Prediction of Cellular Service Quality

Maxmillian Minichetti[1] | Stanford University

I. Introduction

The primary objective of this study is to make intelligent predictions about a user's cellular service quality from widely accessible network data and basic device information. I use ordinary least squares (OLS), support vector regression (SVR), and random forest regression (RFR) to predict the received signal strength indicator (`RSSI`) of a mobile device. `RSSI`, measured in decibels (dB), is generally negative with 0 dB corresponding to the strongest possible signal. This number represents the power level that a device is receiving from an access point or router in some wireless network [13]. The champion predictive model achieves a root-mean-square error (RMSE) of 15.5 dB on `RSSI` test data ranging from 0 to 250 dB in magnitude. The input to this algorithm consists of discrete and continuous variables. Continuous inputs are given by the following list: battery level, latitude, longitude, round-trip delay time, data packets lost, and data packets sent. The discrete inputs are as follows: battery charging status, network type, time of day, device manufacturer, and cell carrier.

In the past, supervised learning algorithms have been used to locate radio frequency emitters from the `RSSI` of a transmitted signal [2]. Other works have employed machine learning in the task of indoor sensor localization for improved precision and noise reduction in high performance IoT sensor networks [7, 8]. These studies use `RSSI` data to obtain meaningful geolocation insights for optimizing the efficiency of wireless systems. Cheng et al. demonstrates that SVR can effectively predict sensor location as a function of `RSSI`, outperforming K-nearest neighbor regression and artificial neural networks [3, 4]. They report an ensemble learning approach, in which the resulting SVR model represents the weighted average of multiple SVR models trained on random subsets of the original data. This is a clever way to enhance generalization performance by preventing overfitting and reducing variance; however, the learning process is computationally intensive, time consuming, and very difficult to tune. Solving the opposite problem of predicting `RSSI` as a function of potentially noisy location data has been a topic far less explored. Previous work on this subject posed the issue as a classification problem, likely because noisy location data makes the continuous prediction of signal strength very challenging [9]. In my project, I focus on improving the continuous prediction of `RSSI`, by incorporating device information, such as battery level, charging status, and device manufacturer, which is conventionally ignored. Intuitively, I suspect a high indirect correlation between these basic device properties and signal strength. For instance, from prior experience I am led to believe that mobile device users typically charge their electronics at home or at work when they have access to strong WiFi signals. As a result, it would make sense for signal strength magnitude to be positively correlated with the indicator of charging status and potentially battery level.

Current mobile devices can tell us when a cell signal or WiFi connection is weak; however, they fail to provide useful information for obtaining a better signal. A machine learning model capable of predicting `RSSI` from a combination of network conditions and geolocation could provide mobile device users with a personal heuristic for locating areas of optimal and suboptimal cell service. In addition, the model for predicting `RSSI` would benefit cell service providers in determining a favorable deployment of cell towers over a region. Thus, as cell carrier companies work to improve their coverage by using this learning algorithm, periodically retraining the model to reflect these targeted improvements results in a positive feedback loop to benefit mobile device users all over the world.

II. Dataset and Feature Extraction

MobiPerf is an open source application that monitors network performance of mobile devices [14]. Each log contains network measurements taken across many devices at specific times throughout the day [15, 16]. The data is recorded in `json` format, spanning days from over five years. In this project I use a collection of MobiPerf traces between November 1, 2016 and November 15, 2016. Each log records latency, bytes transferred, and other relevant attributes of mobile devices sending queries through `ping`, `traceroute`, and `HTTP` requests. I focus only on `ping` requests for the purposes of this study, as they are most relevant to network connectivity testing [17]. The following set of features were considered for model training: battery level, charging status, network type, cell carrier, device model, manufacturer, latitude, longitude, data packets sent, packets lost, and time of day. Features corresponding to columns with a large number of `<NA>` entries were discarded. Covariates irrelevant to the regression problem, namely device serial numbers and IP addresses, were also removed. Samples containing `<NA>` entries were ignored for training. After compiling the MobiPerf logs into a single data frame of nearly 9,000 samples, 20% of the data was set aside for testing. The remaining 80% was used for model training, validation, and cross-validation. A subset of the features and samples is shown atop the following page:

[1] maxmin@stanford.edu, Undergraduate Student, Department of Electrical Engineering

	battery_level	is_charging	network_type	RSSI	longitude	latitude	stddev_rtt	mean_rtt	max_rtt	min_rtt	packet_loss	packets_sent
72971	100	0	1	-51	-83.74	42.26	2.9097251	16.650	22.90	12.60	0.00000000	10
10051	24	1	1	-46	0.00	0.00	4.2147361	44.900	53.10	39.50	0.00000000	10
7058	100	0	0	-127	25.58	-33.80	103.4922702	273.500	513.00	192.00	0.09090909	11
6149	100	0	1	-67	-83.66	42.25	11.3560557	29.800	57.00	20.00	0.00000000	10
3061	100	0	1	-51	-43.95	-19.89	4.3543082	30.800	41.00	25.00	0.00000000	10
7417	62	1	0	-127	0.00	0.00	10.6152909	63.740	77.50	39.00	0.00000000	10
2361	100	1	1	-53	0.00	0.00	9.0677505	41.630	52.00	26.20	0.00000000	10

III. Methods

Models were trained in R using OLS, SVR, and RFR. RMSE obtained via 10-fold cross-validation was used to compare between models. Under K-fold cross-validation the training set is randomly divided into K equal groups. Let A_k denote the set of training examples $(\mathbf{X}_i, Y_i)$ grouped into the k-th subset. For $k = 1, 2, \ldots, K$, a model is trained on all but the k-th fold. Let $\hat{f}^{-k}$ denote the resulting fitted model. The cross-validation error metric is computed as follows:

$$\text{Error}_{\text{cv}} = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{n/K}\sum_{i\in A_k}\left(Y_i - \hat{f}^{-k}(\mathbf{X}_i)\right)^2\right)}$$

Averaging the generalization performance observed over all K folds provides a fairly reliable estimate for the actual test error of a model trained on the entire sample [1].

A. Ordinary Least Squares Regression

Given the training data, we seek to minimize the following objective function, $J(\theta)$, by generating a prediction vector as close as possible to the vector of observed values. This entails finding an optimal set of weights θ^* corresponding to the linear predictor function $h_\theta(x)$. By convention, note that $x_0^{(i)} = 1$ for all $i \in \{1, 2, \ldots, m\}$. This value corresponds to the intercept term θ_0 of the learned OLS model.

$$J(\theta) = \|X\theta - y\|^2 = \sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2, \quad h_\theta(x) = \sum_{i=0}^{n}\theta_i x_i = \theta^\top x, \quad \text{with} \quad \theta^* = \left(X^\top X\right)^{-1}X^\top y$$

After adding higher-order terms and interactions to the baseline feature set, I used the forward search algorithm as an initial approach to model selection. The algorithm works as follows. To start, let R be a set of all possible regressors under consideration for the final model, and let $\mathcal{F} = \emptyset$. On each iteration, the algorithm identifies a single regressor $r \in R$ that leads to the greatest reduction in model score. The covariate r is removed from R and added to the current feature set $\mathcal{F}$. The algorithm terminates once the addition of any and all features remaining in R fails to lower the current model score. Traditionally, the Akaike information criterion (AIC) is used in forward search; however, I consider the Bayesian information criterion (BIC) as another metric for more strictly penalizing model complexity. Since the initial regressor set R is so large for our linear models ($|R| > 500$), it would be interesting to see if forward search using BIC could learn models with comparable performance and fewer regressors than those generated via AIC.

$$AIC = \frac{n}{\hat{\sigma}^2}\left(\text{Err}_{\text{tr}} + \frac{2|S|}{n}\hat{\sigma}^2\right), \quad BIC = \frac{n}{\hat{\sigma}^2}\left(\text{Err}_{\text{tr}} + \frac{2|S|\ln(n)}{n}\hat{\sigma}^2\right)$$

Elastic net regularization is used as a shrinkage method for model selection. Relative to OLS, elastic net regression yields coefficients θ that have been effectively shrunk towards zero. The most explanatory covariates are retained. The addition of L1 and L2 norm penalty terms to the previous objective, $J(\theta)$, effectively reduces overfitting.

$$\begin{aligned} J_{EN}(\theta) &= \|X\theta - y\|^2 + \lambda\left(\alpha\|\theta\|^2 + (1-\alpha)\|\theta\|_1\right) \\ &= \sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \lambda\left(\alpha\sum_{j=1}^{n}|\theta_j|^2 + (1-\alpha)\sum_{j=1}^{n}|\theta_j|\right) \end{aligned}$$

This tends to improve generalization performance of the model while naturally selecting for the most explanatory covariates. The optimal elastic net regression model with hyperparameters λ^* and α^* are obtained using 10-fold cross-validation.

B. Support Vector Regression

In SVR, the goal is to obtain a function $f(x)$ that predicts the response variable within a designated error, ϵ. SVR effectively generates an optimal hyperplane $f(x) = \theta^\top x + b$ by solving the following optimization problem:

$$\text{minimize } \frac{1}{2}\|\theta\|^2 + C\sum_{i=1}^{m}(\delta_i^- + \delta_i^+), \quad \text{subject to} \begin{cases} y^{(i)} - \theta^\top x^{(i)} - b & \leq & \epsilon + \delta_i^- \\ \theta^\top x^{(i)} + b - y^{(i)} & \leq & \epsilon + \delta_i^+ \\ \delta_i^-, \delta_i^+ & \geq & 0 \end{cases}$$

Introducing slack variables δ_i^- and δ_i^+ as well as a user-specified penalty term, $C \geq 0$, allows SVR to generate more robust models. Grid search over the parameter space of C versus ϵ presents an avenue for obtaining an optimal SVR with strong generalization performance.

A standard dualization method with Lagrange multipliers $\lambda_i \geq 0$ and $\lambda_i^* \geq 0$ can be used to solve this optimization problem [10, 11]. The resulting model is given by the following:

$$f(x) = \sum_{i=1}^{m} (\lambda_i - \lambda_i^*)\, K\big(x^{(i)}, x\big) + b, \quad \theta^* = \sum_{i=1}^{m} (\lambda_i - \lambda_i^*)\, x^{(i)}$$

Observe, it is possible to apply a kernel function $K(x, z)$ to provide a nonlinear mapping from the input space to the SVR feature space. I investigate the use of three different kernel functions:

$$\text{Linear: } K(x,z) = x^\top z, \quad \text{Polynomial: } K(x,z) = (x^\top z + c)^d, \quad \text{RBF: } K(x,z) = e^{-\|x-z\|^2/2\sigma^2}$$

C. Random Forest Regression

RFR is a modification on the tree bagging algorithm [12]. In bagging, successive regression trees are trained on bootstrapped subsets of the original data. Unlike boosting, in which successive trees depend on earlier trees, random forest effectively decorrelates the intermediate decision trees by introducing another layer of randomness to the bagging algorithm. In a standard decision tree, nodes are split using the optimal bisection over the entire feature set; however, in RFR, nodes are split using the best among a subset of features randomly selected at each node.

Let n_t represent the number of trees (i.e., the number of learners) in a random forest model, and let m_t be the number of variables selected for each randomly generated feature subset. To start, n_t bootstrapped samples are pulled from the training set. For each bootstrap sample, the algorithm grows an unpruned regression tree using the following rule: at each node, randomly sample m_t features and choose the best split over those predictors. The model generates a forecast by averaging the predictions of all n_t decision trees. The resulting model is given by $f(x) = \frac{1}{n_t}\sum_{i=1}^{n_t} f_b(x)$, where $f_b(x)$ denotes an intermediate regression tree trained on $(\mathbf{X}_b, Y_b)$, a bootstrapped subset of the training data.

IV. Results and Discussion

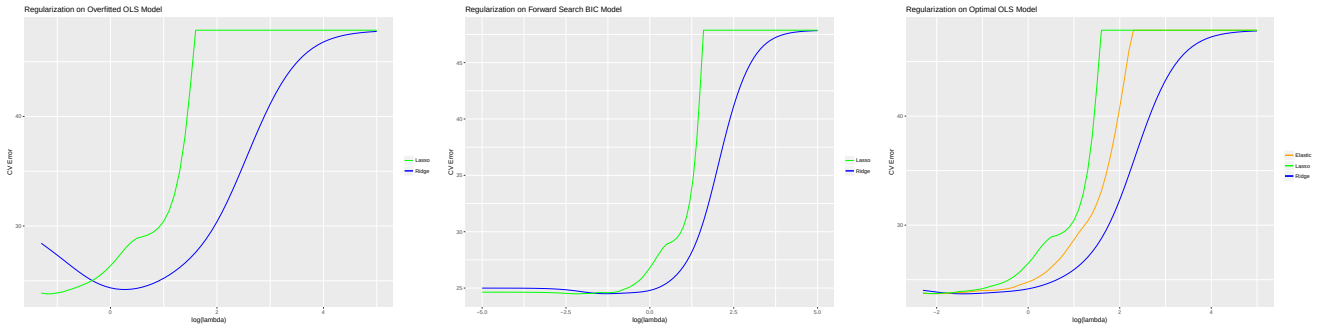
The baseline model incorporates all first-order covariates pulled directly from the training set. OLS regression was used to fit this model, achieving train and test set errors of 27.921 dB and 28.625 dB, respectively. The baseline serves as a naive predictor of `RSSI`, providing general intuition for identifying the most significant regressors. The strongest models obtained from each method and their corresponding performance metrics are given in the following table:

Model	Train	10-Fold CV	Test
OLS (AIC Forward Search)	23.135	24.157	25.129
SVR (RBF, $C = 512$, $\epsilon = 0.14$)	14.665	19.182	20.062
RFR (1000 Learners)	7.761	15.688	15.513

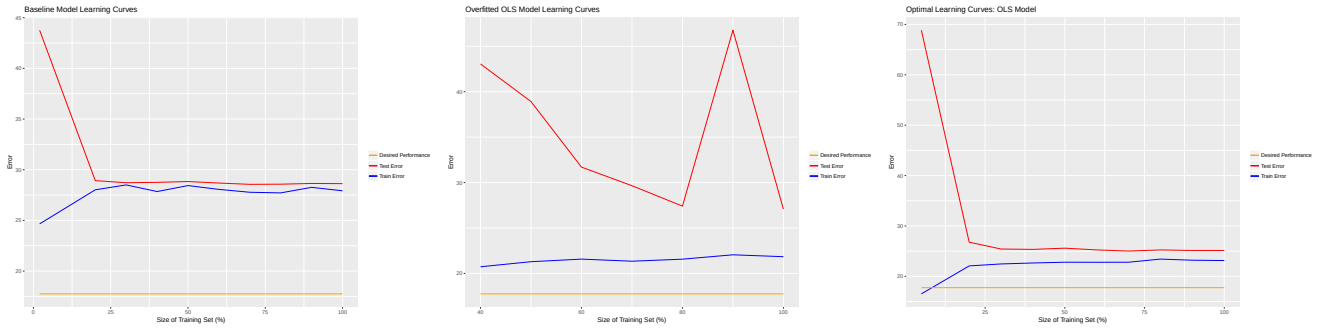
A. Optimizing Ordinary Least Squares Regression

To start, I trained an all covariates model consisting of the baseline feature set, higher-order terms (up to degree 8), and all possible interactions. Forward search using AIC and BIC model scores was used to directly extract the most significant regressors from the all covariates model. The optimal linear model, demonstrating the best tradeoff between bias and variance, was achieved with AIC forward search. However, this model remained highly complex with nearly 130 covariates. Elastic

net regularization was applied to systematically shrink coefficients associated with the weakest predictors. Observe, for α close to 1, elastic net regularization resembles ridge regression, and for α near 0, elastic net more closely resembles LASSO regularization. Frequently, the optimal elastic net regression model was one of these special cases. Plots of cross-validation error versus the hyperparameter, λ, are shown below:



Ridge regression ($\alpha = 1$) on the all covariates model (of more than 500 terms) obtains an optimal $\lambda = 1.99$ and a corresponding cross-validation error of only 24.204 dB. In general, elastic net regression fails to achieve a cross-validation error below 23.807. Subsequent models obtain optimal hyperparameter values even closer to 0, suggesting that nearly all of their corresponding features are deemed significant by the algorithm. This evidence strongly supports that OLS fails to appropriately capture `RSSI`, as increasing complexity of the feature set results in only marginal improvements in cross-validation error. Thus, elastic net regularization explores the performance upper limit of a linear model attempting to predict a highly nonlinear response. The AIC forward search model ultimately outperforms the optimal elastic net models by an average of 1.2 dB in test error.



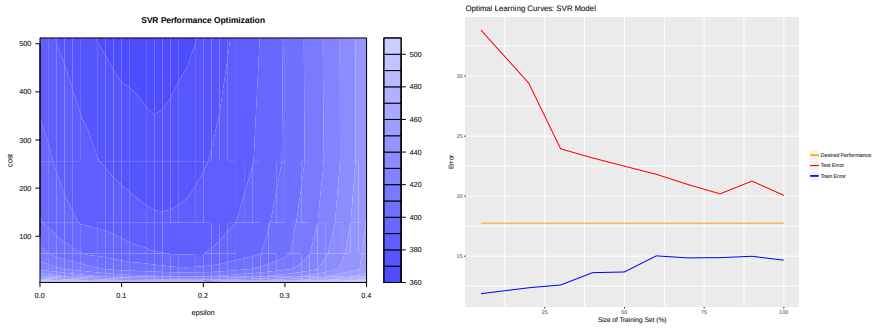
In reference to the learning curves above, the baseline model struggles to achieve a bias margin less than 10 dB relative to the desired performance standard of 17.75 dB. While bias appears to be an issue for the baseline model, variance is the dominating limitation for the all covariates model. Qualitatively, this would suggest that the all covariates model surely overfits the training data. Despite extensive model selection, the optimal OLS regression only marginally reduces bias, relative to that of the baseline. This verifies that a linear model is insufficient for capturing nonlinearities in the relationship between `RSSI` and the current feature set.

B. Optimizing Support Vector Regression

To verify nonlinearity in the data, I trained kernelized SVR models on three different kernel functions: linear, polynomial, and RBF. As expected, the nonlinear RBF kernel produced the most accurate SVR models. Among the feature sets previously obtained by running forward search on OLS models, SVR performed best with the simple, baseline feature set of first-order regressors. SVR models learned on the BIC or AIC feature sets, containing interactions and higher-order terms, overfit the training data resulting in poor generalization performance. Thus, encoding a nonlinear mapping from first-order input features, with the RBF kernel alone, is sufficient for capturing inherent nonlinearities in the data.

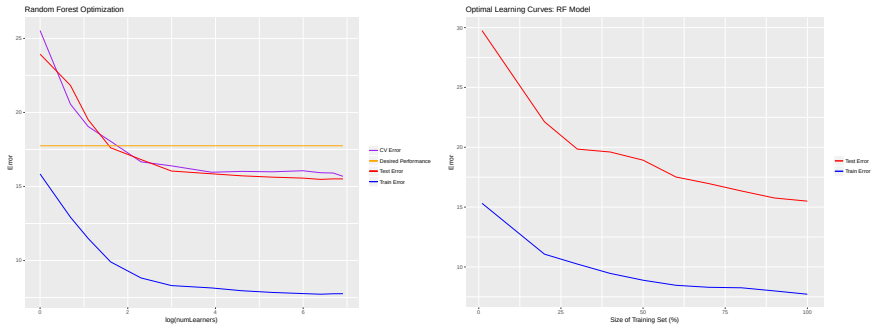
After identifying the most promising structure for SVR, many models were trained over the parameter space of cost, C, versus error margin, ϵ. The following heatmap was generated, showing 10-fold cross-validation error as a function of both hyperparameters. A global optimum has yet to be found, suggesting the need for further exploration of the hyperparameter space. While the optimal SVR model, with $C = 512$ and $\epsilon = 0.14$, achieves a test error of 20.062 dB, this model can surely be improved by considering other cost/epsilon combinations.

Learning curves associated with the optimal SVR model, demonstrate a superior tradeoff between bias and variance. Both the training and test curves are centered around the desired performance standard of 17.75 dB, with a respectable variance gap under 6 dB.



C. Optimizing Random Forest Regression

In general, RFR is relatively straight-forward to tune, as its performance depends on only a few parameters, namely the number of learners, n_t. Similar to SVR, the baseline feature set produced the optimal RFR models relative to those trained on the highly complex AIC and BIC feature sets. The number of learners was varied from 1 to 1000 in order to obtain an upper bound on the performance of the RFR model. Error metrics were then plotted as a function of n_t.



Observe, the RFR performance converges to around 15.5 dB as n_t increases, surpassing the desired performance metric of 17.75 dB used throughout this study. One advantage to RFR is that by construction, these models are impervious to overfitting [12]. Unlike decision tree boosting, which essentially reduces bias by fitting a large number of shallow (weak) models with low variance, RFR seeks to effectively reduces variance by fitting a large number of comparatively deep decision trees with low bias. The aggregation of these randomly generated decision trees enables excellent generalization performance, especially for capturing nonlinear relationships. The optimal learning curves associated with RFR suggest that the model could potentially benefit from more data, as both curves appear capable of achieving more accurate results.

V. Conclusions and Future Work

The RFR model of 1000 learners was the most accurate model for predicting `RSSI` from simple network conditions and basic device information. The SVR model, using a RBF kernel, has the potential to perform much better with some additional searching of the hyperparameter space. SVR outperforms the best OLS model by around 20%, validating the importance of having a nonlinear model to predict `RSSI`. Despite extensive model selection, OLS was insufficient for capturing nonlinearities inherent to the data. From analyzing standard errors and model coefficients, the most important regressors for predicting `RSSI` are listed as follows in order of significance: network type, location, packet loss, round-trip time, battery level, and charging status.

Given more time and more powerful computational resources, I would consider learning an artificial neural network and continue exploring the SVR hyperparameter space. The SVR heatmap previously shown took over 8 hours to run on my personal machine. Having more powerful computational tools at my disposal would help achieve an even better SVR model. This would enable an optimized weight-learning ensemble approach to SVR, improving upon state of the art work done on this topic.

VI. Contributions

This was an individual project. All the work presented here is my very own. I wrote all code associated with this study from scratch in R. For preprocessing, I used the `jsonlite` package to convert `json` files into a single R data frame. The `glmnet` package was used for implementing elastic net regularization. The `e1071` and `randomForest` packages were used for training SVR and RFR models, respectively. Finally, the `cvTools` package was used to compute K-fold cross-validation errors.

References

[1] Cawley, G. et al., On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation, Journal of Machine Learning Research, University of East Anglia, 2010.

[2] Mukherjee, T. et al., RSSI-Based Supervised Learning for Uncooperative Direction-Finding, Intelligent Robotics Inc., 2017.

[3] Cheng, Y. et al., Machine-Learning Indoor Localization with Access Point Selection and Signal Strength Reconstruction, Research Center for Information Technology Innovation, Academia Sinica, 2014.

[4] Cheng, Y. et al., High-Precision Wireless Indoor Localization via Weight-Learning Ensemble Support Vector Regression, Research Center for Information Technology Innovation, Academia Sinica, 2014.

[5] Goudarzi, S. et al., A Novel RSSI Prediction using Imperialist Competition Algorithm (ICA), Radial Basis Function (RBF) and Firefly Algorithm (FFA) in Wireless Networks, PLOS, Malaysia-Japan International Institute of Technology, 2016.

[6] Fjelberg, M., Predicting Data Traffic in Cellular Data Networks, Master Thesis, KTH Royal Institute of Technology, 2015.

[7] Narzullaev, A. et al., Accurate Signal Strength Prediction-Based Positioning for Indoor WLAN Systems, Yeungnam Univeristy, IEEE, 2008.

[8] Suarez, A. et al., Gradient RSSI Filter and Predictor for Wireless Networks Algorithms and Protocols, Network Protocols and Algorithms, University of Las Palmas de Gran Canaria, 2010.

[9] Cass, P. et al., Predicting QoE in Cellular Networks using Machine Learning and in-Smartphone Measurements, Austrian Institute of Technology, 2017.

[10] Gunn, S., Support Vector Machines for Classification and Regression, Technical Report, University of Southhampton, 1998.

[11] Fletcher, R., Practical Methods of Optimization, John Wiley & Sons, 2013.

[12] Liaw, A., and Wiener M., Classification and Regression by randomForest, R News, Merck Research Laboratories, 2002.

[13] MetaGeek, Understanding RSSI, WiFi Lessons, 2017.
`https://www.metageek.com/training/resources/understanding-rssi.html`

[14] Mao M. et al., MobiPerf, University of Michigan, 2012.
`www.mobiperf.com`

[15] Welsh M., Measurement Lab, New America's Open Technology Institute, 2012.
`https://www.measurementlab.net/tests/mobiperf/`

[16] Google Open Source Research, MobiPerf Data Repository, 2012.
`https://console.cloud.google.com/storage/browser/openmobiledata_public`

[17] SearchNetworking, Ping, TechTarget, 2013.
`http://searchnetworking.techtarget.com/definition/ping`