Study Area Recommendation via Network Log Analytics

Anupama Upadhayula[1], Avinash Ravilla[1], Ishwarya Varadarajan[1],
Sowmya Viswanathan[1] and David C. Anastasiu[1,*]

***Abstract*— With the current surge in the number of devices connected to the Internet all over the world, usage of network bandwidth has also seen a huge increase. This has led to network traffic congestion and slowdowns. It has become important to build a model which would anticipate network usage in the future and help us prepare to provide any necessary infrastructure or service support crises. In this paper, we propose a model that uses machine learning techniques to analyze network logs of access points installed throughout the campus of San José State University (SJSU) to predict its network usage at a future point in time. We also propose to build a classification model to classify browsed data of users into the study and leisure classes based on content. Hence, the final model will be able to predict network throughput of access points and the type of activity they will be used for. We utilize these models in a novel mobile phone application we designed that helps students choose an appropriate place to study or hang out at a chosen time in the near future. Our application combines the use of mobile phone sensors for localization with the power of machine learning models executed in the cloud to provide a seamless user experience for students and faculty alike.**

***Index Terms*— mobile phone, machine learning, cloud, network log analytics, classification, regression.**

I. Introduction

Wireless communication is one of the most successful means of communication. Every organization is adopting this form to address network demands. Network congestion is one of the major problems in areas such as security and network management. It leads to loss of packets resulting in poor quality of service. Performance degrades, when the load is not anticipated and necessary measures are not deployed to handle sudden crises. An algorithm that can analyze past data and find patterns in them to anticipate future events, can be used to prepare ahead of time and deal with unexpected situations. This is where machine learning and deep learning techniques come into the picture. Mining network data comes with several challenges. Many factors, such as large amounts of data, network speed, varied dynamic network topology, resource constraints, and others, need to be considered while performing data mining. Most of the studies adopt time series modeling to analyze the network data in order to get a meaningful value out of the data. Time series models consider data over an interval of time. There are different types of time series models, such as autoregressive moving average (ARMA), autoregressive integrated moving average (ARIMA), vector autoregressive moving-average (VARMA), and seasonal autoregressive integrated moving-average (SARIMA), to name just a few. Based on the type of time series models, different machine learning algorithms can be applied on the data. A smart data mining technique that considers all of the above mentioned challenges and factors is vital to obtaining correct predictions for the future.

There are four different techniques that can be considered while mining network data – frequent pattern mining, sequential pattern mining, classification and clustering. There have been many studies which explored the use of time series as a general network traffic prediction model. Data is recorded over hours, days, weeks, months, years, etc. Models such as SARIMA, and back propagation neural network (BPNN) have been used and compared for analyzing such data. Results have shown that neural network models comparatively give better prediction accuracy. Several linear and non linear models can also be adopted to predict the rate of traffic. Some linear models, such as Holt Winters model, and the ARMA model are combined with clustering techniques to yield better results. Prediction models can be built to mine the network data or to forecast the network data by enhancing the existing models.

It is difficult to find an access point (AP) which is highly available and suitable for a user's nature of work at SJSU. Existing solutions revolve around improving the range of APs and using better routers. Our solution is to design an algorithm that considers the usage of various APs located on the SJSU campus and predicts a better AP location for a user. Network data is collected from the APs across the university on a daily basis. More than 15 GB of the data consisted of DNS data spread over 283 APs equipped with wireless sensors. The algorithm predicts the throughput of an AP on campus. It further identifies a location as 'work' or as 'leisure' based on the type of activity the users connected to the AP engage in.

Prediction of an AP with less flowing traffic accurately is largely dependent on the algorithm and the pre-processing steps that are applied on the data. The traffic is also categorized into two categories - study and leisure based on the activities of the user. The data is extremely large and hence, analysis of the data is a complex task. The baseline algorithms that have been considered for the project are ARIMA, Facebook Prophet and Recurrent Neural Network Long Short Term Model (RNN-LSTM). The mobile and web based application puts the best of the above models to use and delivers a recommendation where the student user can find a study area.

[1]Computer Engineering, San José State University, San José, CA
*Corresponding Author, david.anastasiu at sjsu.edu

II. Related Work

A. Network traffic Classification

The network architecture has become complicated with the rapid development of Internet and network services. Classification of network traffic has become one of the most important tasks for network administrators these days. Shafiq et al. [1] discuss various types of network traffic classification methods. They discuss various machine learning algorithms to classify the network data. It is challenging to make real time decisions based on the networks available, the amount of communication between them, or the threshold limits of each of those devices. Such resource constrained computing and resource retention prompts us to analyze the network data. Two of the main challenges faced while designing any classification algorithm is its accuracy and the computation time. Miao et al. [2] have provided a comprehensive discussion about classification of network data. They showed that random forest and neural networks are two of the top machine learning algorithms that gave good classification results. These models can help us understand the bottlenecks and avoid congestion in the network and be able to design a better prediction model of network traffic.

B. Traffic Prediction

Traffic control center manages and collects huge amount of data regarding the traffic, network usage and other information. Sometimes, a prediction model is also required, which will provide information on when the network is busy and when the network load is considerably lower. Prangchumpol [3] proposed an unconventional model which uses association rule to find a relationship between the network traffic, semester and time. It predicted the network traffic for the next day in the semester which was useful in network routing and improved the network performance. Wen and Lee [4] propose a hybrid grey-based recurrent neural network (G-RNN) which helped in traffic prediction. This technique has been proven to be accurate when the data are random, and had spatial and time series properties. The grey preprocessing approach is used to decrease the randomness in the data. Traffic prediction is also useful when trying to forecast congestion in networks.

Aldhyani and Joshi [5] proposed an integrated model of different time series models with soft clustering techniques. They hypothesized that linear time series models like Holt Winters, exponential smoothing, ARMA, and autoregressive neural network, when integrated with a clustering approach, provide better results in network traffic forecasting. Eterovic et al. [6] compare the performance of ARIMA and artificial neural network (ANN) models. ARIMA model assumes the network traffic to be stationary in time. They showed that ARIMA worked for short term predictions but failed for long term predictions. ANN gives better prediction results in this study when taking long term forecasts into consideration. Feng and Shu [7] also discussed about ARIMA and ANN models and conclude that ANN has better prediction accuracy than ARIMA. ARIMA is proved again not to be useful for long range dependent predictions. Ang et al. [8] employed long short-term memory (LSTM) recurrent neural network (RNN) to analyze the traffic flow. They conclude that traffic flow together with speed as inputs to the model, yields good results. LSTM is a machine learning technique which analyzes and learns information from data which spans over a long period of time. It is very useful for time series prediction. Zhuo et al. [9] better accuracy is obtained when using LSTM for predictions in large granularity data set.

III. System Architecture

In this section, we describe the architecture of the system we have devised, which can also be viewed in Fig. 1.

A. Front-end System

We developed a web based and a mobile based application for users to view the predicted average throughput and type of activities that are performed using a particular AP.

B. Back-end System

The back-end system of our application consists of Random Forest, a classification model to categorize APs as being used for work and entertainment classes, and RNN-LSTM, a time-series model to predict the availability of APs.

a) Long Short Term Memory Model(LSTM): Recurrent Neural Network (RNN) is one of the deep learning models, extensively used in time-series analysis. When an RNN contains LSTM units as its cells, it is called an LSTM neural network. RNNs can predict the output by learning the most recent data from the past, but they do not have the capability to remember data further back in time. An LSTM network is capable of remembering data for longer periods and predict the output in the future. Hence, LSTM networks are more reliable for problems with long-term dependencies.

In our work, we used the same LSTM model architecture as that in Olah [10], which consists of two neural network layers with each layer containing 4 LSTM units, for each time step. Each LSTM cell has 20 units which learn data and predict the output for the next time step. The LSTM predicts average throughput for the next 4 hours from the current hour by learning hourly throughput data from the past month. The model is trained with different parameters to yield the best result and is evaluated using Mean Absolute Error (MAE).

b) Random forest: Random forest is a supervised classification algorithm which belongs to the family of ensemble algorithms. Random forest is a collection of decision tress formed from randomly split input data. use an important and powerful technique called bootstrap which uses descriptive statistics. The predicted results are highly dependent on the number of trees used. The tree formulates rules using Gini

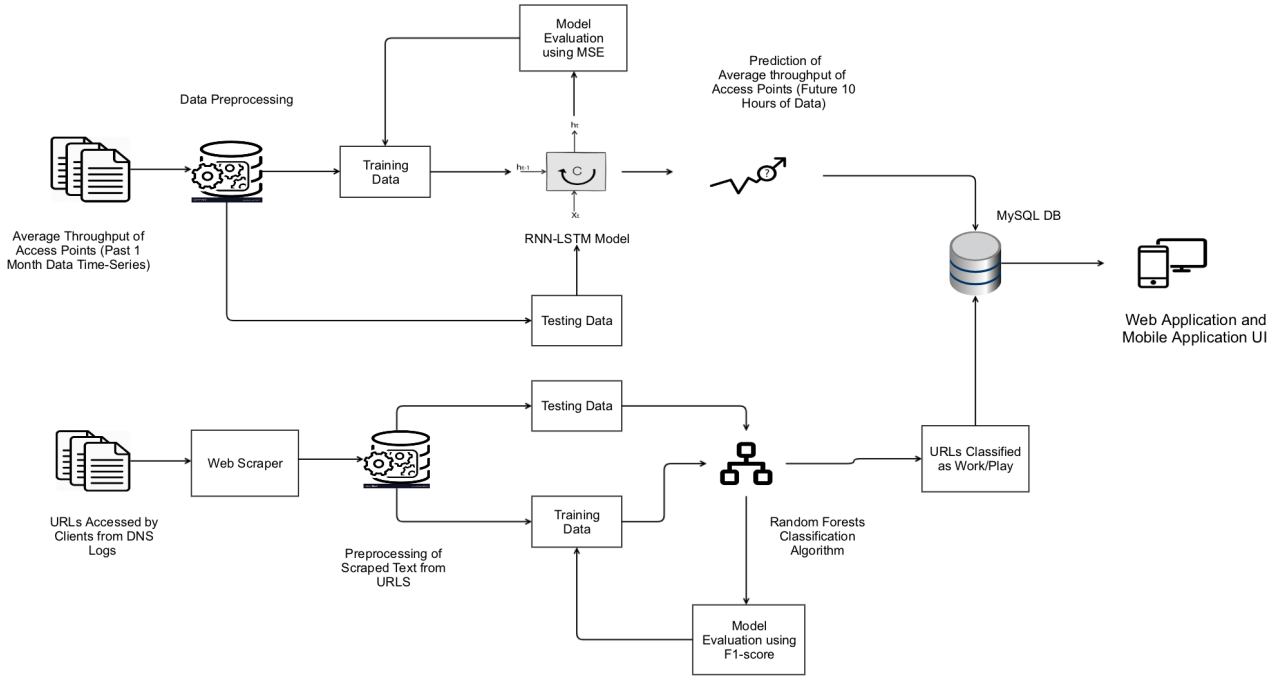

Fig. 1. System architecture.

index when the input is provided to the algorithm and the results are based on these rules.

IV. Project Design

The paper deals with two problems and proposes to solve them using machine learning. The first proposal is to predict the average throughput (Kbps) using time-series models. The second proposal is to categorize APs into work and entertainment classes using classification models.

A. System Design

1) Regression Problem: This problem involves analyzing time-series data containing average throughput from the past month for each AP in the University and predicting the throughput for the next 4 hours from the current hour. Three time-series models were used and their predictions were evaluated using Mean Absolute Error (MAE). The models will be trained and tested with different sets of parameters. On the basis of evaluation, the most efficient model's predictions will be displayed on the front-end applications.

2) Classification Problem: The classification problem involves using three classification models to categorize network access data to multiple categories. The models' efficiencies will be evaluated using F1 score to determine the most efficient model to be used as part of our application. The input data to the models contain texts scraped from corresponding URLs accessed by the users of university's network using a web scraper. The scraped text of each URL is manually labeled as work or entertainment to aide the learning of supervised classification algorithms. The models are trained and tested to classify new URLs. On the basis of evaluation, most efficient model's results will be used to display data on the front-end applications.

B. Client Design

We will develop a Web application and a mobile app for the user to check the availability of the AP. The user interfaces of the applications prompt the user to enter a preferred date and time. Once the request is submitted, the page will display the APs at respective locations in the engineering and MLK library using Google maps. The mobile app can use the phone's GPS sensor, if available, to localize the user within the campus map. APs are denoted using markers on the maps. When the users click on the marker, they will be able to see the details of the corresponding AP, such as its name, the predicted throughput for the selected hour, and the type of activity for which the AP is being used.

V. Methods

A. Classification

We used supervised classification models to categorize the nature of work each AP is being used to two categories namely entertainment and education. This section explains the different algorithms we used for the classification problem and their results.

1) Models: A brief description to the classification models are given below.

a) Random forest: Random forest is one of the supervised classification models that belongs to the class of ensemble algorithms. An ensemble method is a technique that combines the result from different machine learning algorithms to produce more accurate results. In the case of random forest, different random subsets of features are used to train decision tree classification models, and the final classification result is given by aggregating the predictions of the individual tree models for a given test sample.

b) Gaussian naïve Bayes: Naïve Bayes is a collection of classification algorithms that are purely based on Bayes theorem. It is a linear classifier. The common principle shared by this collection of algorithms is the strong independence between the features. Gaussian naïve Bayes is one of these algorithms. In Gaussian process, the random variables are normally distributed [11].

c) XGBoost classifier: XGBoost, like random forest, belongs to the family of ensemble algorithms [12]. It combines the learning rate of multiple algorithms. The result is an aggregated value of several algorithms.

2) Implementation:

a) Data Tier: We used the network access logs which contains the following features: Date, client IP, client ID, URL, query type, query DNS IP, answer code, answer DNS IP, record type, and error.

b) Data Preprocessing: We considered the columns Date, client IP, URL and Record Type from network log access data as the source of input to our application.

The column URL contains the DNS address of websites browsed by the users' of the university's network. Duplicate URLs are removed and a web scraper script is used to scrape texts of the corresponding websites. The scraped text is classified to categories; entertainment and education. The web scraper directly assigns classes to URLs based on words in the DNS address.

We manually labeled 12,358 scraped texts to prepare the training data for the classification algorithms. We removed stop words (commonly used words such as 'the', 'in', 'a', 'an', 'for' etc.) and retained the base form of words, also called as lemmatization. The texts are converted to sparse vectors using a count vectorizer in a format readable by machine learning algorithms.

Directly categorized URLs and those categorized using classification models are uploaded to a MYSQL database table.

3) Evaluation Methodology:

a) Holdout method: The manually classified data is split into two datasets; training and testing. The algorithms were trained with 75 percent of the manually classified data and the algorithm was tested on remaining 25 percent of the data.

b) Training: Training involves the process of providing an error free dataset for the algorithm to learn. The results of a classification algorithm depend on how well it is trained. We used different values for the meta parameters for training each algorithm. For random forest, we used different values for number of trees or estimators through which it formulates the rules. Since the data is huge, lesser number of trees yielded results with very low accuracy. When the number of trees was increased to 1000, it produced considerably more accurate results. Increasing the number of trees above 1000 did not have much impact on reducing the error. We tested both the Gini index and entropy as criteria. We found that the Gini index produced better results when compared to entropy.

c) Cross validation: The holdout method for the sparse dataset can produce high error rate as it is has a single train and test datasets. In the hold out method, there are high chances where an algorithm might break and produce bad results. Therefore, we considered k-fold cross validation where the data are split into k sections. Each section, in turn, is used for testing a model that is trained using data from all the other sections. Results from each tested fold are then averaged to produce the final F1 score for the experiment. In our experiments, we used a value of 4 for k, i.e., in each experiment we used 25% of the data for testing and 75% for training, and the final result is the average of the results for each of the 4 test folds.

d) Evaluation Metric: As suggested by P. Tao [13], we considered weighted F1 score as a metric for evaluating the efficiency of the classification algorithms. F1 score conveys a balance between precision and recall. Since, the data had imbalanced distribution of classes, we calculated F1 scores for the three models to compare the performance of each model and determine the model with more accurate results.

4) Evaluation Results: The following are the F1 scores of the three algorithms: The F1 score of random forest classifier is 0.7184, GaussianNB's F1 score is 0.6733 and that of XGBoost algorithm is 0.565. Since random forest classifier model has the highest value of F1 score, it is more accurate than the other models. The results of random forest and XGBoost models with different parameters are provided in Tables I and II.

TABLE I
RANDOM FOREST RESULTS FOR DIFFERENT PARAMETERS

Estimators	Criterion	F1 Score
100	Entropy	0.592
500	Entropy	0.621
500	Gini	0.677
1000	Gini	0.718

TABLE II
XGBOOST RESULTS FOR DIFFERENT PARAMETERS

Learning rate	Max Depth	Estimators	F1 Score
0.1	2	100	0.43
0.1	4	500	0.49
0.3	5	1000	0.56

B. Regression

1) Implementation:

a) Data Tier: Data has been sourced from the DHCP log of the University's network. It has the following features: Client User name, Client IP Address, Client MAC Address, Association Time, Vendor, AP Name, Radio Type, Device Name, Map Location, SSID, Profile, VLAN ID, Protocol, Session Duration, Policy Type, Avg. Session Throughput (Kbps). We used Avg. Session Throughput (Kbps) for prediction of results using the regression models.

b) Data Preprocessing: One of the fields from the log, average throughput (Kbps), has been aggregated on an hourly basis of the day, for each AP. So, ideally every AP will have 24 sets of values of average throughput, corresponding to each hour in a day for a month. The data is then split into train and test sets, and the model then makes the predictions for the test data.

2) Models: We used three machine learning models in our project, and their performances are compared based on their MAE scores to find the most effective algorithm suited for solving the regression problem.

a) ARIMA: The Auto Regressive Integrated Moving Average (ARIMA) method is very useful for analyzing and predicting time series data. Its input is a univariate time series. The acronym can be described as: AR: Auto-regression - Dependency between an observation and a number of lagged observations is used in this model, I: Integrated - To make the time series data stationary, differencing of observations is used, and MA: Moving Average - Dependency between an observation and error from a moving average model is applied to the observations that are in lag [14]. Parameters p, q, and d constitute the standard notation of ARIMA and can be defined as the number of lags (p), the degree of differencing (q), and the size of moving average window (d).

Training: We leveraged the code for ARIMA from Machine Learning website [14], we trained the model by tuning various parameters that make up the model. The model was repeatedly run for different values of parameters p, q and d such that the predictions were more accurate and it resulted in low error results. Parameter p is deduced by looking at the autocorrelation function of the time series data. Parameter q is deduced by looking at the partial correlation function. The number of lags going beyond the critical range will be assigned to q [15].

Testing: The code was tested by using the above deduced parameters. Some combinations of the parameters gave error "SVD did not converge" and some gave different output values – predictions and error results which are discussed under the 'Evaluation Results' section.

For different values of parameters p, q, and d, their respective mean absolute error values for the AP with ID KNG-M-M30-1 are listed in Table III.

TABLE III
ARIMA RESULTS

Parameters - p, q, d	MAE
5, 1, 1	324.423
4, 1, 1	326.857
6, 1, 3	323.839

Fig. 2 shows an example of actual values vs. predicted values using the ARIMA model for a randomly chosen location, the AP ID KNG-M-M30-1.

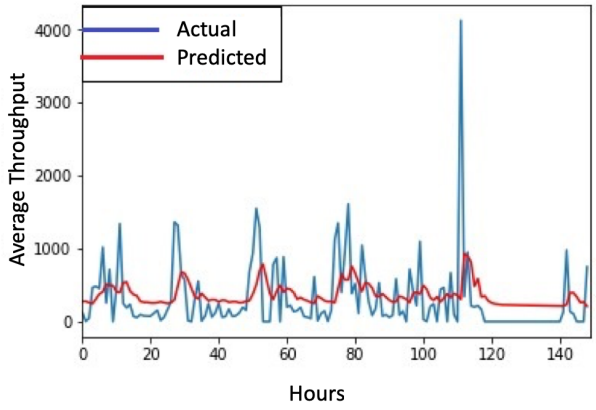

Fig. 2. ARIMA prediction results.

b) Facebook prophet: Prophet is used for forecasting time series data based on additive model. Holidays, weekly, monthly, and yearly trends can be used to predict using this non-linear model. It automatically removes outliers and fills missing data. It can be easily integrated into R and Python project which are the most commonly used technologies for data science projects.

Training: The input to prophet is always a data frame containing 2 columns 'ds' and 'y', it is to be observed that the column names should not be altered. The column 'ds' should be a date or time stamp in one of two possible formats, 'YYYY-MM-DD' or 'YYYY-MM-DD HH:MM:SS', and column 'y' should be a numeric value which should be used as historic data for the forecast.

Testing: Data is divided to train and test. Forecasts are generated by training the Prophet model and it shows details about optimization of data, convergence, and relative gradient magnitude.

Evaluation Results: The value of MAE for AP KNG-M-M30-1 was found to be 332.20057. The below graph shows the actual vs predicted values for prophet model:

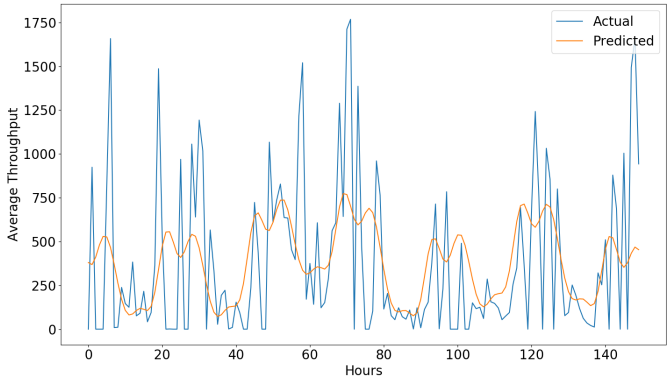

Fig. 3. Facebook Prophet prediction results.

c) RNN-LSTM: As explained in the Section 'Back-end System', RNN-LSTM is a machine learning model extensively used for time-series prediction problems.

Data Preparation: As explained in the section Data Preprocessing for regression models, we aggregated average throughput values for each AP for a month on an hourly basis. The aggregated values are scaled in the range of 0 and 1 to nullify the effect of large values on the prediction.

Training: We split the data for each AP as 0.1% of total length of the dataset for test and 0.9% of total length of the dataset for training. Training and test data are prepared with a window size of 4 time steps, with the independent variable (X train) containing previous 4 hours of average throughput values and the dependent variable (Y train) containing next 4 hours including the 5th hour. Batch size is used to have equal number of records in each batch and is set to 32 for this model. Hence, the shape of each batch fed to the model is 32 records, 4 time steps and 1 feature (average throughput).

Testing: We experimented the model on the data prepared with variations in its parameters, such as number of LSTM layers, number of units in each cell of the LSTM layer, number of iterations and number of epochs, and evaluated using MAE for each time step. The test data is prepared in the same manner as that of the training data with the independent variable (X test) containing previous 4 hours of throughput for every 5th hour and the model predicted the next 4 hours of every hour from the 5th hour. Table IV shows results from our experiments with the RNN-LSTM model.

TABLE IV
RNN-LSTM RESULTS

LSTM units	Epochs	Iterations	MAE
100,100	40	200	420.06
50,50	40	215	347.60
20,25	40	200	289.44

Fig. 4 depicts the actual and predicted values of average throughput measured in Kbps for a randomly chosen location.

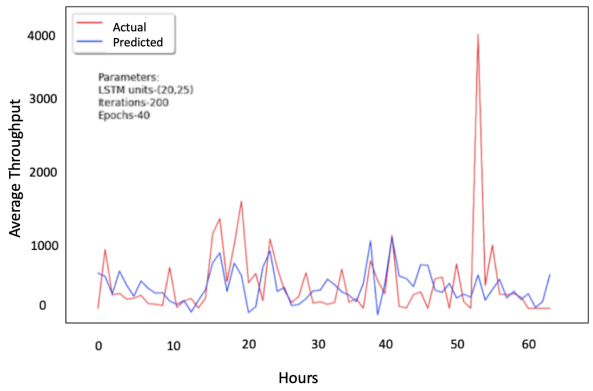

Fig. 4. RNN-LSTM prediction results.

C. Client Implementation

The mobile application is built using android studio and signed using debug.keystore. The mobile app is made available in Google Play store to be installed on android devices. The web application is developed using technologies such as Python Flask, HTML, jQuery, JavaScript and CSS to provide users a geographical visualization of APs, with their availability and type of activity performed. The web server is launched and listens to the port 5000 for requests. The application can be accessed at `https://localhost:5000`.

1) Client Components: Below are the components of the user interface of our mobile and web based applications.

a) Date and time pickers: The launch page of both the web and mobile based applications provide users with a date-time picker widget to view availability and type of usage of APs for the selected date and time.

b) Map: APs in the MLK library and engineering building are displayed on a map using Google maps API. Coordinates of APs in the respective buildings are recorded and fed to the maps API to place markers at appropriate locations.

c) Markers: Each AP is represented with a marker on the map. On clicking each marker, details about the AP such as its name, predicted throughput and type of activity it was used by majority of users are displayed. Color codes are used to differentiate highly available and busy APs used for work and entertainment activities. A legend on the map describes the color codes used for differentiating intensity and type of activity performed using each AP.

2) Mobile application: The mobile application contains an interactive date and time picker widgets. After selecting date and time, user is displayed a set of markers on Google maps

for each building color coded based on their availability and type of usage. On selecting a marker, details of the AP are shown in marker info window.

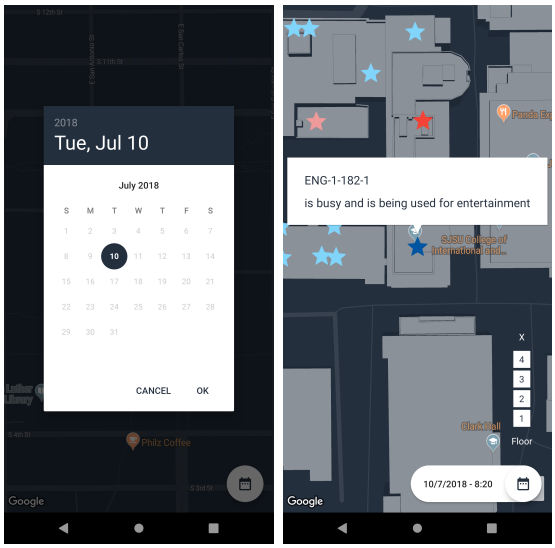

Fig. 5. Mobile application – date picker and busy marker in the entertainment category.

3) Web Application: In this section, we explain the design of our project's Web application

a) Launch page: The user can select a preferred date and time for which he/she wants to know the availability of the APs suitable to perform work or entertainment activities.

b) Maps page: After the user submits the request on the launch page, the user is directed to a page which displays the type of activity performed and availability of APs in SJSU's MLK Library and Charles Davidson Engineering buildings using Google Maps as shown in Fig. 7. Based on the date provided, it takes the output from the classification and regression models stored in files on the system. The output of RNN-LSTM model, stored in a CSV format contains AP's name and the predictions for the next four hours. The output of the random forest model is stored in a MySQL database. An SQL query is executed to obtain the number of users using APs for work and entertainment activities and converted to JSON format to be fed to the application. The results are grouped together and displayed accordingly on the user interface.

VI. Benchmarks and Performance

A. Benchmarks

1) Classification Models: In his study, George Forman [16] took several algorithms into consideration for text classification, including Naïve Bayes and random forest. He found that the random forest models performed better than alternatives, resulting in higher precision.

2) Regression Models: Zhuo et al. [9] showed that LSTM provides better accuracy for a time series data. Additionally, Feng and Shu [7] showed that Neural Network predictor performs better than other popular network traffic predictors such as ARIMA, FARIMA.

B. Performance

Performance of the Classification and Regression models are described below:

1) Classification Models: The performance of each of the classification model was compared based on weighted F1 score. The table with the scores is given in Table V.

TABLE V
Evaluation Results of Classification Models

Models	F1 Score
Random forest	0.718
Gaussian naïve Bayes	0.673
XGBoost	0.565

2) Regression Models: The performance of the Regression models are compared using Average MAE. The average mean absolute error for the three models is given in Table VI.

TABLE VI
Evaluation Results of Regression Models

Models	Average Mean Absolute Error
RNN-LSTM	63.350
ARIMA	110.961
Facebook Prophet	122.137

We can deduce, from the extensive evaluation we performed, that the RNN-LSTM model performs better than the ARIMA and Facebook prophet model alternatives.

VII. Conclusions

Based on the average MAE scores of the regression models, we conclude that RNN-LSTM model is more efficient than ARIMA and Facebook Prophet. F1-scores of classification models confirm that random forest performs better than the Gaussian naïve Bayes and the XGBoost classification models. Therefore, we used the RNN-LSTM as our regression model to predict average throughput for up to 4 hours in future and the random forest classification model to classify URLs as work or entertainment. Based on these models, we designed a Web based application and a mobile app that recommends users a study area on SJSU's campus. Our application takes advantage of mobile phone features (for localization) as well as cloud-based processing for machine learning inference.

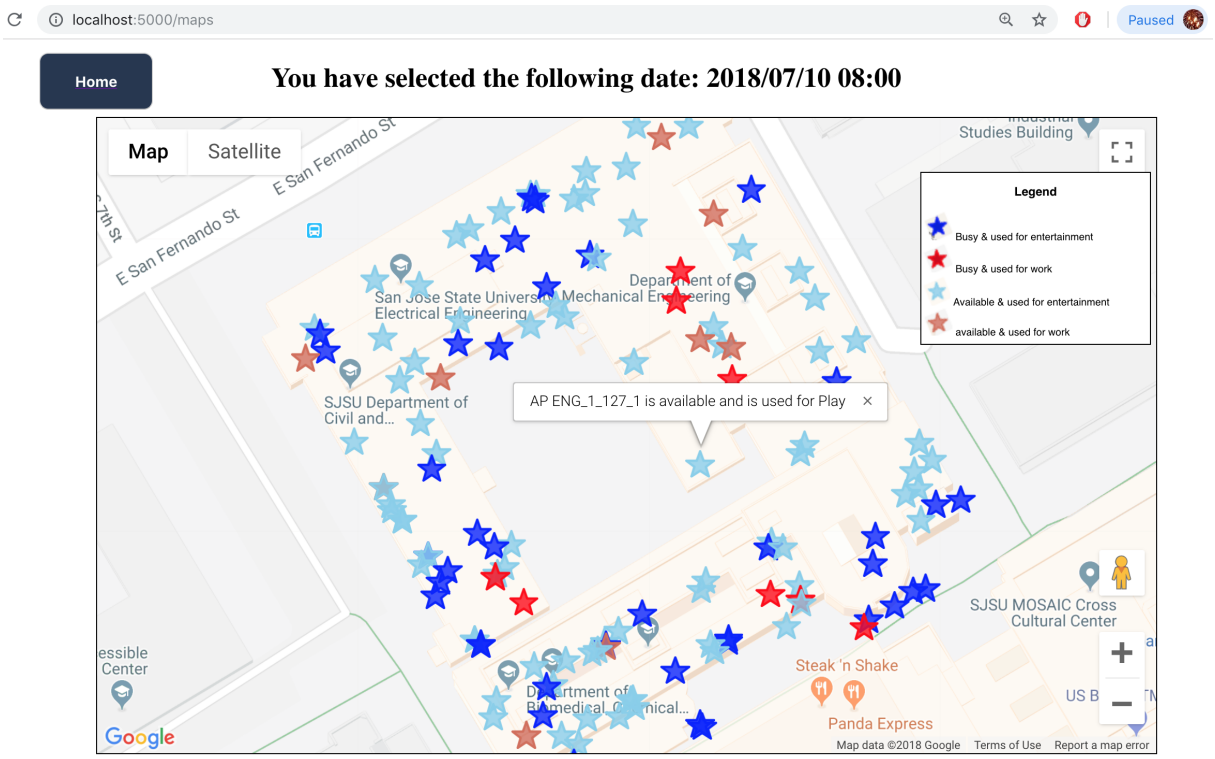

Fig. 6. Web Application – user interface – Engineering building access points.

REFERENCES

[1] M. Shafiq, X. Yu, A. A. Laghari, L. Yao, N. K. Karn, and F. Abdessamia, "Network traffic classification techniques and comparative analysis using machine learning algorithms," in *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*, Oct 2016, pp. 2451–2455.

[2] M. Yuantian, R. Zichan, P. Lei, Z. Jun, and X. Yang, "Comprehensive analysis of network traffic data," vol. 30, no. 5, p. e4181, e4181 cpe.4181. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4181

[3] D. Prangchumpol, "Improving the performance of network traffic prediction for academic organization by using association rule mining," in *Fourth edition of the International Conference on the Innovative Computing Technology (INTECH 2014)*, Aug 2014, pp. 93–96.

[4] Y.-H. Wen and T.-T. Lee, "Fuzzy data mining and grey recurrent neural network forecasting for traffic information systems," in *IRI -2005 IEEE International Conference on Information Reuse and Integration, Conf, 2005.*, Aug 2005, pp. 356–361.

[5] T. H. H. Aldhyani and M. R. Joshi, "Integration of time series models with soft clustering to enhance network traffic forecasting," in *2016 Second International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*, Sept 2016, pp. 212–214.

[6] T. Eterovic, S. Mrdovic, D. Donko, and Z. Juric, "Data mining meets network analysis: Traffic prediction models," in *2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, May 2014, pp. 1479–1484.

[7] H. Feng and Y. Shu, "Study on network traffic prediction techniques," in *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, vol. 2, Sept 2005, pp. 1041–1044.

[8] D. Kang, Y. Lv, and Y. y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–6.

[9] Q. Zhuo, Q. Li, H. Yan, and Y. Qi, "Long short-term memory neural network for network traffic prediction," in *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2017, pp. 1–6.

[10] C. Olah. Understanding lstm networks. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[11] R. W. Talbot, C. Acheampong, and R. H. Wicentowski, "Swash: A naive bayes classifier for tweet sentiment identification," in *2013 25th Chinese Control and Decision Conference (CCDC)*, 2015, pp. 4287–4290.

[12] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 785–794. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939785

[13] P. Tao, H. Yi, C. Wei, L. Y. Ge, and L. Xu, "A method based on weighted f-score and svm for feature selection," in *2013 25th Chinese Control and Decision Conference (CCDC)*, May 2013, pp. 4287–4290.

[14] J. Brownlee. How to create an arima model for time series forecasting in python. [Online]. Available: https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python

[15] Abishek. How to identify arima p d and q parameters and fit the model in python. [Online]. Available: https://www.youtube.com/watch?v=bqvZL8Ww3aA

[16] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pages=1289-1305, 2003.