is one of the most popular and powerful ML algorithms that is based on Bootstrap Aggregation or Bagging [10]. That is, a horde of decision trees are build by using bootstrap samples of the training data. The final predictions are the average of the predictions across the different trees. Throughout the years RF has found success to a wide variety of applications in fields such as geography, computer vision, and ecology [11]–[13].

There are two principal hyperparameters that highly dictate the performance and computational complexity of RF. The first is the number of features randomly sampled within each individual tree. This hyperparameter heavily relies on the data and it is usually defined either as the square root of the available explanatory variables or by using a cross validation scheme towards finding the optimal value. The second is the number of trees that are used during the prediction phase. The selected value should provide a good compromise between accuracy, computational complexity and probability of overfitting in the training data.

B. Feature Selection

In ML and data analytics, the process of selecting the most important features for a predictive model is known as *feature selection* or *variable selection*. Feature selection is independent of any genre of ML algorithms and it is recommended as a preprocessing stride before progressing to any algorithmic paradigms. The objective of feature selection is two-fold: first, to eliminate possible features in the dataset that constitute noise and, second, to reduce the outcome error and subsequently enhance the accuracy of the model. Therefore, all features that do not contribute *valuable* information to the learning algorithm are removed. Variable selection not only reduces the feature space but also helps prevent overfitting on the training data. As a side note, variable selection is a special case of dimensionality reduction. The main point of difference though is that in dimensionality reduction the new set of features do not have to be a subset of the original set but can rather be synthetic outcomes from linear combinations. In this work, we do not consider dimensionality reduction as data does not suffer from the *curse of dimensionality*.

Data collection when it comes to speed tests is by no means an easy or a forthright process and at times comes with a certain *cost*, in terms of data required to collect a specific attribute. Although there exists no available quantitative variable that measures cost in neither datasets, we can still approximate it by the amount of data consumed which results in a different monetary cost depending on the data plan of the user. That being said, there are numerous features that we can collect free of charge. Some examples include the LTE device category, MNO, RSRP, RSRQ, etc. On the contrary, collecting mobile and network related data such as throughput, requires a clear-cut number of over-the-air transmissions, hence, data consumption is inevitable. In Table II, we divide the space of features in three diverse clusters based on how *pricey* they are to collect in terms of data consumption.

Features $1 - 9$ correspond to the group that does nor incur any cost and it is denoted with the `CL1` identifier. In LTE networks, the mobile device systematically measures the Received Signal Strength Indication (RSSI) and RSRP to decide if a handover between two cells is in order. Both parameters are reported by the Android API while RSRQ is determined by using the expression: $RSRQ = N * (RSRP/RSSI)$, where N is the number of available resource blocks in the channel. Latitude and longitude dictate the geo-location of the mobile device as exploited to estimate the covered distance during a speed test. Both coordinate values are available from the built-in GPS tracker system. The temporal related parameters and the MNO name are available from the Android clock application and a combination of the home and access network, respectively. A number of continuous transmissions of TCP ping packets are required to estimate latency between the client and the server. Due to the data exchange being brief, latency related statistics are appraised as zero-cost features.

Download Test and *Upload Test* are the main components of a speed test where multiple TCP threads between the client and the server are opened for a fixed duration of time to estimate throughput. In terms of data consumption, *Download Test* test is more *expensive* as the supported bandwidth for the two links is most of the times not symmetric. Therefore, we split the remainder of the features in two clusters. `CL2` consists of the UL related features (10 and 11) and `CL3` of the DL related features $(12 - 14)$.

Prior to the variable selection scheme, we fit a MLR model with `downlink` as the dependent variable and all the remaining features as the explanatory variables. This is a preliminary step that aims to divulge the impact of higher speed test duration times on the estimation error. We define the *error metric* as the absolute values of the residuals divided by the groundtruth values. The median value is also known as MdAPE. Figure 1 depicts error boxplots along different speed test duration scenarios. To capture signs of diversity between the two datasets, boxplots are overlaid. We observe that variance and MdAPE follow a decreasing trend for higher duration scenarios. Moreover, RTR appears to have a slight edge over MONROE that is due to differences between the datasets characteristics. Such examples include geographical differences (Austria vs Norway), number of features (`move` and `lte_cat` do not exist in MONROE), operating systems (Android vs Debian Linux/GNU), data collection timespan (10 vs 5 months) and so on. However, feature selection reveals that any superiority can be diminished or yet reversed if less features are accessible.

Forward Selection is a well-known data-driven iterative approach that makes use of a model fit criterion to decide on the importance of the available explanatory variables. The forward selection algorithm begins with a *null* model, where zero features are present, and gradually adds features that improve the regression fit the most. The process terminates when no significant improvement is obtained by the addition of a feature. Examples of stopping metrics include the adjusted R-squared, Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and probability values (P-values). In this paper, we select the adjusted R-squared which is defined as the percentage of the dependent variable variation that is explained by the linear model. More specifically, it measures how close the data points are to the regression line, or in other words, how well the model fits the data. In MLR, Adjusted R-squared is also known as the coefficient of multiple determination. Opposed to R-squared, which always increases along the number of features, adjusted R-squared adds a *penalty* term when increasing the number of features in the predictive model. Since we split the available features in clusters, forward selection takes place in *levels*. During the first level, we consider only features from `CL1` as available candidates for selection. When no improvement is observed, we update the list of candidates with features from `CL2` or `CL3`. That way, we clearly show the contribution of each cluster to the DL throughput estimation. To cover a variety of scenarios and quantify the error diversity, we select subsets from both datasets for five different duration scenarios (0.35 s, 0.7 s, 1.4 s, 2.8 s and 4.2 s). Therefore, we repeat the forward selection algorithm five times for each dataset.

In Figure 2, we present results from the forward selection algorithm. We divide the subplots in three parts, each one mapping to a cluster as defined above. The x-axis shows the ordering of the features at each iteration of the forward selection process while the y-axis shows the corresponding adjusted R-squared values. All five time duration scenarios are illustrated by a dotted line as sketched in the legend. Lines in `CL1` coincide among different time durations since the features involved remain unchanged through time. We observe a slight improvement when adding features from `CL2`. However, the bump in adjusted R-squared is paltry even when the duration

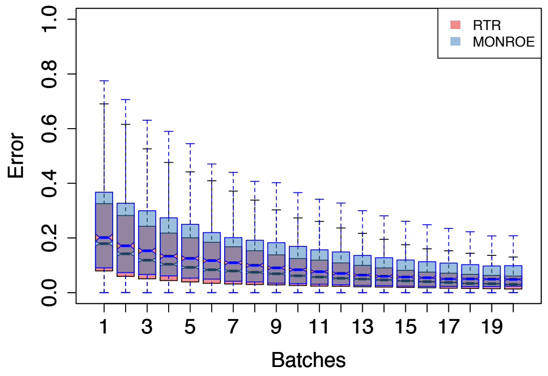
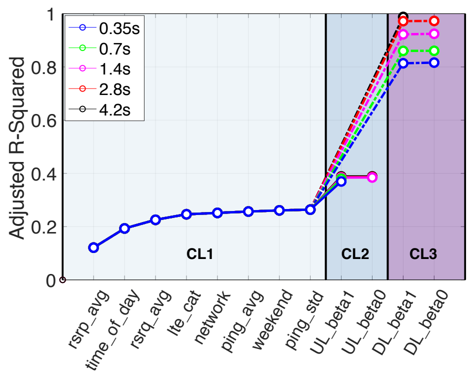
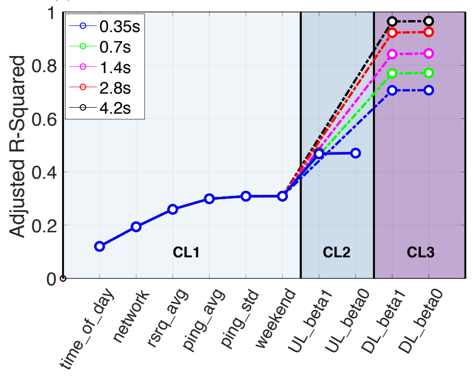


Fig. 1: Error boxplots along different speed test duration scenarios. To better visualize the differences between the two platforms, the boxplots are overlaid.



(a) Forward selection for RTR-NetTest.



(b) Forward selection for MONROE-NetTest.

Fig. 2: Feature selection algorithm. x-axis depicts the selected features at each iteration while y-axis shows the respective adjusted R-squared values. Both plots are divided in three clusters (`CL1`, `CL2` and `CL3`), signified by different coloring.

of the *Upload Test* is 4.2 s, which makes it ineffective for DL throughput estimation. Conversely, the improvement in adjusted R-squared is tremendous with the addition of features from `CL3` which clearly reveals the significant gains that DL information brings in the learning model.

VI. Performance Evaluation

To efficiently demonstrate the potential of the proposed supervised ML solution for DL throughput estimation in

MBB networks, we divide performance evaluation in two subsections. First, we present model selection and comparison between the three forecasting algorithms presented in Section V-A and second, we exhibit the trade-off between error and over-the-air data consumption under different combinations of clusters and speed test duration scenarios.

A. Performance Comparison of the Forecasting Models

To address the effectiveness of each forecasting algorithm, we present results in terms of Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and R-Squared. We carry out comparison by adopting repeated cross validation on the training data with 10 folds and 3 repeats, which is a common configuration when comparing ML models[13]. The final evaluation takes place on the testing data. We select the *Radial Basis* as the kernel function for SVR as it is shown to outperform the *Polynomial* [14]. As for RF, we decide on the number of random variables available for splitting at each tree node by using cross validation techniques while we restrain the number of trees to 100 [15].

In Figure 3 we show density plots for MAE, RMSE and R-squared. Density plots portray the precise form of a distribution, making it easier to detect and expose discrepancies and similarities between numerous instances. Each line color maps to a different forecasting algorithm while the shape dictates the combination of clusters. The solid lines (`Algorithm_name1`) represent `CL1`, the dashed lines (`Algorithm_name12`) a combination of `CL1` and `CL2` and the dotted lines (`Algorithm_name13`) a combination of `CL1` and `CL3`. Thickness does not encode any piece of information but aims to ease readability of the plot. We show results only for RTR since the picture is nearly similar for the MONROE use case. To provide a fair comparison between the forecasting algorithms, we only include samples from a single speed test duration (2.8 s). This selection is arbitrary whilst signs of disparity can be observed for durations smaller than 3.5 s where limited DL information is current. Among the available algorithms, the distribution of RF density function has nearly always the lowest mean values for MAE and RMSE and highest mean values for R-squared with SVR and MLR following short behind. However, the performance of all three algorithms becomes close even with the insertion of `CL3`. It is remarkable that in that case (`_13`) MLR performs as well as RF despite the fact that it under performs in scenarios `_1` and `_12`. As MAE and RMSE decreases and R-squared increases with the addition of `DL_beta0` and `DL_beta1`, the distributions become extremely narrow, thence, variance is reduced remarkably. As for the models computational complexity, RF has the highest followed by SVR and MLR.

B. Trade-off Between Data Consumption and Error

In this paper we propose a ML solution that estimates DL throughput accurately while reducing the amount of data traffic exchanged over the network. Therefore, the trade-off between error and data consumption is of utmost importance.

Figures 4a and 4b illustrate the percentage of data consumption for each platform. *Data consumption* is determined as the number of DL traffic exchanged until a nominal duration of time divided by the total traffic it would have been exchanged if the speed test was run for the full duration of 7000 ms[14]. Since `CL1` comprises of features that do not incur any cost during data collection, the data consumption percentage is 0% for each of the engaged features. For both datasets, the DL regression coefficients obtain a close to double increase in terms of data consumption compared to the UL regression coefficients. The above indicates that, first, the two links are not symmetric (in terms of bandwidth capacity), and second, the NetTest measurement methodology for both *Download Test* and *Upload Test* is consistent and can be generalized for all possible speed test duration scenarios.

Likewise, Figures 4c and 4d depict the reciprocal MdAPE values for each of the feature selection steps. Again, we observe that the addition of `CL3` highly affects the performance of our models and contributes in reducing the MdAPE significantly. Furthermore, we show that higher speed test duration values improve the accuracy levels considerably.

VII. Discussion

We discuss next on the principal takeaways from Section VI. As we already determined from the feature selection phase, the gains that the UL related features bring to the DL throughput estimation error are trifling and they are not improving even for large speed test duration scenarios. Therefore, one should never consider UL traffic as an alternative traffic type to assess DL throughput. Furthermore, the improvement in terms of MdAPE between any speed test duration scenario is only highly relevant for `CL3`, where DL comes in play. For example, the decrease in MdAPE between 1.4 s and 2.8 s is 4.55% in average which is pretty significant for highly sensitive applications.

Another important topic of discussion is with regards to the model validity. In ML, retraining a model is required when the new data belongs to a disparate distribution than the one the model was originally trained. Retraining can be performed either online, offline or by using the batch approach. Hence, one question that remains is how often should we retrain our models. Intuitively, if no change occurs in the way a NetTest speed test runs, retraining should not be too often and it is only needed to catch up with new device models and seasonal trends. However, if a switch takes place in NetTest methodology (see Section II) or it is replaced by a new tool, retraining is highly recommended as it can lead to a significant accuracy degradation. We plan to run a detailed analysis of the horizon of the model predictions in future work.

[13]In general, cross validation techniques for RF are not needful to get an unbiased estimate of the testing data error, since it is estimated internally. However, for consistency's sake, we use the same control setup when reporting results for all forecasting algorithms.

[14]We consider 7000 ms as the baseline speed test duration value to beat. In case a different speed test tool is available, the baseline value has to be updated accordingly in order to estimate the gains (in terms of data usage) that our proposed approach offers.

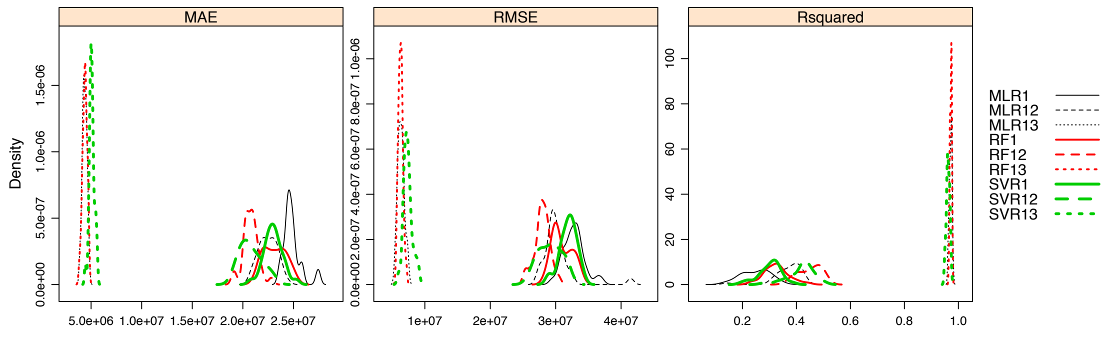
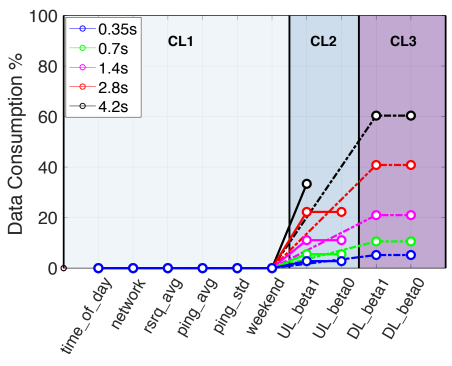
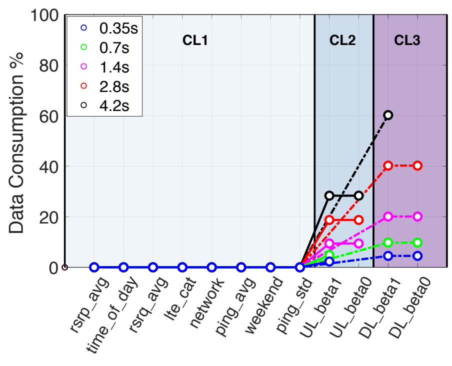


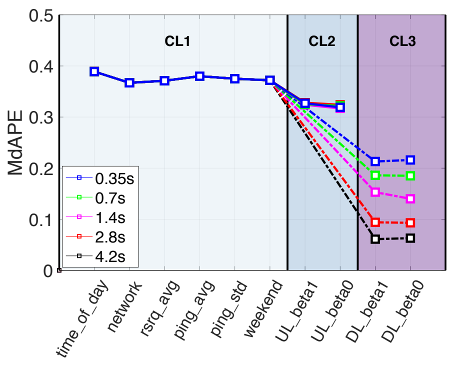
Fig. 3: Illustration of the distribution of MAE, RMSE and R-squared with density plots for RTR-NetTest. Line colors map to regression algorithms while line shapes dictate combinations of clusters.



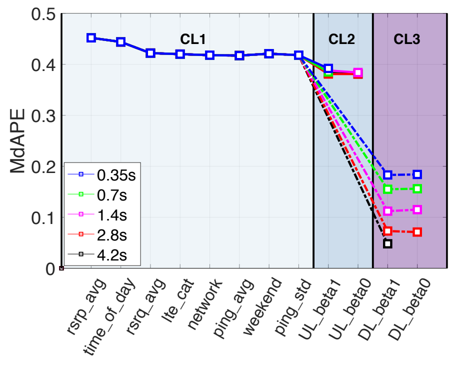
(a) Data consumption in MONROE.



(b) Data consumption in RTR.



(c) MdAPE in MONROE.



(d) MdAPE in RTR.

Fig. 4: Trade-off comparison between MdAPE and over-the-air data consumption in MONROE and RTR. For all displayed results, we use MLR as the supervised forecasting method. To ease comparison, illustration is in accordance with the plots of Figure 2.

It is also noteworthy to mention that we have to design a single model for each NetTest testing methodology (crowdsourced and controlled). As mentioned in Section V, although the measurement methodologies are similar, datasets characteristics differ sufficiently with regards to the network infrastructure, therefore, the design of a global supervised model is not recommended. In other words, training a forecasting model on one dataset and testing on another would fail to generalize and produce reliable results.

To conclude the discussion part, we have to comment on the appropriate speed test duration value that should be leveraged on each dataset. We find that there is no straightforward answer to that argument as the *optimal* duration value highly relates to the application's error sensitivity requirements. For example, if an application requires a rough estimation of the network bandwidth capacity without having to consume a lot of data traffic, it should always opt duration values smaller than a couple of seconds. However, if high accurate DL throughput estimates are of vital importance, exploiting higher duration values will improve the performance error but inevitably increase data consumption considerably. We recommend the reader to use the four subplots in Figure 4 as a guideline to decide for a duration value that offers an appropriate trade-off for the specific purpose of the measurement campaign.

VIII. Related Work

Over the years, there have been many attempts to model and predict throughput behavior in wireless networks. In [16], the authors propose an empirical approach for TCP throughput prediction based on past file transfers and measurements of simple path properties, such as queuing delay and loss. Evaluation analysis is being held in a fully controlled environment and the results reveal a significant improvement in terms of accuracy over pure history-based methodologies. Experimental approaches are also followed in [17], where the authors conduct a thorough comparison of regression algorithms to determine the prediction of TCP throughput in operational MBB networks, and [18], where Rattaro et al. leverage SVR as a ML tool to model throughput on IEEE 802.11 networks.

In contrast with the prior empirical approaches, a novel stochastic model for user throughput prediction in MBB networks is presented in [19]. The proposed solution takes into consideration sources of prediction precision, like fast fading and user location. A crowdsourced large-scale dataset from Bredbandskollen in Sweden is utilized in [20] to estimate network performance. The authors present a scalable performance map approach to characterize measurement usage in the spatial domain. Furthermore, they present statistical tools to analyze bandwidth variation and predictability of the download speeds observed within and across different locations. Last, in [21], the authors leverage machine learning techniques including Gaussian process regression, exponential smoothing of time series, and random forests to predict performance and improve user experience in cellular networks. They focus on three types of prediction problems (spatial, temporal, and multidimensional) while for the validation process they use a real world dataset in US consisting of 4G measurements.

IX. Conclusions and Future Work

In this paper, we presented a supervised ML solution for Downlink throughput estimation in MBB networks that delivers highly accurate results and reduces the over-the-air data consumption considerably. While we carried out the evaluation analysis for the RTR-NetTest crowdsourced dataset and its counterpart wrapper MONROE-NetTest, our approach can be also applied in real time applications with limited data plans where throughput prediction is crucial for enhancing QoS. Future work includes a more exhaustive study of ML (or even Deep Learning) schemes that can boost accuracy at higher levels. Finally, we will consider the development of an open-source application that wraps up the functionality of the proposed methodology and could be used as an alternative speed test solution in the Android (or iOS) stock market.

X. Acknowledgments

This work is funded by the EU H2020 research and innovation programme under grant agreements No. 644399 (MONROE) and No. 815178 (5GENESIS), and by the Norwegian Research Council project No. 250679 (MEMBRANE).

References

[1] Ö. Alay, A. Lutu, R. García, M. Peón-Quirós, V. Mancuso, T. Hirsch, T. Dely, J. Werme, K. Evensen, A. Hansen *et al.*, "Measuring and assessing mobile broadband networks with monroe," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2016 IEEE 17th International Symposium on A*. IEEE, 2016, pp. 1–3.

[2] Ö. Alay *et al.*, "Experience: An open platform for experimentation with commercial mobile broadband networks," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. ACM, 2017, pp. 70–78.

[3] C. Midoglu, L. Wimmer, A. Lutu, Ö. Alay, and C. Griwodz, "Monroe-nettest: A configurable tool for dissecting speed measurements in mobile broadband networks," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 342–347.

[4] T. Özel and Y. Karpat, "Predictive modeling of surface roughness and tool wear in hard turning using regression and neural networks," *International Journal of Machine Tools and Manufacture*, vol. 45, no. 4-5, pp. 467–479, 2005.

[5] J. H. Bolin, "Hayes, andrew f.(2013). introduction to mediation, moderation, and conditional process analysis: A regression-based approach. new york, ny: The guilford press," *Journal of Educational Measurement*, vol. 51, no. 3, pp. 335–337, 2014.

[6] J. D. Angrist and J.-S. Pischke, *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

[7] H. Passing and W. Bablok, "A new biometrical procedure for testing the equality of measurements from two different analytical methods. application of linear regression procedures for method comparison studies in clinical chemistry, part i," *Clinical Chemistry and Laboratory Medicine*, vol. 21, no. 11, pp. 709–720, 1983.

[8] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[9] D. Basak, S. Pal, and D. C. Patranabis, "Support vector regression," *Neural Information Processing-Letters and Reviews*, vol. 11, no. 10, pp. 203–224, 2007.

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] P. O. Gislason, J. A. Benediktsson, and J. R. Sveinsson, "Random forests for land cover classification," *Pattern Recognition Letters*, vol. 27, no. 4, pp. 294–300, 2006.

[12] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, 2007, pp. 1–8.

[13] A. M. Prasad, L. R. Iverson, and A. Liaw, "Newer classification and regression tree techniques: bagging and random forests for ecological prediction," *Ecosystems*, vol. 9, no. 2, pp. 181–199, 2006.

[14] Z. Ramedani, M. Omid, A. Keyhani, S. Shamshirband, and B. Khoshnevisan, "Potential of radial basis function based support vector regression for global solar radiation prediction," *Renewable and Sustainable Energy Reviews*, vol. 39, pp. 1005–1011, 2014.

[15] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas, "How many trees in a random forest?" in *International workshop on machine learning and data mining in pattern recognition*. Springer, 2012, pp. 154–168.

[16] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to tcp throughput prediction," in *ACM SIGMETRICS Performance Evaluation Review*, vol. 35, no. 1. ACM, 2007, pp. 97–108.

[17] Y. Liu and J. Y. Lee, "An empirical study of throughput prediction in mobile data networks," in *Global Communications Conference (GLOBECOM), 2015 IEEE*. IEEE, 2015, pp. 1–6.

[18] C. Rattaro and P. Belzarena, "Throughput prediction in wireless networks using statistical learning," in *LAWDN-Latin-American Workshop on Dynamic Networks*, 2010, pp. 4–p.

[19] N. Bui, F. Michelinakis, and J. Widmer, "A model for throughput prediction for mobile users," in *European Wireless 2014; 20th European Wireless Conference; Proceedings of*. VDE, 2014, pp. 1–6.

[20] T. Linder, P. Persson, A. Forsberg, J. Danielsson, and N. Carlsson, "On using crowd-sourced network measurements for performance prediction," in *Wireless On-demand Network Systems and Services (WONS), 2016 12th Annual Conference on*. IEEE, 2016, pp. 1–8.

[21] J. Riihijarvi and P. Mahonen, "Machine learning for performance prediction in mobile cellular networks," *IEEE Computational Intelligence Magazine*, vol. 13, no. 1, pp. 51–60, 2018.