**Evaluate Techniques for Wifi Locationing**

### Overall Impression

In this classification task, I first used library() to recall a series of previously installed packages like caret, readr, tidyr etc. We have almost 20,000 observations and 529 attributes in the WIFI Location training Dataset. We need to preprocess these data in order to lower the granularity and achieve the dimensionality reductions. Preprocessing is a vital step that involves transforming raw data into an understandable format. The phrase “garbage in, garbage out” is particularly applicable to machine learning projects. After testing on various models, I compared the Accuracy and Kappa values and Confidence Interval and P-Values for each model, I chose Random Forest model for my final prediction.

### Preprocessing

#### Step 1 – Removing the Near Zero Variance

When exploring the entire dataset, I found that there are total 520 WAP WIFI signal attributes, and a lot of data have the same value, i.e. 100 which means in these WAPs that have not been detected by the device. These are the near Zero Variance data which are less informative and could be taken away. Hence, I applied rzv\_training <- processed\_trainingData[, sapply(processed\_trainingData, var) != 0] to keep only the non-near zero variance data in the dataset. Then I manually deleted five irrelevant attributes, i.e. longitude, latitude, userID, phoneID and Timestamp, thus reducing my total attributes from 529 to 469.

#### Step 2 – Creating a unique identifier

Furthermore, I tried to combine the building, floor, and specific location attributes into a single unique identifier for each instance. I used the unite () function in the tidyr package to create a new attribute “LOCATION” and put it in the first column of the dataset. Because this is a Classification question, I must FACTORIZE this dependent variable before feeding into various models.

#### Step 3 – Sub-setting the dataset and refactorize the sub-sets

We have three buildings (i.e. 0, 1, 2) here. I subset into 3 by specifying the buildingID# for each subset and then I removed Floor, BuildingID, SpaceID, and RelativeLocation from these subsets. So my final attributes that feed into these algorithms are 466. Lastly, I applied factorizing again after subsetting in order to drop factor levels in a subsetted data frame.

#### Step 4 - Set seed and Create Data Partition

In order to produce the same results every time, I used the set.seed() function and followed by the 75%/25% train/test split of the dataset on each building subset. The syntax is like below:

inTraining0 <- createDataPartition(trainingBUD0$LOCATION, p = .75, list = FALSE)

training0 <- trainingBUD0[inTraining0,]

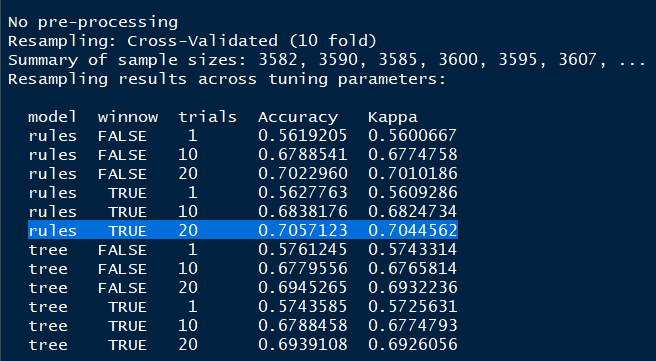
testing0 <- trainingBUD0[-inTraining0,]

### Comparison of the various models based on performance metrics

### **C5.0**

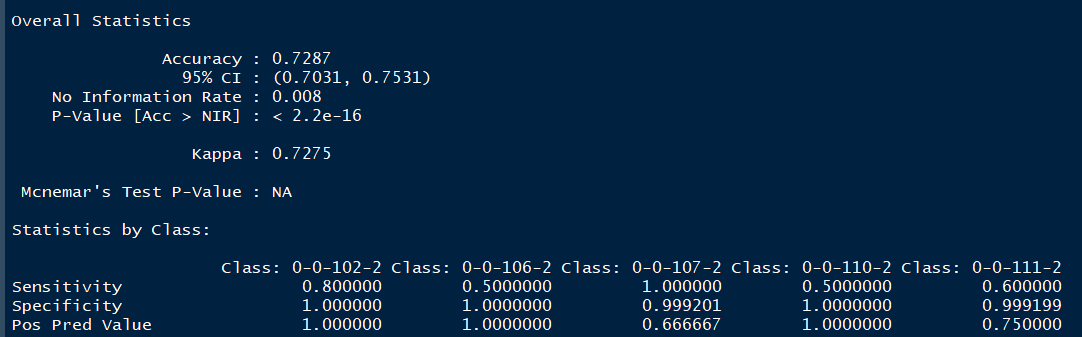
#### C5.0 with 10-fold Cross-Validation for Building0

C50\_BUD0 <- train(LOCATION~., data = training0, method = "C5.0", trControl=fitControl)

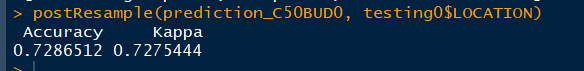


#### Kappa & Accuracy Value on C5.0 model and partial result on Confusion Matrix (Building0)

cm\_C50\_BUD0 <- confusionMatrix(prediction\_C50BUD0, testing0$LOCATION)



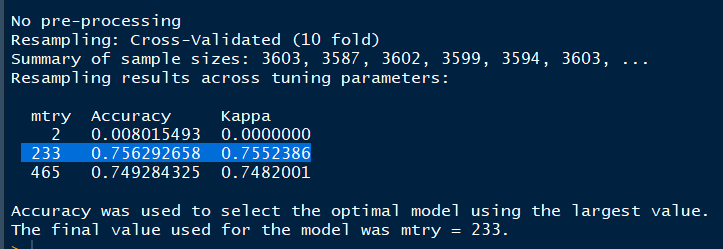
#### C5.0 Model: PostResample function on Building0



### **Random Forest Model (RF)**

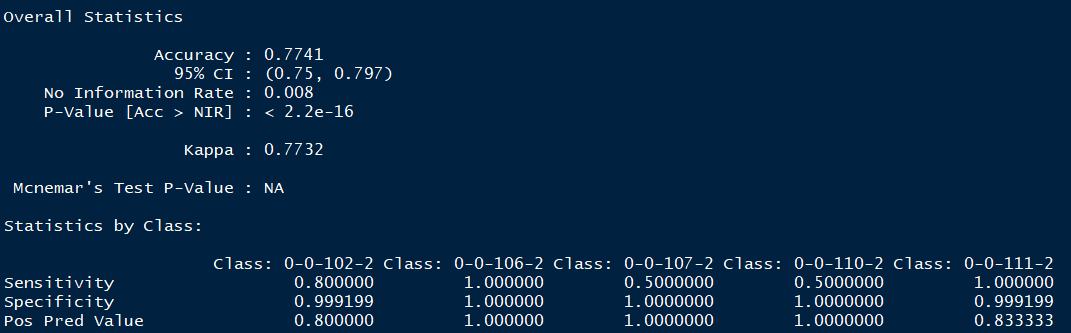
#### RF Model with 10-fold Cross-Validation for Building0

rf\_BUD0 <- train(LOCATION~., data = training0, method = "rf", trControl=fitControl)

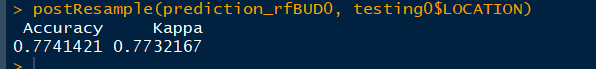


#### Kappa & Accuracy Value on RF model and partial result on Confusion Matrix (Building0)

cm\_rf\_BUD0 <- confusionMatrix(prediction\_rfBUD0, testing0$LOCATION)



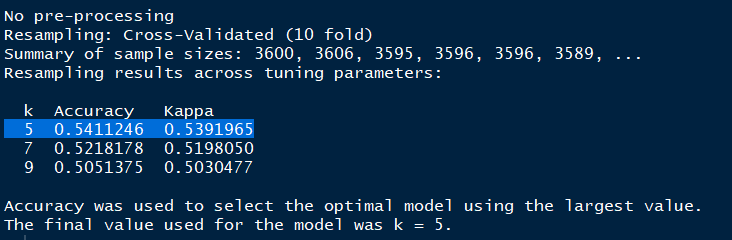
#### RF Model: PostResample function on Building0



### **K Nearest Neighbor Model (KNN)**

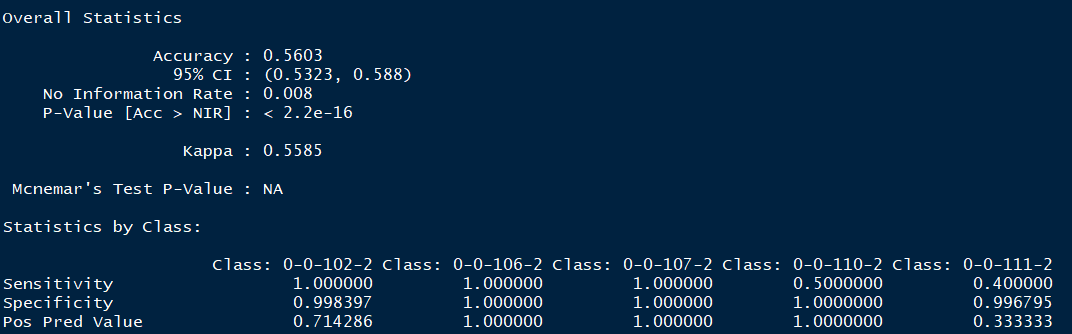
#### KNN with 10-fold Cross-Validation for Building0

KNN\_BUD0 <- train(LOCATION~., data = training0, method = "knn", trControl=fitControl)

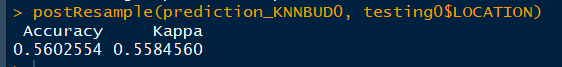


#### Kappa & Accuracy Value on KNN model and partial result on Confusion Matrix (Building0)

cm\_KNN\_BUD0 <- confusionMatrix(prediction\_KNNBUD0, testing0$LOCATION)



#### RF Model: PostResample function on Building0

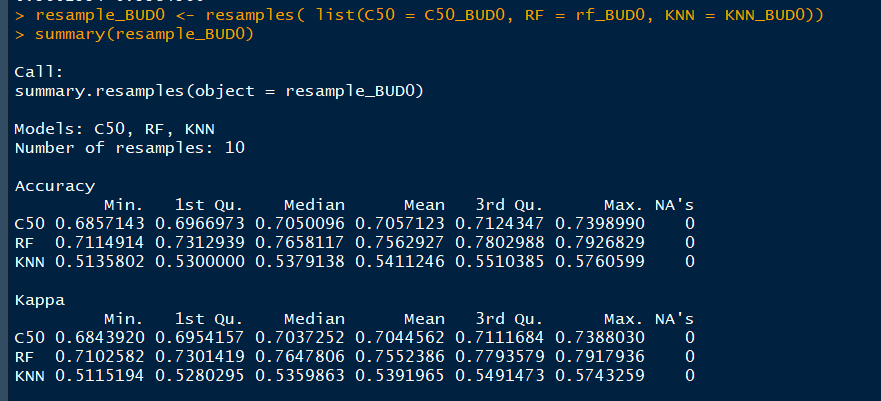


### A recommendation of the algorithm you believe to be best for this data and a justification why it is the preferred choice in terms of your interpretation of the performance metrics.

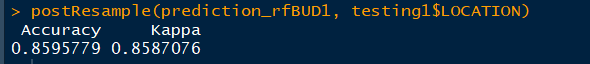
#### Use Resample function to compare three trained models

As we can see from below chart, random forest model performs the best among all three models, which gives the highest Accuracy and Kappa Values (i.e. mean= 0.7562927 for Accuracy and mean= 0.7552386 for Kappa). Followed by C5.0 model and least desirable model in this case is KNN (about 0.54 for both Kappa and Accuracy values)

Therefore, I can safely apply the Random Forest algorithm on two other buildings. As shown in the 3rd and 4th screenshots of this page, the Accuracy and Kappa value are 0.8595779 & 0.8587076 respectively on building1 which gives us the best prediction among all three buildings.



#### RF algorithm on Building 1



#### RF algorithm on Building 2

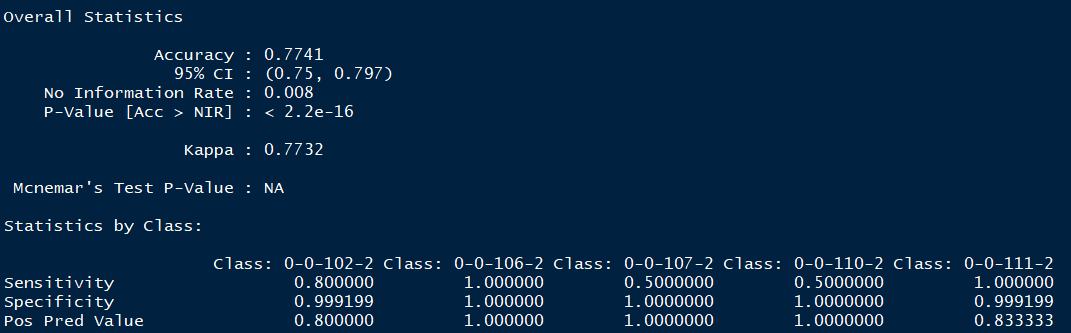


#### Brief Indication on Kappa Vs. Accuracy and Sensitivity Vs. Specificity

Kappa Score is a metric that compares an Observed Accuracy with an Expected Accuracy. In general, it is less misleading than simply using accuracy as a metric; computation of Observed Accuracy and Expected Accuracy is integral to comprehension of the Kappa Score, and it is most easily seen in the use of a confusion matrix.

Sensitivity (true positive rate) measures the proportion of actual positives that are correctly identified.

Specificity (the true negative rate) measures the proportion of actual negatives that are correctly identified.



### Recommendations, based on your own research on indoor locationing and/or experimentation with the dataset, for how the results might be improved.

### Other Indoor Positioning systems

Indoor positioning systems use different technologies, including distance measurement to nearby anchor nodes (nodes with known fixed positions, e.g. WiFi/LiFi access points or Bluetooth beacons), magnetic positioning, dead reckoning. They either actively locate mobile devices and tags or provide ambient location or environmental context for devices to get sensed.

### How to enhance the signal for existing WAP detectors with Beacon Technology

Among 520 WAP attributes in this dataset, we have a lot of identical value of 100(no signal has been detected) these near zero variances which previously have been removed from our initial dataset during the preprocessing steps. To solve this problem, we can try to apply Beacon technology, with based on Bluetooth low energy proximity sensing by transmitting a universally unique identifier picked up by a compatible app or operating systems. iBeacon can be used with an application as an indoor positioning system, which helps smartphones determine their approximate location in a building or a store.

If we can combine existing WAP detectors with this iBeacon technology, the signal should be stronger enough to cover more areas in the building. The total near Zero Variance attributes might be less, and our dataset might be more meaningful and informative than the current training dataset.

### Lesson Learnt from this project

Initially, I had faced a difficult in getting a reasonable Kappa and Accuracy Value. My Kappa and Accuracy for C5.0 is about 0.55-0.58 and my Random Forest’s result is only 0.001. I have struggled for two days, and I tried some different preprocessing methods but still did not work. Finally, I tried to remove the tunelength =1 from my script in C5.0.

Same thing applied for my random forest script: I skipped rfGrid <- expand.grid(mtry=c(1,2,3,4,5)) It is the manually grid tune method.

This time, both models took way longer time to tun and the results are quite good! I need to let Caret tune more by itself without limiting the tunelength and grid.