Additional Results for Task 3 – Course 3 – WiFi Fingerprinting

By Roberto Siegert – Nov. 18th 2019

Part II

Summary

The Kappa value is a coefficient that measures the level of agreement between classification and truth values. A kappa value equals to one (1) represents a perfect agreement, whereas a kappa value closer or equal to zero (0), represents little or no agreement.

The overall accuracy is the result of summing the number of correctly classified values and dividing the result by the total number of values. In here, the correctly classified values are located along the upper=left to lower-right diagonal of the confusion matrix.

This analysis complements the initial report submitted but makes emphasis on a better inspection of the population of signal strength intensity datasets. The approach used focused on sorting out the population of WAP values by BuildingID, which resulted in three (3x) subsets, which were then analyzed by floor.

The three resulting datasets were evaluated using the following classification algorithms: kkn, kknn, random forest and C5.0. All algorithms yielded to kappa and accuracy levels very close to unity, which implies that the algorithm selection made is sound and effective when addressing the task as a classification problem.

```
R version 3.3.2 (2016-10-31) -- "Sincere Pumpkin Patch"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

Microsoft R Open 3.3.2
The enhanced R distribution from Microsoft
Microsoft packages Copyright (C) 2017 Microsoft

Loading Microsoft R Client packages, version 3.3.2.0033.
Microsoft R Client limits some functions to available memory.
See: https://msdn.microsoft.com/en-us/microsoft-r-client-windows for informat
ion
```

```
about additional features.

Type 'readme()' for release notes, privacy() for privacy policy, or
'RevoLicense()' for licensing information.

Using the Intel MKL for parallel mathematical computing(using 4 cores).
Default CRAN mirror snapshot taken on 2016-11-01.
See: https://mran.microsoft.com/.

[Workspace loaded from C:/Users/rjsie/Desktop/UT Data Analytics/course3/Cours
e3-task3/Wifi-fingerprinting-assignment/.RData]

> knn_CM
Confusion Matrix and Statistics

          Reference
Prediction    0    1    2
         0 1312    0    0
         1    0 1299    0
         2    0    0 2373

Overall Statistics

               Accuracy : 1
                 95% CI : (0.9993, 1)
    No Information Rate : 0.4761
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            1.0000   1.0000   1.0000
Specificity            1.0000   1.0000   1.0000
Pos Pred Value         1.0000   1.0000   1.0000
Neg Pred Value         1.0000   1.0000   1.0000
Prevalence             0.2632   0.2606   0.4761
Detection Rate         0.2632   0.2606   0.4761
Detection Prevalence   0.2632   0.2606   0.4761
Balanced Accuracy      1.0000   1.0000   1.0000
>
>
> Build_1 <- subset(training_data, BUILDING == 0)
Error in eval(expr, envir, enclos) : object 'BUILDING' not found
> library(readr)
> library(plyr)
> library(dplyr)

Attaching package: ‘dplyr’

The following objects are masked from ‘package:plyr’:

    arrange, count, desc, failwith, id, mutate, rename,
    summarise, summarize
```

```
The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union

> library(ggplot2)
> library(caret)
> Build_1 <- subset(training_data, BUILDINGID == 0)
> Build_2 <- subset(training_data, BUILDINGID == 1)
> Build_3 <- subset(training_data, BUILDINGID == 2)
> dim(Build_1)
[1] 5249  474
> dim(Build_2)
[1] 5196  474
> dim(Build_3)
[1] 9492  474
> # we can remove columns in which values are All zero
> #-----------------------------------------------------------
> uniquelength <- sapply(Build_1, function(x) length(unique(x)))
> Build_1 <- subset(Build_1, select=uniquelength > 1)
> uniquelength <- sapply(Build_2, function(x) length(unique(x)))
> Build_2 <- subset(Build_2, select=uniquelength > 1)
> uniquelength <- sapply(Build_3, function(x) length(unique(x)))
> Build_3 <- subset(Build_3, select=uniquelength > 1)
>
>
>
> dim(Build_1)
[1] 5249  208
> dim(Build_2)
[1] 5196  215
> dim(Build_3)
[1] 9492  211
>
> View(Build_1)
> #We now evaluate only building specific dataset and select only columns wit
h WAP values and the FLOOR column, we will do it for Buil_1, 2 and 3
> Build_1 <- Build_1F
Error: object 'Build_1F' not found
> Build_1F <- Build_1
> Build_2F <- Build_2
> Build_3F <- Build_3
> Build_1F$FLOOR <- factor(Build_1F$FLOOR)
> columns <- c(1:200, 203)
> training1 <- Build_1F[,columns]
> set.seed(123)
> intraining1 <- createDataPartition(y=training1$FLOOR, p=0.75, list=FALSE)
> train_set1 <- training1[intraining1,]
> test_set1 <- training1[-intraining1,]
>
> ctrl <- trainControl(method="cv", number = 10)
> knn_1 <- train(FLOOR ~.), data = train_set1, method = "knn", trControl = ct
rl, tuneLength = 5)
Error: unexpected ',' in "knn_1 <- train(FLOOR ~.),"
```

```
> knn_1 <- train((FLOOR ~.), data = train_set1, method = "knn", trControl = c
trl, tuneLength = 5)
> knn_1
k-Nearest Neighbors

3939 samples
 200 predictor
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3545, 3544, 3545, 3545, 3546, 3546, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9956833  0.9942196
   7  0.9959391  0.9945624
   9  0.9939080  0.9918433
  11  0.9934029  0.9911679
  13  0.9918820  0.9891330

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 7.
> knnPredict1 <- predict(knn_1, newdata = test_set1)
> knnPredict1
   [1] 0 0 0 0 0 0 0 0 0 2 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
  [34] 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0
  [67] 0 0 0 0 0 0 0 0 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
 [100] 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0
 [133] 0 0 0 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3
 [166] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 1 2 1 1 1 1 1
 [199] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [232] 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1
 [265] 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3
 [298] 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
 [331] 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3 3
 [364] 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 2 2 2 2 2
 [397] 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0
 [430] 0 0 0 0 0 0 0 2 0 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 [463] 2 2 2 2 2 2 2 2 2 2 2 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0
 [496] 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
 [529] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0
 [562] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1
 [595] 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0
 [628] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2
 [661] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [694] 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [727] 2 1 2 2 2 2 2 2 2 2 1 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0
 [760] 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
 [793] 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [826] 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2
 [859] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0
 [892] 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
 [925] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [958] 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [991] 1 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 310 entries ]
```

```
Levels: 0 1 2 3
> knn_CM1 <- confusionMatrix(knnPredict1, test_set1$FLOOR)
> knn_CM1
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 264   3   2   0
         1   0 336   1   0
         2   0   0 356   0
         3   0   0   1 347

Overall Statistics

               Accuracy : 0.9947
                 95% CI : (0.989, 0.9978)
    No Information Rate : 0.2748
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9928
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9912   0.9889   1.0000
Specificity            0.9952   0.9990   1.0000   0.9990
Pos Pred Value         0.9814   0.9970   1.0000   0.9971
Neg Pred Value         1.0000   0.9969   0.9958   1.0000
Prevalence             0.2015   0.2588   0.2748   0.2649
Detection Rate         0.2015   0.2565   0.2718   0.2649
Detection Prevalence   0.2053   0.2573   0.2718   0.2656
Balanced Accuracy      0.9976   0.9951   0.9944   0.9995
> dim(Build_2F)
[1] 5196  215
> Build_2F$FLOOR <- factor(Build_2F$FLOOR)
> columns <- c(1:207, 209)
> training2 <- Build_2F[,columns]
> VIew(training2)
Error: could not find function "VIew"
> View(training2)
> columns <- c(1:208, 210)
> training2 <- Build_2F[, columns]
> columns <- c(1:207, 210)
> training2 <- Build_2F[, columns]
> set.seed(123)
> intraining2 <- createDataPartition(y=training1$FLOOR, p=0.75, list=FALSE)
> train_set2 <- training1[intraining2,]
> test_set2 <- training2[-intraining2,]
> train_set2 <- training2[intraining2,]
> knn_2 <- train((FLOOR ~.), data = train_set2, method = "knn", trControl = c
trl, tuneLength = 5)
Error in na.fail.default(list(FLOOR = c(3L, 3L, 3L, 3L, 3L, 3L, 3L, 3L,  :
  missing values in object
> View(train_set2)
> intraining2 <- createDataPartition(y=training2$FLOOR, p=0.75, list=FALSE)
> train_set2 <- training2[intraining2,]
```

```
> test_set2 <- training2[-intraining2,]
> knn_2 <- train((FLOOR ~.), data = train_set2, method = "knn", trControl = c
trl, tuneLength = 5)
>
>
> dim(Build_3F)
[1] 9492  211
> columns <- (1:203, 206)
Error: unexpected ',' in "columns <- (1:203,"
> columns <- c(1:203, 206)
> training3 <- Build_3F[, columns]
> View(training3)
> intraining3 <- createDataPartition(y=training3$FLOOR, p=0.75, list=FALSE)
> train_set3 <- training3[intraining3,]
> test_set3 <- training3[-intraining3,]
> knn_3 <- train((FLOOR ~.), data = train_set3, method = "knn", trControl = c
trl, tuneLength = 5)
> knnPredict2 <- predict(knn_2, newdata = test_set2)
> knnPredict3 <- predict(knn_3, newdata = test_set3)
> knn_2
k-Nearest Neighbors

3897 samples
 207 predictor
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3506, 3507, 3509, 3507, 3507, 3508, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9974346  0.9965518
   7  0.9958928  0.9944792
   9  0.9953800  0.9937899
  11  0.9902485  0.9868978
  13  0.9887074  0.9848272

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 5.
> knn_3
k-Nearest Neighbors

7121 samples
 203 predictor
   5 classes: '0', '1', '2', '3', '4'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 6409, 6409, 6408, 6409, 6409, 6409, ...
Resampling results across tuning parameters:

  k   Accuracy   Kappa
   5  0.9955068  0.9942647
   7  0.9943830  0.9928300
   9  0.9919969  0.9897840
  11  0.9898908  0.9870948
```

```
  13  0.9872226  0.9836876

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 5.
> knnPredict2
   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
  [68] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [135] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [202] 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1
 [269] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 1
1 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1
 [336] 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 3 3
 [403] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 3 3 3 3 2 2 3 2 2 2 2 2 2 3 3
 [470] 3 2 2 2 2 3 2 2 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 3 3 2 2 2 2 3 3 3 2 2 2
2 2 3 3 3 3 2 2 3 3 2 2 2 2 2 2 3 3 3 3 2 2 2 3 3 3 3 3 2 2 2 3
 [537] 3 2 2 3 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 2 2 3 2 3 2 2 2 2 2 2 3 3 3 3 3
3 2 2 2 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 2 2 2 2 2 2 2 3 3 3 3 3 3
 [604] 2 2 2 3 3 3 2 2 2 2 2 2 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3
2 2 2 2 3 3 3 3 2 2 2 2 2 2 3 2 3 3 3 3 2 2 2 2 2 2 3 3 3 3 2 2
 [671] 2 2 3 3 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1
 [738] 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1
1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1
 [805] 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1
1 1 1 0 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [872] 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1 1
 [939] 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 0 0 0 0 0 1 1 1 1 0 0 0 0 0 1 1 1 1
1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0 0
 [ reached getOption("max.print") -- omitted 299 entries ]
Levels: 0 1 2 3
> knnPredict3
   [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
  [68] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [135] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 2 2 1 3 3 3 3
 [202] 3 3 3 3 3 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 3 3 3 3 3 3 2 2
2 1 3 3 3 3 3 3 3 3 3 2 2 2 2 3 3 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3
 [269] 3 3 3 3 2 2 2 2 2 2 3 3 3 3 3 3 3 3 2 2 2 1 2 3 3 3 3 3 3 3 3 3 3 3 3
3 3 2 2 2 2 3 3 3 3 3 3 3 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 [336] 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 3 4 4 4 4 0 4 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4
 [403] 4 4 4 4 4 4 4 2 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 4 4 4 4 4 4 4 4 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3
 [470] 3 3 3 3 3 3 3 3 3 3 3 0 3 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4
 [537] 4 4 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 3 3 3 3 3 3
3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 1 1 1 1 1 1 1 1 1 1
```

```
 [604] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [671] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [738] 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [805] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [872] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [939] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [ reached getOption("max.print") -- omitted 1371 entries ]
Levels: 0 1 2 3 4
> knn_CM2 <- confusionMatrix(knnPredict2, test_set2$FLOOR)
> knn_CM3 <- confusionMatrix(knnPredict3, test_set3$FLOOR)
>
> knn_CM2
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 342   0   0   0
         1   0 371   0   0
         2   0   0 348   0
         3   0   0   1 237

Overall Statistics

               Accuracy : 0.9992
                 95% CI : (0.9957, 1)
    No Information Rate : 0.2856
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.999
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   1.0000   0.9971   1.0000
Specificity            1.0000   1.0000   1.0000   0.9991
Pos Pred Value         1.0000   1.0000   1.0000   0.9958
Neg Pred Value         1.0000   1.0000   0.9989   1.0000
Prevalence             0.2633   0.2856   0.2687   0.1824
Detection Rate         0.2633   0.2856   0.2679   0.1824
Detection Prevalence   0.2633   0.2856   0.2679   0.1832
Balanced Accuracy      1.0000   1.0000   0.9986   0.9995
> knn_CM3
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3   4
         0 485   2   0   2   1
         1   0 538   3   0   0
         2   0   0 391   0   1
         3   0   0   0 675   2
```

```
         4   0   0   0   0 271

Overall Statistics

               Accuracy : 0.9954
                 95% CI : (0.9917, 0.9977)
    No Information Rate : 0.2855
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9941
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            1.0000   0.9963   0.9924   0.9970   0.9855
Specificity            0.9973   0.9984   0.9995   0.9988   1.0000
Pos Pred Value         0.9898   0.9945   0.9974   0.9970   1.0000
Neg Pred Value         1.0000   0.9989   0.9985   0.9988   0.9981
Prevalence             0.2046   0.2278   0.1662   0.2855   0.1160
Detection Rate         0.2046   0.2269   0.1649   0.2847   0.1143
Detection Prevalence   0.2067   0.2282   0.1653   0.2855   0.1143
Balanced Accuracy      0.9987   0.9973   0.9959   0.9979   0.9927
>
>
>
>
> knn_1 <- train((FLOOR ~.), data = train_set1, method = "knn", trControl =
ctrl, tuneLength = 5)
Loading required package: knn

Attaching package: ‘knn’

The following object is masked from ‘package:caret’:

    contr.dummy

> knn_2 <- train((FLOOR ~.), data = train_set2, method = "knn", trControl =
ctrl, tuneLength = 5)
> knn_3 <- train((FLOOR ~.), data = train_set3, method = "knn", trControl =
ctrl, tuneLength = 5)
> knn_Predict1 <- predict(knn_1, newdata = test_set1)
> knn_Predict2 <- predict(knn_2, newdata = test_set2)
> knn_Predict3 <- predict(knn_3, newdata = test_set3)
> knn_CM1 <- confusionMatrix(knnPredict1, test_set1$FLOOR)
Error in confusionMatrix(knnPredict1, test_set1$FLOOR) :
  object 'knnPredict1' not found
> knn_CM1 <- confusionMatrix(knn_Predict1, test_set1$FLOOR)
> knn_CM2 <- confusionMatrix(knn_Predict2, test_set2$FLOOR)
> knn_CM3 <- confusionMatrix(knn_Predict3, test_set3$FLOOR)
>
>
> knn_1
k-Nearest Neighbors

3939 samples
 200 predictor
```

```
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3544, 3546, 3547, 3545, 3544, 3544, ...
Resampling results across tuning parameters:

  kmax  Accuracy   Kappa
   5    0.9824846  0.9765502
   7    0.9845176  0.9792678
   9    0.9850266  0.9799492
  11    0.9850266  0.9799492
  13    0.9850266  0.9799492

Tuning parameter 'distance' was held constant at a value of 2
Tuning parameter 'kernel' was held constant at a value of optimal
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were kmax = 13, distance = 2 and kernel =
optimal.
> kknn_2
k-Nearest Neighbors

3897 samples
 207 predictor
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 3508, 3508, 3508, 3507, 3508, 3507, ...
Resampling results across tuning parameters:

  kmax  Accuracy   Kappa
   5    0.9935877  0.9913787
   7    0.9935877  0.9913787
   9    0.9935877  0.9913787
  11    0.9935877  0.9913787
  13    0.9935877  0.9913787

Tuning parameter 'distance' was held constant at a value of 2
Tuning parameter 'kernel' was held constant at a value of optimal
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were kmax = 13, distance = 2 and kernel =
optimal.
> kknn_3
k-Nearest Neighbors

7121 samples
 203 predictor
   5 classes: '0', '1', '2', '3', '4'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 6410, 6408, 6409, 6409, 6409, 6409, ...
Resampling results across tuning parameters:

  kmax  Accuracy   Kappa
   5    0.9827279  0.9779693
```

```
   7    0.9834301  0.9788638
   9    0.9835707  0.9790432
  11    0.9835707  0.9790432
  13    0.9835707  0.9790432

Tuning parameter 'distance' was held constant at a value of 2
Tuning parameter 'kernel' was held constant at a value of optimal
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were kmax = 13, distance = 2 and kernel =
optimal.
>
>
> rf_1 <- train((FLOOR ~.), data = train_set1, method = "rf", trControl = ctr
l, tuneLength = 5)
Loading required package: randomForest
randomForest 4.6-12
Type rfNews() to see new features/changes/bug fixes.

Attaching package: ‘randomForest’

The following object is masked from ‘package:ggplot2’:

    margin

The following object is masked from ‘package:dplyr’:

    combine

> rf_2 <- train((FLOOR ~.), data = train_set2, method = "rf", trControl = ctr
l, tuneLength = 5)
> rf_3 <- train((FLOOR ~.), data = train_set3, method = "rf", trControl = ctr
l, tuneLength = 5)
> rf_Predict1 <- predict(rf_1, newdata = test_set1)
> rf_Predict2 <- predict(rf_2, newdata = test_set2)
> rf_Predict3 <- predict(rf_3, newdata = test_set3)
> rf_CM1 <- confusionMatrix(rf_Predict1, test_set1$FLOOR)
> rf_CM2 <- confusionMatrix(rf_Predict2, test_set2$FLOOR)
> rf_CM3 <- confusionMatrix(rf_Predict3, test_set3$FLOOR)
>
>
> rf_CM1
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 264   1   0   0
         1   0 337   1   0
         2   0   1 359   0
         3   0   0   0 347

Overall Statistics

               Accuracy : 0.9977
                 95% CI : (0.9933, 0.9995)
    No Information Rate : 0.2748
    P-Value [Acc > NIR] : < 2.2e-16
```

```
                 Kappa : 0.9969
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9941   0.9972   1.0000
Specificity            0.9990   0.9990   0.9989   1.0000
Pos Pred Value         0.9962   0.9970   0.9972   1.0000
Neg Pred Value         1.0000   0.9979   0.9989   1.0000
Prevalence             0.2015   0.2588   0.2748   0.2649
Detection Rate         0.2015   0.2573   0.2740   0.2649
Detection Prevalence   0.2023   0.2580   0.2748   0.2649
Balanced Accuracy      0.9995   0.9965   0.9981   1.0000
> rf_CM2
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 342   1   0   0
         1   0 370   0   0
         2   0   0 349   1
         3   0   0   0 236

Overall Statistics

               Accuracy : 0.9985
                 95% CI : (0.9944, 0.9998)
    No Information Rate : 0.2856
    P-Value [Acc > NIR] : < 2.2e-16

                 Kappa : 0.9979
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9973   1.0000   0.9958
Specificity            0.9990   1.0000   0.9989   1.0000
Pos Pred Value         0.9971   1.0000   0.9971   1.0000
Neg Pred Value         1.0000   0.9989   1.0000   0.9991
Prevalence             0.2633   0.2856   0.2687   0.1824
Detection Rate         0.2633   0.2848   0.2687   0.1817
Detection Prevalence   0.2640   0.2848   0.2694   0.1817
Balanced Accuracy      0.9995   0.9987   0.9995   0.9979
> rf_CM3
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3   4
         0 485   1   0   0   0
         1   0 539   1   0   0
         2   0   0 393   0   2
         3   0   0   0 677   1
         4   0   0   0   0 272

Overall Statistics
```

```
               Accuracy : 0.9979
                 95% CI : (0.9951, 0.9993)
    No Information Rate : 0.2855
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9973
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            1.0000   0.9981   0.9975   1.0000   0.9891
Specificity            0.9995   0.9995   0.9990   0.9994   1.0000
Pos Pred Value         0.9979   0.9981   0.9949   0.9985   1.0000
Neg Pred Value         1.0000   0.9995   0.9995   1.0000   0.9986
Prevalence             0.2046   0.2278   0.1662   0.2855   0.1160
Detection Rate         0.2046   0.2273   0.1658   0.2855   0.1147
Detection Prevalence   0.2050   0.2278   0.1666   0.2860   0.1147
Balanced Accuracy      0.9997   0.9988   0.9982   0.9997   0.9945
>
>
>
>
>
> C50_1 <- train((FLOOR ~.), data = train_set1, method = "c50", trControl = c
trl, tuneLength = 5)
Error in train.default(x, y, weights = w, ...) :
  Model c50 is not in caret's built-in library
> C50_1 <- train((FLOOR ~.), data = train_set1, method = "c5.0", trControl =
ctrl, tuneLength = 5)
Error in train.default(x, y, weights = w, ...) :
  Model c5.0 is not in caret's built-in library
> C50_1 <- train((FLOOR ~.), data = train_set1, method = "C5.0", trControl =
ctrl, tuneLength = 5)
Loading required package: C50
> C50_2 <- train((FLOOR ~.), data = train_set2, method = "C5.0", trControl =
ctrl, tuneLength = 5)
> C50_3 <- train((FLOOR ~.), data = train_set3, method = "C5.0", trControl =
ctrl, tuneLength = 5)
> C50_Predict1 <- predict(C50_1, newdata = test_set1)
> C50_Predict2 <- predict(C50_2, newdata = test_set2)
> C50_Predict3 <- predict(C50_3, newdata = test_set3)
> C50_CM1 <- confusionMatrix(C50_1, test_set1$FLOOR)
Error in match.arg(norm, c("none", "overall", "average")) :
  'arg' must be NULL or a character vector
> C50_CM1 <- confusionMatrix(C50_Predict1, test_set1$FLOOR)
> C50_CM2 <- confusionMatrix(C50_Predict2, test_set2$FLOOR)
> C50_CM3 <- confusionMatrix(C50_Predict3, test_set3$FLOOR)
> C50_CM1
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 263   1   0   0
         1   1 338   2   0
         2   0   0 357   0
```

```
         3   0   0   1 347

Overall Statistics

               Accuracy : 0.9962
                 95% CI : (0.9911, 0.9988)
    No Information Rate : 0.2748
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9949
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.9962   0.9971   0.9917   1.0000
Specificity            0.9990   0.9969   1.0000   0.9990
Pos Pred Value         0.9962   0.9912   1.0000   0.9971
Neg Pred Value         0.9990   0.9990   0.9969   1.0000
Prevalence             0.2015   0.2588   0.2748   0.2649
Detection Rate         0.2008   0.2580   0.2725   0.2649
Detection Prevalence   0.2015   0.2603   0.2725   0.2656
Balanced Accuracy      0.9976   0.9970   0.9958   0.9995
> C50_CM2
Confusion Matrix and Statistics

          Reference
Prediction   0   1   2   3
         0 342   3   0   0
         1   0 368   0   0
         2   0   0 346   0
         3   0   0   3 237

Overall Statistics

               Accuracy : 0.9954
                 95% CI : (0.99, 0.9983)
    No Information Rate : 0.2856
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9938
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            1.0000   0.9919   0.9914   1.0000
Specificity            0.9969   1.0000   1.0000   0.9972
Pos Pred Value         0.9913   1.0000   1.0000   0.9875
Neg Pred Value         1.0000   0.9968   0.9969   1.0000
Prevalence             0.2633   0.2856   0.2687   0.1824
Detection Rate         0.2633   0.2833   0.2664   0.1824
Detection Prevalence   0.2656   0.2833   0.2664   0.1848
Balanced Accuracy      0.9984   0.9960   0.9957   0.9986
> C50_CM3
Confusion Matrix and Statistics
```

```
          Reference
Prediction   0   1   2   3   4
         0 485   1   0   0   0
         1   0 539   0   0   0
         2   0   0 394   0   1
         3   0   0   0 677   0
         4   0   0   0   0 274

Overall Statistics

               Accuracy : 0.9992
                 95% CI : (0.997, 0.9999)
    No Information Rate : 0.2855
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9989
 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3 Class: 4
Sensitivity            1.0000   0.9981   1.0000   1.0000   0.9964
Specificity            0.9995   1.0000   0.9995   1.0000   1.0000
Pos Pred Value         0.9979   1.0000   0.9975   1.0000   1.0000
Neg Pred Value         1.0000   0.9995   1.0000   1.0000   0.9995
Prevalence             0.2046   0.2278   0.1662   0.2855   0.1160
Detection Rate         0.2046   0.2273   0.1662   0.2855   0.1156
Detection Prevalence   0.2050   0.2273   0.1666   0.2855   0.1156
Balanced Accuracy      0.9997   0.9991   0.9997   1.0000   0.9982
```