4G LTE Network Throughput Modelling and Prediction

Habiba Elsherbiny*, Hazem M. Abbas†, Hatem Abou-zeidϒ, Hossam S. Hassanein† , Aboelmagd Noureldin†‡
*Electrical and Computer Engineering Dept., Queen's University, Canada, habiba.elsherbiny@queensu.ca
†School of Computing, Queen's University, Canada, hazem.abbas@queensu.ca, hossam@cs.queensu.ca
‡Electrical and Computer Eng. Dept., Royal Military College of Canada, Canada, aboelmagd.noureldin@rmc.ca
ϒEricsson Canada, hatem.abou-zeid@ericsson.com

GLOBECOM 2020 - 2020 IEEE Global Communications Conference | 978-1-7281-8298-8/20/$31.00 ©2020 IEEE | DOI: 10.1109/GLOBECOM42002.2020.9322410

Abstract— **The past decade has witnessed a staggering evolution in cellular networks. Mobile wireless technologies have undergone four distinct generations; from uncomplicated voice calls in the first generation to high-speed, low latency and video streaming in the fourth generation. The numerous services brought to the users by 4G network have caused an increasing load demand. This increasing demand in network usage has proven the necessity of further service enhancements, such as predictive resource allocation techniques and handover analysis. For these techniques to be deployed, network quality and performance analysis must be performed on real-world network data. Since throughput is a major indicator of the network's performance, throughput modelling and prediction can be utilized for analyzing network quality. In this paper, two approaches for throughput analysis are examined: classical machine learning and time series forecasting. For the first approach, various machine learning models were deployed for throughput prediction and our analysis showed that the random forest model achieved the highest prediction performance. For time series forecasting, statistical methods as well as deep learning architectures were used. The evaluation shows that the machine learning models had a higher throughput prediction performance than the time series forecasting techniques.**

Keywords— Cellular Networks, 4G, LTE, Throughput Prediction.

I. INTRODUCTION

The advancements in the cellular network technologies over the past decade have brought endless services and capabilities to the users. Smartphone users today rely on their phones for work as well as leisure activities, such as online gaming and video streaming. This has increased the load on cellular networks, causing the network traffic to increase rapidly. Cellular network operators are always looking for solutions to cope with this rising demand by developing new mechanisms for resource allocation and load balancing. An emerging paradigm for improving the network Quality of Service (QoS) and addressing network scalability issues is adopting an anticipatory approach to resource allocation and network management. The fundamental idea is to predict fluctuations in the network connectivity that a user will experience before they occur – and then leverage these predictions to take preventative measures to meet user QoS requirements. For instance, a user may be granted more resources in advance to prebuffer video content by predicting that future throughput values will decrease for that user [1]. The need and benefit of such proactive allocation schemes will only increase with the emergence of more bandwidth hungry applications, increased media publishing and streaming on social media platforms, and the rise of connected and autonomous vehicles.

Among the various network quality measures, predicting throughput is the most useful to guide anticipatory network functions. However, throughput prediction is not trivial as it is dependent on not only the received signal strength but also other measures including context information such as geographical location, landscape, and the time of the day.

The key contribution of this paper is developing several methods to model and forecast network throughput using measurements collected from a live network. This is achieved by evaluating various machine learning models and time series forecasting techniques for downlink throughput prediction. The models' performance on the test data is evaluated based on multiple evaluation metrics and an in-depth comparative analysis is made. To the best of our knowledge, this is the first 4G LTE network throughput analysis to be conducted along public transportation in Kingston, Canada.

A. Related Work

Over the years, several researchers have investigated the problem of throughput prediction. Kamakaris and Nickerson have proposed the concept of using a connectivity map for throughput estimation [2]. They investigated the relationship between the signal strength and the throughput in Wi-Fi networks. The authors found that the dynamic variations in the network conditions led to a short average lifetime of the connectivity map predictions. Pögel and Wolf also proposed the concept of using a connectivity map for predicting different network performance metrics, such as the RSSI, bandwidth and latency in a vehicular context [3]. They also performed several drive tests to collect measurements of network performance parameters in an HSDPA network. Furthermore, in their later work [4], they used the data and information they gathered previously to enhance different network services such as adaptive video streaming and the handover between different network technologies.

Furthermore, Xu et. al., developed a system interface, PROTEUS, for instantaneous throughput prediction [5]. PROTEUS uses the previous 20 seconds of observed network performance and relies on regression trees for prediction. Liu et. al., applied and compared seven different algorithms for mobile networks throughput prediction [6]. They used trace-driven data from 3G/HSPA networks to train their models. The measurements were collected in a stationary scenario and

978-1-7281-8298-8/20/$31.00 ©2020 IEEE

the model employed the throughput during each 300 seconds to predict the throughput for the next 300 seconds.

In 2015, Jin investigated the problem of applying throughput prediction algorithms in 4G LTE networks [7]. They used the collected data using the QXDM toolset and developed LinkForcast, which is a machine learning based framework for throughput prediction. The framework uses lower-layer information to predict instantaneous link bandwidth. Samba et. al., also performed throughput prediction using the random forest algorithm [8]. For throughput prediction, they relied on additional data from network operators along with the data that they collected using a crowdsourcing approach. The additional data contained radio access network measurements such as the average cell throughput, average number of users in each cell and the connection success rate. The authors concluded that these additional measurements improved the throughput prediction accuracy.

In this paper, an overview of the used dataset is provided followed by our proposed methodology for throughput prediction. We will then demonstrate and discuss the experimental results giving a comparative analysis of the different throughput modelling and prediction.

II. Dataset

The dataset used in our models was collected along a public transit bus in Kingston, Ontario and is publicly available [9]. It consists of 30 repeated bus trips, where each trip lasts almost an hour. The trips were carried out at three different times of the weekday, namely 9 am, 12 pm and 6 pm. The route of the bus is shown in Fig. 1.

The dataset contains multiple network parameters, such as the reference signal received power (RSRP), reference signal received quality (RSRQ), received signal strength indicator (RSSI), signal-to-noise ratio (SNR), downlink and uplink throughput. In addition to network parameters, the dataset contains context information such as the GPS coordinates and speed of the bus. The dataset has a one-second granularity, meaning that measurements are logged every second.

Fig. 1: The public transit bus route in Kingston, Ontario.

III. Data Preprocessing

Raw data is often noisy and incomplete; therefore, machine learning models cannot be deployed directly on raw data. To ensure the accuracy and efficiency of the machine learning models, some preprocessing steps were performed before the data was fed to the models.

A. Outlier Detection and Removal

The first preprocessing step performed in this research was outlier detection and removal. Outlier detection, also known as anomaly detection, is the process of finding data points that significantly deviate from the rest of the data. The presence of outliers may affect the performance of the machine learning model, as the quality of the data determines the quality of the prediction model. Therefore, it is often desirable to detect and remove outliers before the data is passed to the model.

The z-score method was used to detect outliers in the data. The z-score measures the multiples of standard deviation above or below the mean a data point is, giving indication of how far from the mean a data point is located [10]. The z-score of a data point can be calculated using the following equation:

$$z = \frac{(x-\mu)}{\sigma} \quad (1)$$

where x is a data point, μ is the mean of all data points and σ is the standard deviation of all data points.

B. Missing Values Imputation

The second preprocessing step was missing values imputation. Missing data are one of the most common sources of error in any code, and the machine learning models do not work when the input data contains missing values. To avoid such errors, imputation with mean/median was done by replacing the missing values in each column of the data with the mean/median of the other values in that column.

C. Feature Scaling

Feature scaling is often required in machine learning when the data features have different ranges, especially when the tested models rely on the distance between the features. In that case, features with a larger range would influence the result more than those with a smaller range. As a preprocessing step, z-score normalization, also called standardization, was performed. Z-score normalization is a scaling technique that transforms the distribution of the data, causing it to have a mean of 0 and a standard deviation of 1. In this technique, each data point is replaced by its z-score. As mentioned earlier, the z-score can be computed using (1).

D. Data Binning

As mentioned before, raw data tend to be noisy. Data binning or discretization is a preprocessing technique that reduces the effect of noisy data [11]. This technique works by first sorting the data, dividing the values into bins with smaller intervals, and then replacing the values with a more general value calculated for each bin, such as the mean or median. Data binning has a smoothing effect on the data and causes the model to have better generalization power. In this work, binning was performed by aggregating data for every five seconds. Moreover, smoothing using the bin mean was used to get the mean of each data attribute during the five seconds interval.

IV. Throughput Prediction

Prediction techniques are one of the proactive means of optimizing network resources. By analyzing previous fluctuations in the network parameters, the prediction model can learn the behavior of the network and be able to anticipate the fluctuations before they occur. This can lead to efficient

resource allocation, power saving and anomaly detection. Similarly, network throughput prediction can boost the performance of several applications, such as content prebuffering and predictive resource allocation [12]. For example, a user approaching an area with a poor throughput can have content prebuffered before reaching that area.

Two approaches are commonly used for prediction. The first approach is the regression predictive modelling using classical machine learning techniques. This approach predicts the instantaneous value of the target variable based on the values of the independent variables. For throughput prediction, the approach relies on the fact that the network throughput depends on various factors, such as network quality parameters and context information. The approach also takes advantage of the fact that throughput has a strong correlation with the signal strength. Accordingly, the instantaneous throughput values are predicted based on current network parameters and context information.

The second approach is time series forecasting. A time series is a collection of observations taken at consecutive points in time. The aim of time series forecasting is to build a model to estimate future values of a time series based on the previous values. Unlike regression predictive modelling, a time series adds an explicit order dependence between the observations. In this approach, future throughput values are estimated based on past throughput values. Therefore, the approach does not require any knowledge about network parameters or context information. As this approach allows for future throughput measurements estimation, it can significantly help in planning future scheduling decisions and predictive resource allocation.

The classical machine learning approach requires proper feature selection methods to produce accurate predictions. Feature selection is the process of selecting the features that contribute most to the prediction variable. It is an important step in the machine learning pipeline as training a model with irrelevant features could decrease the accuracy of the model and result in erroneous predictions. Moreover, training the model with fewer attributes reduces the complexity of the model, and makes the model simpler and easier to understand.

A common feature selection technique is analyzing the feature importance property of the model, which computes the importance of each feature in the dataset. The idea of feature importance is to calculate a certain score for each feature in the dataset, where higher scores indicate that this feature contributes more toward the prediction of the target variable. The random forest model measures the feature importance based on the mean decrease in impurity. Features at the top of the trees influence the prediction decision of a larger fraction of the data samples. As a result, these features have higher feature importance scores than the features at the bottom of the trees.

The feature importance property of the random forest model ranked the most important features for downlink throughput prediction as follows: RSRP, RSSI, SNR, RSRQ, timestamp, longitude, and latitude. As a result, only these variables were used as input to the machine learning models and the remaining data variables were disregarded.

A. Classical Machine Learning Models

Several machine learning models were trained and tested on the data to determine the best one for throughput prediction.

1) K-Nearest Neighbor (KNN) for Regression

The KNN is a supervised learning algorithm that is commonly used in machine learning tasks for classification and regression due to its simplicity and applicability in various real-world problems. The algorithm predicts the target value based on the similarity between different points. To find similar points, the algorithm uses a distance measure such as Euclidean distance, Hamming distance, Manhattan distance or Minkowski distance.

The KNN algorithm works as follows. For every input point, P_i , it finds the nearest k neighboring points of P_i using one of the distance measures mentioned above, and then predicts the output of P_i based on the value of its nearest neighbors. In the case of regression, the output target value of P_i would be the mean of its nearest k neighbors. The KNN algorithm has one hyperparameter, which is the number of neighbors k.

2) Support Vector Machine Regression (SVR)

SVR is a variant of the support vector machine (SVM) algorithm used for regression problems. SVR uses a technique called the kernel trick to map the input into a higher dimensional feature space and then constructs a linear regression model in this higher dimensional space [13].

Some definitions should be clarified before explaining how SVR works. A hyperplane is the line that fits the data, while the boundary lines bound the points that are considered for prediction. The support vectors are points that are closest to the boundary and can even lie on it. The distance between the hyperplane and the boundary line is denote by ε. The goal is to find the optimal value of ε so that the support vectors lie within that boundary line.

The linear function is given by:

$$\boldsymbol{y = wx + b} \quad \textbf{(2)}$$

where y is the target variable, x is the input variable, w is the weight coefficient, and b is the bias.

The optimization function in SVR works to maximize the margin by minimizing the squared sum of the weight coefficients w to ensure the function is as flat as possible, and it is given by the following formula:

$$\boldsymbol{\frac{1}{2}|w|^2} \quad \textbf{(3)}$$

The constraint that all the residuals are less than ε is ensured by the following rule:

$$\boldsymbol{\forall_n: |y_n - (wx_n + b)| \leq \varepsilon} \quad \textbf{(4)}$$

The hyperparameters used in SVR training are the kernel function, $\boldsymbol{\varepsilon}$, and the regularization parameter $\boldsymbol{C}$ that penalizes misclassification and margin errors.

3) Ridge Regression

Ridge regression is a variant of linear regression, which is a machine learning algorithm that performs regression analysis. In linear regression, the model finds a linear relationship between the dependent and independent variables. A simple regression line can be modelled by the following equation:

$$\boldsymbol{y = B_0 + B_1 x} \quad \textbf{(5)}$$

where x is the input training data and y is the target variable. During training, the model tries to find the best-fit line to predict the value of y for a given value of x, by finding the optimal values for the coefficients B_0 and B_1.

Ridge regression is a form of linear regression that performs an L2 regularization to prevent overfitting. L2 regularization is given by:

$$\boldsymbol{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^P} \parallel \boldsymbol{y} - \boldsymbol{X\beta} \parallel_2^2 + \boldsymbol{\lambda} \parallel \boldsymbol{\beta} \parallel_1 \qquad (6)$$

where β is the coefficient vector and λ is the shrinkage hyperparameter. A larger value of λ reduces the model complexity and prevents overfitting. However, after a certain point, increasing the value of λ may lead to underfitting.

4) Random Forest for Regression

Random Forest is a type of ensemble learning, where a group of weak models are combined to form a strong model [14]. It is a supervised learning algorithm that is constructed through an aggregation of decision trees, where each tree is trained on a subset taken from the data. Unlike linear models, decision trees are capable of mapping nonlinear relationships within the data attributes. Decision trees use a tree-like graph to formulate rules and make predictions based on these rules. In decision trees, each node represents a feature, each branch represents a decision, and each leaf represents an output. The goal is to create a tree for all the features in the dataset and use it to produce a different output at each leaf. The output that is produced depends on the set of decisions made by the tree as it processes the input feature vector.

The random forest algorithm works by randomly picking a number of sub-samples from the data samples to build each tree. Then, several features are selected randomly from all the features to ensure that the trees are not highly correlated. To make a prediction, each tree makes a vote by predicting the target value. The forest then takes the average of all the votes by the different trees in the forest.

Hyperparameters in a random forest include the number of decision trees in the forest and the maximum number of features considered by each tree for splitting a node.

B. Time Series Forecasting

Various techniques and mechanisms have been proposed for time series forecasting. In this work, Autoregressive Integrated Moving Average (ARIMA) and Long Short-Term Memory (LSTM) models were used [15].

1) ARIMA

An ARIMA model is a class of statistical models and is a commonly used technique for time series forecasting. Autoregressive models use a linear combination of the current observation and several lagged observations to forecast future values of a variable. On the other hand, moving average models use a linear combination of the residual errors to predict the forecast error at the next time step. ARIMA models combine both approaches into one model to perform time series forecasting [16].

ARIMA models require the time series to be stationary. A stationary time series is one whose statistical properties are not dependent on the time at which the series is observed [16]. One approach that could convert a non-stationary time series to a stationary one is to compute the differences between successive observations. This approach is called differencing and is an important step in time series forecasting using ARIMA models. The ARIMA model is given by:

$$\hat{y}_t = \boldsymbol{\mu} + \boldsymbol{\phi_1 y_{t-1}} + \cdots + \boldsymbol{\phi_p y_{t-p}} - \boldsymbol{\theta_1 e_{t-1}} - \cdots - \boldsymbol{\theta_q e_{t-q}} \qquad (7)$$

where μ is a constant, $y_{t-1} \ldots y_{t-p}$ are lags of the series, $e_{t-1} \ldots e_{t-q}$ are the error terms, $\phi_1 \ldots \phi_p$ are the coefficients of the autoregressive model and $\theta_1 \ldots \theta_q$ are the coefficients of the moving average model.

ARIMA models have three parameters, namely p, d, and q, where p is the number of lag observations, d is the degree of differencing, and q is the size of the moving average window, also called the order of the moving average.

2) LSTM

LSTM is a type of recurrent neural networks (RNNs) [17] that is used in the field of deep learning. Unlike feedforward neural networks, recurrent neural networks have feedback connections. These feedback connections make the recurrent neural networks capable of handling sequence dependencies. LSTM networks use special units in addition to the standard units of the RNN networks in order to capture long term temporal dependencies.

The architecture of LSTM networks includes a memory cell, which is used to maintain information for long periods of time. The memory cell consists of four gates: forget gate, input gate, state update gate, and output gate. The input and state update gates are responsible of feeding information into the memory cell, while the forget gate is used for resetting the memory cell. The output gate determines how the output of the memory cell affects other LSTM cells [18]. Fig. 2 shows the LSTM structure, where i_t is the input gate, f_t is the forget gate, o_t is the output gate, σ and tanh correspond to the activation function, c_t is the state of the memory cell and h_t is the output of the cell.

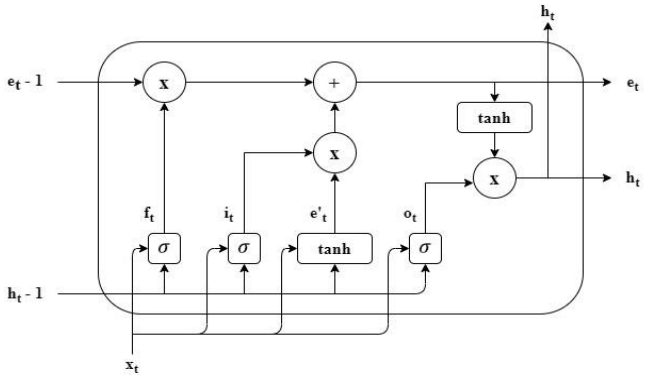

Fig. 2: LSTM cell structure.

LSTM models can be used for time series modelling and forecasting. For LSTM models to produce accurate predictions, they require very large amounts of data.

C. Evaluation Metrics

The performance of the models was evaluated using two well-known metrics: the R^2 score and the root mean square error (RMSE). In the following subsections, the two metrics are described.

1) R^2 score

Also called the coefficient of determination, the R^2 score is a goodness-of-fit measure for regression models. It indicates the percentage of the variance in the dependent variable that can be explained by the independent variables [19]. The R^2 score measures the strength of the relationship between the model and the dependent variable on a convenient 0 – 1 scale. The R^2 score is computed by:

$$\boldsymbol{R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}} \quad \textbf{(8)}$$

where y is the actual value, $\hat{y}$ is the predicted value and $\bar{y}$ is the mean of all y values. The R^2 score has a range from 0-1, where 0 indicates that the model does not explain any of the variability of the response data around its mean, and 1 indicates that the model explains all the variability of the response data around its mean. Accordingly, a higher R^2 score specifies that the model fits the data better.

2) Root Mean Square Error (RMSE)

The RMSE is the standard deviation of the prediction errors, also called the residuals [16]. The residuals show the difference between the actual data values and values predicted by the model. The RMSE is used to measure how the residuals are dispersed. Moreover, it can be used to compare the prediction errors of different models to determine the model with the highest performance on the data. The RMSE can be calculated by the following equation:

$$\boldsymbol{RMSE = \sqrt{\frac{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}{N}}} \quad \textbf{(9)}$$

where $\hat{y}$ corresponds to the model's predictions, $\bar{y}$ corresponds to the actual values of the data samples, and N is the number of samples. Lower RMSE values indicate that the model fits the data better.

The RMSE range depends on the range of the predicted variable. In the dataset used in this work, downlink throughput measurements range from 0 to 80000 Kbps.

V. Results and Discussion

To effectively evaluate the performance of the machine learning models, cross-validation was used to split the dataset into 70 % for training and 30% for testing and validation.

For preprocessing the data and training the machine learning models, an ASUS computer with an INTEL® CORE™ i5-7200U Processor and a 4 GB RAM was used. Python 3 was used on a Linux operating system. Scikit-Learn [20], Numpy [21], Pandas [22] and Matplotlib [23] libraries in Python comprise the machine learning framework.

To compare between the different models, the models' performance was evaluated using the two-evaluation metrics mentioned earlier, the R^2 score and the RMSE. Table 1 shows the R^2 score and the RMSE value of each of the machine learning models.

As shown in the table, the random forest achieved the best performance with the highest R^2 score of around 0.78 and the least RMSE value of approximately 8200. Fig. 3 demonstrates the performance of the random forest model on the test set, showing the actual as well as the predicted throughput values. The reason for this high performance is that random forests have a strong generalization capability that prevents overfitting and improves the accuracy of the model. By choosing a few random sub-samples to build each tree and selecting a random set of features, the random forest algorithm decreases the correlation between the different trees and greatly reduces the variance in the predictions made by the model.

TABLE I
MACHINE LEARNING MODELS' COMPARISON.

Model	R^2	RMSE
SVR	0.36 ± 0.03	14000 ± 300
KNN	0.38 ± 0.02	13700 ± 300
Ridge Regression	0.71 ± 0.02	9300 ± 200
Random Forest for Regression	0.78 ± 0.01	8200 ± 200

Moreover, the performance of the ridge regression model was close to that of the random forest, with an R^2 of around 0.71 and a RMSE value of around 9300. On the other hand, the performances of the KNN and the SVR models were not satisfactory. The KNN model had an R^2 score of around 0.38 and a RMSE value of around 13700, while the SVR model had the lowest performance, with an R^2 score of around 0.36 and a RMSE value of around 14000.

Table 2 compares the performance of the time series forecasting models based on the evaluation metrics mentioned before. One can see from the table that the ARIMA model achieved a better performance than the LSTM model. The performance of the ARIMA model on the test data is displayed in Fig. 4. The poor results of the LSTM model are attributed to the fact that the LSTM networks require a large amount of data in order to have an adequate performance. This is mainly because LSTMs have various units that require large number of weights to be trained.

We believe that a higher throughput prediction performance could have been achieved if we had access to additional data from network operators as in the work of [2]. Training the models on data from network operators along with our client-side data would significantly improve the prediction performance. In addition, having a dataset with a higher granularity as in the related work in [7] would improve the accuracy of the data and therefore lead to a higher prediction performance.

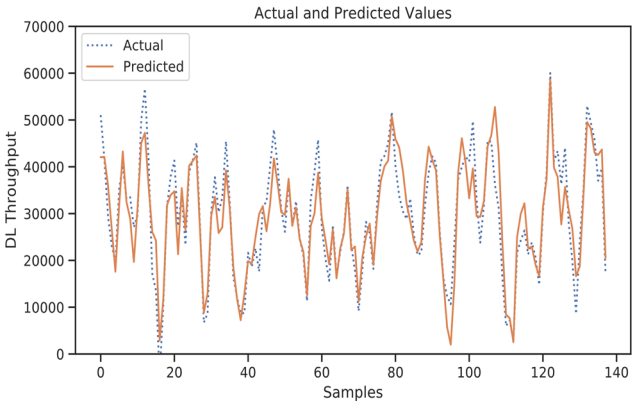

Fig. 3: Random forest model performance on test set.

TABLE II
TIME SERIES FORECASTING MODELS' COMPARISON.

Model	R^2	RMSE
ARIMA	0.62	10400
LSTM	0.59	10800

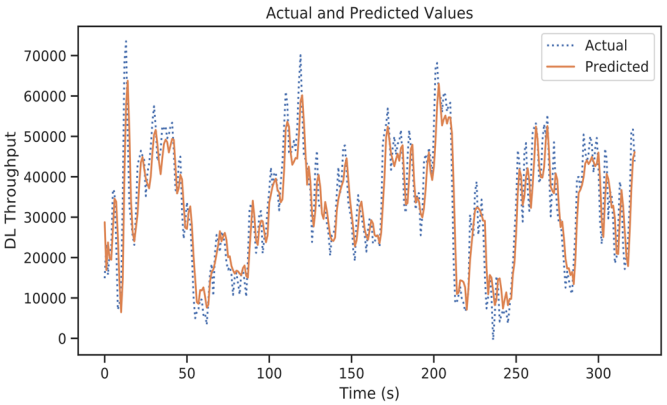

Fig. 4: ARIMA model performance on test set.

VI. Conclusion and Future Work

In this paper, throughput prediction techniques were applied to 4G LTE network data. We explored the performance benefits of various machine learning models for throughput prediction, namely SVR, KNN for regression, ridge regression, and random forest for regression. Moreover, time series forecasting models were deployed for forecasting future throughput values. The merits and limitations of these different modeling methods were analyzed in addition to a comparative analysis using different evaluation metrics in order to determine the optimal one for throughput prediction. The highest prediction accuracy was achieved using the random forest model since the model added an additional layer of randomness to the features, which greatly reduced the variance and assisted the model to have a significant generalization capability. The additional layer of randomness has also resulted in a higher model performance on unseen data and prevented overfitting.

For future work, throughput prediction techniques could be investigated on data with higher granularity. Furthermore, acquiring additional data from network operators with information about the average cell throughput and average number of users per cell could significantly improve the performance of the throughput prediction models.

References

[1] H. Abou-zeid, H.S. Hassanein, and S. Valentin, "Energy-efficient adaptive video transmission: exploiting rate predictions in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no 5, pp. 2013-2026, Jun. 2014

[2] T. Kamakaris and J. V. Nickerson, "Connectivity maps: Measurements and applications," *in Proc. of the 38th Annual Hawaii International Conf. on System Sciences,* Big Island, HI, USA, pp. 307–307, 2005.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] T. Pögel, and L. Wolf, "Prediction of 3G network characteristics for adaptive vehicular connectivity maps (poster)," in *Vehicular Networking Conf. (VNC), IEEE,* pp. 121–128, 2012.

[4] T. Pögel, and L. Wolf, "Optimization of vehicular applications and communication properties with connectivity maps," in *Local Computer Networks Conference Workshops (LCN Workshops)*, IEEE 40th, pp. 870–877, 2015.

[5] Q. Xu, S. Mehrotra, Z. Mao, and J. Li, "PROTEUS: network performance forecast for real-time, interactive mobile applications," in *Proc. of the 11th Annual International Conf. on Mobile Systems, Applications, and Services*, pp. 347–360, 2013..

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

[7] R. Jin, *Enhancing upper-level performance from below: Performance measurement and optimization in LTE networks.* Doctoral Dissertations, Univ. of Connecticut, United States, 2015.

[8] A. Samba, Y. Busnel, A. Blanc, P. Dooze, G. Simon, "Instantaneous Throughput Prediction in Cellular Networks: Which Information Is Needed?" in *IFIP/IEEE International Symposium on Integrated Network Management (IM),* Lisbonne, Portugal, 2017.

[9] H. Elsherbiny, A. M. Nagib, and H. S. Hassanein, "4G LTE User Equipment Measurements along Kingston Transit 502 Bus Route," 2020. [Online]. https://doi.org/10.5683/SP2/EQWKO1

[10] E. Kreyszig, *Advanced Engineering Mathematics*, 4th ed. Wiley. pp. 880-881, 1979.

[11] S. Cannistra, "Small explanation of binning in image processing". [Online]. Available: http://www.starrywonders.com/binning.html. [Accessed Feb. 10, 2020].

[12] H. Abou-zeid, H.S. Hassanein, and S. Valentin, "Optimal Predictive Resource Allocation: Exploiting Mobility Patterns and Radio Maps," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, pp. 4877–4882, 2013.

[13] H. Drucker, C. C. Burges, L. Kaufman, A. J. Smola, and V. N. Vapnik, "Support Vector Regression Machines," in *Advances in Neural Information Processing Systems 9*, pp.155–161, 1996.

[14] H. TK, "The Random Subspace Method for Constructing Decision Forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no.8, pp. 832–844, 1998.

[15] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp.1735–1780, 1997.M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[16] R. J. Hyndman and G. Athanasopoulos, *Forecasting: Principles and practice*. Vic. Heathmont: OTexts, 2014.

[17] S. Dupond, "A thorough review on the current advance of neural network structures," *Annual Reviews in Control*, vol. 14, pp. 200–230, 2019.

[18] L. M. Camarinha-Matos, R. Almeida and J. OliveiraI (eds.) "Technological Innovation for Industry and Service Systems," *10th IFIP WG 5.5/SOCOLNET Advanced Doctoral Conference on Computing,* Electrical and Industrial Systems, DoCEIS, Costa de Caparica, Portugal, 2019.

[19] R. G. D. Steel and J. H. Torrie, *Principles and Procedures of Statistics with Special Reference to the Biological Sciences*. McGraw Hill, 1960.

[20] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion and O. Grisel, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, pp. 2825–2830, 2011.

[21] T. E. Oliphant, *A guide to NumPy (Vol. 1).* Trelgol Publishing, USA, 2006.

[22] W. McKinney, "Data structures for statistical computing in python," in *Proc. of the 9th Python in Science Conf.,* vol. 445, pp. 51–56., 2010.

[23] J. D. Hunter, "Matplotlib: A 2D graphics environment," in *Computing in Science & Engineering*, vol. 9, no.3, pp. 90–95, May-June 2007.