Evaluation of machine learning prediction models for WiFi saturation sensing

Wouter Slegers
University West
Uppsala, Sweden
wjslegers@gmail.com

***Abstract* – In this paper we use various statistical methods to test and create Machine Learning (ML) classifier prediction models to predict saturation of WiFi signals. We use data gathered in [1] and test suitability of a Stochastic Gradient Descent classifier (SGDc) and Random Forest classifier (RFc) and compare those to the method suggested in [1], namely a Convolutional Neural Network (CNN). We find, just like [1], that it is possible to predict the saturation of WiFi networks with high accuracy, using inter-frame spacing statistics without the need for decoding. We also find that especially the RFc performs nearly as well as a CNN, and that this can serve as an alternative in case a model which is more suitable to CPU computations rather than GPU computations is required.**

Index Terms – Wi-Fi saturation, traffic load estimation, machine learning, statistical analysis, SciKit-Learn

1. Introduction

Modern industry is extending the deployment of wireless networks, looking into efficient networking solutions that can increase network performance. This expansion of wireless network deployments along with the rapidly growing penetration of wireless network consumer devices like smartphones and tablets have led to an exponential growth of wireless traffic demand [1]. The question of efficient connections for multiple wireless devices on the same network is an ever more important point of research. In [1] the goal is to use machine learning to predict the saturation of a WiFi network. Using, in particular, inter-frame spacing statistic of WiFi frames could lead to a boost in performance, because these statistics are 'cheaper' to obtain than more detailed information about the status and requirements of the networks. Without a need for decoding, being able to predict the saturation of a WiFi network could lead to a decrease in cost and complexity of coexistence schemes.

Problem statement

Demonstrated in [1] is the suitability of CNN algorithms for these purposes. We consider several other prediction models and evaluate their suitability to this problem.

2. The data

The data provided in [1] contains 54 columns of features and 1 column with the labels where a 1 represents a saturated WiFi signal and 0 an unsaturated one. The data is generated without any missing data, so not a lot of cleaning is necessary. Of course we do investigate and find some interesting results.

A. Studying the dataset

We take a look at the correlation between the different columns. First we see how high their correlation is to the label that we wish to predict, see fig. 1.

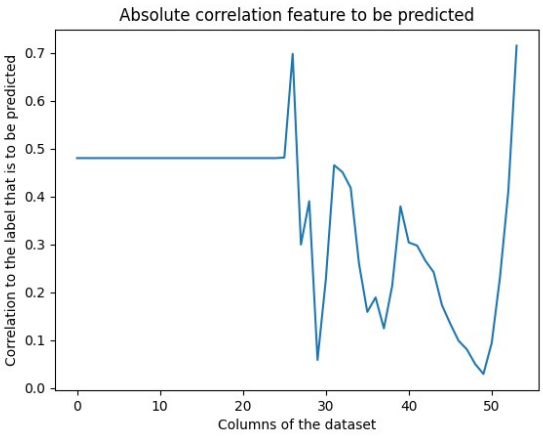

Fig. 1: Last column of the correlation matrix of the dataset

Two things are to be noted. A lot of the columns are rather highly correlated, this means that we are quite likely to make good predictions of the labels. Two of the columns even have a correlation higher than 0.7. Secondly we see that the first 26 rows have a pretty much identical correlation with our label, so let's see what that is about. When we look at the first column of the correlation matrix for the dataset, i.e. the correlation between the first feature and all the others, we see that the values for columns 0 to 24 are all 1.000000 and for columns 25 it is 0.997343. Now of course we expected the value to be 1 for column 0, the correlation of the column with itself, but not for the following 24 columns. It turns out that the columns are not identical copies of eachother, but rather the second column is pretty much exactly equal to the first column times 2. The third column is equal to the first columns times 3, etc. So our first step will be to remove columns 1 through 25, and we only keep the first column number 0. We reset the indices so we end up with columns 0 through 28 as data to predict the label in column 29.

B. Transform

Upon investigation most features seem to be distributed unequally with most entries having relatively low values and fewer features having high values. We test to see if a logarithmic transformer followed by a standard scaler or a power transformer, that automatically distributes does standard scaling as well, works better to prepare the data for our machine learning algorithms. We end up deciding to use a power transformer using yeo-johnson for all of the prediction data features.

3. Evaluation of the models

We will be considering Stochastic Gradient Descent classifier (SGDc), Decision Tree classifier (DTc), k-Nearest Neighbors classifier (kNNc), Random Forest classifier (RFc) and Convolutional Neural Network

classifier (NNc) models.

A. Suitability

1. A SGDc is a relatively 'lightweight' machine learning algorithm where learning is quick and after having learned it is also rather fast when making predictions, as compared to some of the other models we will be using. This may be a suitable method considering these traits, as our focus is on increasing efficiency. However accuracy is sometimes lower as a consequence of its efficiency. An important factor to consider is though that data may change over time due to unforeseen other factors. This model allows for easy continuous learning to maintain/improve its accuracy as new data becomes available.
2. A Decision Tree classifier (DTc) is relatively fast too, has the potential for high accuracy, but is more prone to overfitting. If we can obtain high enough accuracy it may be worth considering.
3. kNNc can obtain rather high levels of accuracy on appropriate data, but there is no learning process and the bulk of computations are saved for prediction time. This makes it a slightly cumbersome model due to its high memory usage and slow predictions. However it does allow for continued training by just saving new data alongside the previous data.
4. RFc has good potential for high accuracy and is less prone to overfitting than DTc. However increased calculation times may form a concern, both for learning and at prediction time. Also it might require slightly more memory usage.
5. NNc is the only model considered in [1] and does have the benefits of high accuracy, if trained correctly, relatively low memory usage after having been trained and high speeds at prediction times. However learning can be rather computation expensive. It does however allow for adjustments when new data becomes available.

B. Results

We perform a gridsearch with 5 folds of cross validation in order to find the best hyperparameters using the GridsearchCV method from SciKitLearn for each of the models. We find that the accuracy and f1 scores end up being pretty much identical, which makes sense since the data provided has approximately equally many entries with satisfied WiFi signals as unsatisfied ones.

For details about parameters used and exact results of the gridsearch, see the code at [3]. After finding appropriate hyperparameters we ran a cross validation training of the data for those parameters to evaluate training time. We also timed the time it took to provide predictions for the test data to give some indication of the models' efficiencies. The experiment was run on a 13th Gen Intel(R) Core(TM) i5-13400F 2.50 Ghz processor.

Models	Accuracy train set %	Accuracy test set %	Cross validation fit time	Test set prediction time
SGDc	98.55	98.52	0.1	0.0030
DTc	99.76	98.78	0.2	0.0030
RFc	99.99	99.35	18	0.0340
kNNc	99.32	99.53	0.6	0.1880
NNc	99.81	99.72	14.6	0.0043

Table 1: Results of cross validation gridsearch

One of the first things to note from the results is that the accuracy of the method on the test data is really rather high. Scoring as high as 99.7% is only possible because the data seems to exhibit high correlation with the label we aimed to predict. Secondly we can note that Rfc, kNNc and NNc scored really high on accuracy of the test set. With scores this high each of these methods would be sufficient for our purposes. However, like we mentioned earlier kNNc is not suitable for this problem due to its slow performance at prediction time and Rfc scores lower on both training and prediction time than NNc. NNc ends up being most accurate as well as having prediction times similar to SGDc and DTc. With the settings used SciKit learn does not make use of the GPU so these results were obtained through CPU usage.

4. Interesting notes on the data and models

Since the data showed high correlation it is no surprise that we were able to give accurate predictions of the labels. A question that arose was whether we really needed all the features provided to give accurate results. The RFc model provides a list of feature importances. This lets us see which features were most important for getting the results. We decided to sort our features from most important to least important and let both RFc and NNc use cross validation to fit a new dataset consisting only of the i most important features, where i is an integer between 1 and 29. Incredibly, even using a single feature made it possible for the RFc method to predict the

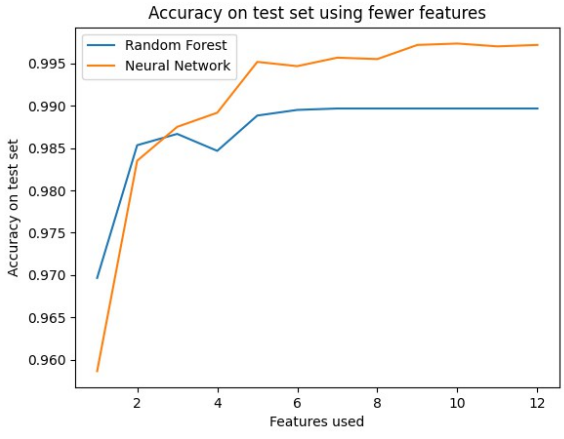

Fig. 2: Using very few features still yields high accuracy results

saturation with 96.97% accuracy on the *test* set! When the number of features goes up the NNc catches up and overtakes the accuracy of RFc like you can see in fig. 2. On each of these tests where we use fewer features the f1 score and the accuracy were pretty much identical with no larger differences than 0.0001, i.e. 0.01%. Note that choosing our features of the dataset according to the feature importances of RFc means that the RFc gets an advantage over the other method, NNc. Choosing only the features that the RFc uses most is inherently an unfair comparison and yet the NNc accuracy is already higher than that of the RFc from 3 features or more. We could also have chosen our features by ranking them by how high their correlation is compared to the label to be predicted, but this gaves us worse results for both RFc and NNc.

5. Conclusion

We confirm the result of [1] about being able to accurately predict WiFi saturation using Machine Learning. The kNNc, RFc and NNc models are each able to predict it accurately and NNc does reach the best accuracy and seem to show good performance on prediction times. We note too that we can leave out a lot of the features from the dataset and still get accurate results, to the point that using more than 9 features out of the 54 provided does not improve results and even just using 5 can give 99.5% accuracy on the test set.

References

The code used can be found on:
https://github.com/WouterSlegers/ML-for-WiFi-saturation-prediction

[1] M. Girmay, A. Shahid, V. Maglogiannis, D Naudts, I. Moerman, “Machine Learning Enabled Wi-Fi Saturation Sensing for Fair Coexistence in Unlicensed Spectrum”. 2021
[2] A. Géron, “Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow”. O'reilly, 2019