ML Models Prediction in Wifi Locationing Data

M. Martínez

October 24, 2021

Abstract

WiFi fingerprint technology is widely used to register the location of its users in indoor spaces. In this work, a WLAN fingerprint database, that contains the signal of the WiFi fingerprint located in indoor spaces, is used to test several machine learning (ML) models and look for the one with the best accuracy. The database is for three buildings with four or more floors. Six ML models were tested; logistic model tree, k-Nearest Neighbors, **C5.0**, **stochastic garden boosting, GBM**, **random forest, RF**, and support vector machine. However, only the three highlighted in bold letters converged. Among them, we found that GBM is the model with the best accuracy ($\sim 60\%$). We also noted that the accuracy could be improved if a major simplification is applied to the dataset, like separating the sample by buildings.

1 Introduction

It is widely known that there is a growing demand for applications that provide the outdoor and indoor location information of users. There is a large variety of positioning system, some of them work well for outdoor locations and other for indoor. For example, GPS technology has contributed tremendously to the interaction of people with their surroundings in outdoor spaces, while WLAN fingerprint-based technology allows providing information on location in indoor areas.

In this work, a WiFi-Fingerprint database is used to test several machine learning (ML) models. The goal is to look for the one that better predicts the user's location according to the signal of the WiFi fingerprint.

2 Data

The UJIIndoorLoc database covers three buildings of Universitat Jaume I with four or more floors and almost 110.000 m^2. The database consists of 19937 observations and 529 the features. They

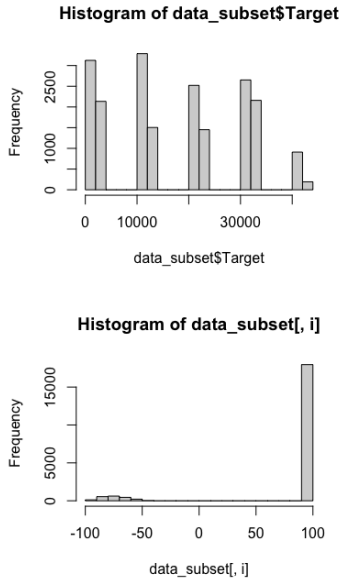

Figure 1: **Dataset**. **Upper panel**: location distribution according to the dependent variable *Target*. **Bottom**: intensity distribution of a particular WiFi fingerprint (WAP).

are,

- Location: LONGITUD, LATITUDE, FLOOR, BUILDING, SPACEID, RELATIVEPOSITION, USERID, and PHONEID
- TIME, in UNIX units
- Signal intensity: WAP001...WAP520, in decibels (dB)

In order to simplify the databased the FLOOR, BUILDING, SPACEID, RELATIVEPOSITION are combined into a single dependent variable called *Target*. In Figure 1 we plot the location and intensity distributions. The intensity distribution is plotted for a particular WiFi Fingerprint detector. The intensity ranges from -100 dB (for extremely poor signal) to 0, while a value of 100 is used as a flag indicating not-detection.

Additionally, unused columns are dropped. They are, some predictors with an uniform inten-

sity distribution, and the additional information of the user; USERID, PHONEID, TIMESTAMP, LATITUDE, and LONGITUDE.

3 ML models for classification

In this work the programing language R and six ML models from the CARET package were used; logistic model tree, k-Nearest Neighbors, **C5.0**, **stochastic garden boosting**, **random forest**, and support vector machine. However, only the three highlighted in bold letter converged; therefore, in the following a brief description is giveng only for these three models.

The three algorithms that will be described are based on classification trees and/or ruled-based models. An example of a classification tree might be,

if **Predictor B** >=0 then
|| if **Predictor A** >=-100 then **Class** =1
|| else **Class** =2
else **Class** =2

In the previous example, the two-dimensional predictor has been divided into three regions (also called nodes), and each region has been categorized according to two classes (Class 1 and Class 2). This nested IF-THEN statements can be written as rules, in the following way,

if **Predictor A**>=-100 and **Predictor B**>=0 then **Class**=1
if **Predictor A**>=-100 and **Predictor B**<0 then **Class**=2
if **Predictor A**<-100 then **Class**=2.

All algorithms are based on these classifications. However, their differences lie in the process of finding the optimal splits and rules, which depend on the optimization criteria assumed.

3.1 C5.0

This algorithm is based on trees and rules classifications. In the former case, the algorithm combines non occurring conditions for splits with several categories, resulting in a final global pruning procedure that removes the sub-trees based. In the latter, a first tree is built (based on rules), then the rules are simplified considering the most likely class. In CARET the method C5.0 works only for classification and allow the tunning of three parameters; the number of boosting iterations (*trials*), the model type (*model*: rules or tree), and *winnow*, which gives the feature selection approach. For example, predictors are considered unimportant if they are not in any split in the winnowing tree.

3.2 Stochastic garden boosting

A basic gradient boosting algorithm has two tunning parameters, the *tree depth* and the *number of iterations*, and the stochastic framework is included by adding a random sampling approach. In CARET the namely *gbm* method works for classification and regression, and two additional parameters can be tunned, the *shrinkage* that regularize the learning rate, and the minimal terminal node size *n.minobsinnode*.

3.3 Random forest

In this algorithm, each tree in the forest picks a vote for new sample classification. Then, the proportion of votes in each class are used to build the predicted probability vector. This algorithm has a single tunning parameter called *mtry*, which is the number of randomly selected predictors within each split.

4 Analysis

To do the analysis, I built a pipeline that started by fixing the see in order to secure the reproducibility of the results. Then, a subsample of 300 observations was selected; this subsample included all the predictors. This sample was divided into the training (70%) and testing data (30%).Along all this process, the function DROPLEVES() was used to remove the levels no longer used due to the subsetting. Next, 10-fold cross-validation for ten iterations was chosen.

For the C5 model we vary the following parameters, trials = c(1:9, (1:10)*10), model = c("tree","rules"), winnow = c(TRUE, FALSE), for the GBM model, shrinkage = c(0.01,.01, .1), interaction.depth = c(1, 3,5), n.minobsinnode = c(5, 10, 15), n.trees = c(10,50,100), and for the RF model, mtry=c(1,2,3). Using five cores with a velocity of 2.3 GHz the time running the pipeline was of 10 minutes.

After a first running, we selected the most important features (the first 10) for each model. Then we reduced the number of features by choosing only the most important and ran the pipeline again. We find that the accuracy of the models improved. Table 1 lists the metrics and the parameters that produced the best model.

There is not a large difference between the metrics (accuracy and kappa) of the models, although the GBM is the one with the largest accuarcy. We

Model	Accuracy	Kappa	Parameters
GBM	0.59	0.47	*Shrinkage=0.01*, n.minobsinode=5, n.trees=100, int.depth=5
RF	0.58	0.45	*mtry*=3
C5	0.58	0.45	trials=40, model=rules, winnow=false

Table 1: Metrics and parameters.

found that WAP144 and WAP248 are consistently between the most important features among the three models. In Figures 2 we plot the metrics obtained from the grid of parameters and for each model and in Figure 3 the feature importance for the GBM model. Finally, in Table 2 we list the references and predictions for the tested dataset, which ar obtained with and accuracy of $\sim 68\%$.

We conclude that all models work well for the training of the WiFi fingerprint database. However, better accuracies might be obtained if more simplification are done to the data, for example, by choosing a lower number of predictors and improving the grid of parameters.

5 Recomendations

We list the following recommendations,

- The GBM algorithm is the most accurate for training the wife fingerprint data.
- The models are very sensitive to the number of features and combined features (BUILDING, FLOOR, SPACEID, and RELATIVEPOSITION) that are used to create a single dependent variable. For example, combining only the BUILDING, FLOOR features improve the performance of the models, likely due to the simplicity of this combination.
- It is very important to supervise the lack of information during the sampling or when creating the training and testing dataset and dropping those levels with missed information.
- It is necessary to keep at least 20 features since a lower number produces lower accuracies.
- The accuracy of the models is very sensitive to the shape of the sample, for example, the number of predictors. Finally, we highlight that a better accuracy might be obtained if the sample is divided by buildings.

References

[1] https://stats.stackexchange.com/

[2] Papapostolou A., Chaouchi H. (2009) *WIFE: Wireless Indoor Positioning Based on Fingerprint Evaluation*. In: Fratta L., Schulzrinne H., Takahashi Y., Spaniol O. (eds) NETWORKING 2009. NETWORKING 2009. Lecture Notes in Computer Science, vol 5550. Springer, Berlin, Heidelberg.

[3] Max Kuhn and Kjell Johnson, *Applied Predictive Modeling*, Springer, 2016, ISBN 978-1-4614-6848-6

[4] Max Kuhn, The CARET Package, 2019, http://topepo.github.io/caret/index.html

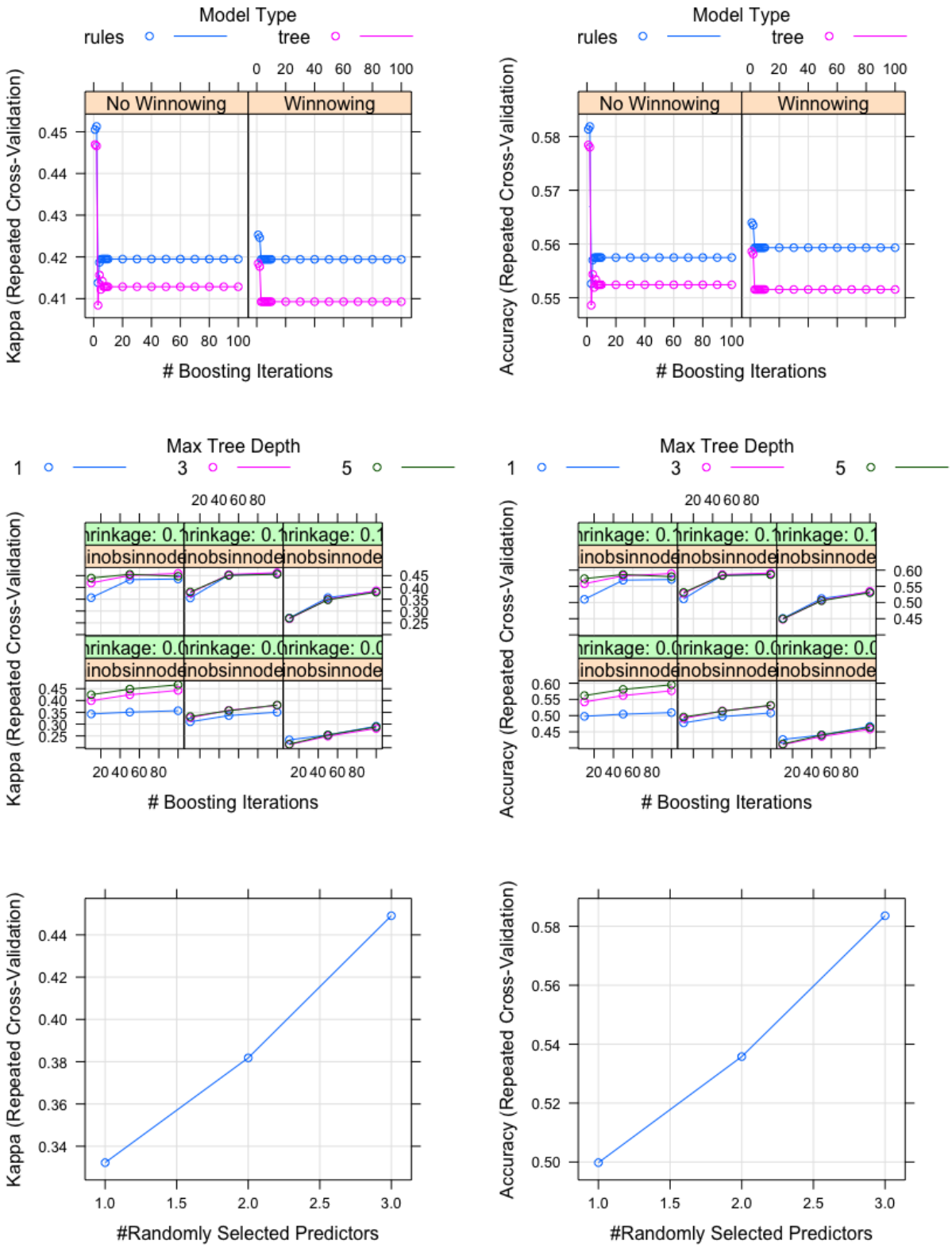

Figure 2: **Right panel: Kappa**. From up to bottom is the kappa metric for C5, GBM, and RF models. Note that a larger Kappa indicates a better prediction of the classes. **Left panel: as of right panel but for the accuracy**.

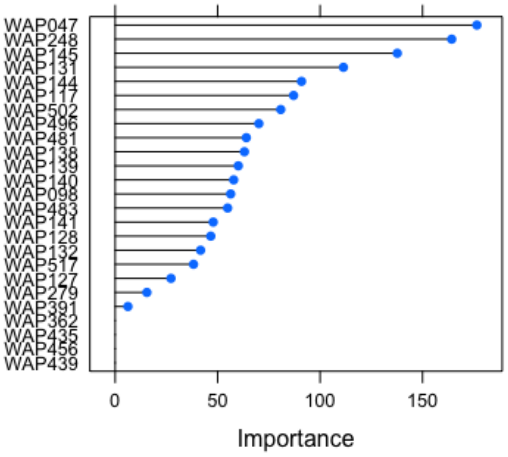

Figure 3: **Feature importance for the GBM model**.

Reference	0	1	2	3	4
Prediction					
0	24	13	5	4	0
1	0	9	5	0	0
2	0	1	11	1	0
3	0	1	3	10	0
4	0	0	0	0	7

Table 2: Confusion Matrix.