C3T3 Evaluation of techniques for Wi-Fi Locationing

The goal of this work is to investigate the feasibility of using "Wi-Fi fingerprinting" to determine location inside complex buildings. In this case a large database containing Wi-Fi finger prints for a multi building campus (building, floor, location within floor ID) associated with each finger print is analyzed. The approach is to first study the data and determine if classification or regression learning will be applied. In this case classification is appropriate; the data was then preprocessed by removing unnecessary attributes including attributes with zero variance. The data set is large and includes data from three buildings of a university; thus, it was found feasible to separate the data set by building, and combine location attributes into one single attribute as an identifier. The identifier was then used as the dependent variable to create machine learning models by building. Three machine learning models were created for each building and described below.

Objective:

To evaluate multiple machine learning models to determine which produces best location predictions from Wi-Fi fingerprints. This will allow us to make a recommendation to the client. If the accuracy of the selected model is high enough it will be applied into an app for indoor locationing.

Data Set Description:

A large data set with approximately 20,000 observations and 529 variables was provided. The data contains Wi-Fi access point readings (finger prints) for an industrial campus containing 3 buildings. The location of the finger prints are accompanied by the building, floor, relative position and location ID.

Data Sources is the [UCI Machine Learning Repository]

- The data encompasses three buildings with 4 or more floors and almost 110.000m2
- The data was collected in 2013 using more than 20 different users and 25 Android devices.
- The database consists of 19937 training records and 1111 validation/test records.

Attributes Information

- WAP001- WAP520: Intensity reading value for WAP. Negative integer values from -104 to 0 and +100. Positive value 100 used if WA was not detected.
- Longitude and Latitude readings
- Floor, Building ID, Space ID (office, corridor, classroom)
- Relative Position (Inside or outside of the space)
- UserID, PhoneID, TimeStamp

 # uploading the data
library(readr)
trdata <- read.csv("trainingData.csv")
validata <- read.csv("validationData.csv")

#set up for parralel processing
library(doParallel)
#find how many cores are on your machine
detectCores()
#create cluster with desired number of cores
cl<- makeCluster(2)
#Register cluster
registerDoParallel(cl)
 # confirm how many cores are now assinged to R and R studio
getDoParWorkers()
 # stop cluster)after performing all tasks...
stopCluster(cl)

 # Data at a Glance

head(trdata, n=3)[1:10]

```
  WAP001 WAP002 WAP003 WAP004 WAP005 WAP006 WAP007 WAP008 WAP009 WAP010
1    100    100    100    100    100    100    100    100    100    100
2    100    100    100    100    100    100    100    100    100    100
3    100    100    100    100    100    100    100    -97    100    100
```

tail(trdata, n=5)[519:529]

```
      WAP519 WAP520 LONGITUDE LATITUDE FLOOR BUILDINGID SPACEID RELATIVEPOSIT
ION USERID PHONEID  TIMESTAMP
19933    100    100 -7485.469  4864875     3          1       1
2     18       10 1371710683
19934    100    100 -7390.621  4864836     1          2     140
2     18       10 1371710402
19935    100    100 -7516.841  4864889     3          1      13
2     18       10 1371710921
19936    100    100 -7537.322  4864896     3          1     113
2     18       10 1371711049
19937    100    100 -7536.166  4864898     3          1     112
2     18       10 1371711025
```

 # check for data structure

str(trdata)
str(trdata$LATITUDE)
str(trdata$LONGITUDE)
str(trdata$FLOOR)
str(trdata$BUILDINGID)
str(trdata$SPACEID)
str(trdata$RELATIVEPOSITION)
str(trdata$USERID)
str(trdata$PHONEID)

 # convert location attributes to factors

trdata$FLOOR <- as.factor(trdata$FLOOR)
trdata$BUILDINGID <- as.factor(trdata$BUILDINGID)
trdata$SPACEID <- as.factor(trdata$SPACEID)
trdata$RELATIVEPOSITION <- as.factor(trdata$RELATIVEPOSITION)
trdata$USERID <- as.factor(trdata$USERID)
trdata$PHONEID <- as.factor(trdata$PHONEID)

 # determine if there is 0 variance in variables
library(caret)
library(dplyr)
library(tibble)

trdata_0var <- nearZeroVar(trdata, saveMetrics= TRUE)
zero_vals_trdata <- select(trdata_0var, zeroVar)
table(zero_vals_trdata)

```
table(zero_vals_trdata)
zero_vals_trdata
FALSE  TRUE
  474    55
```

there are 55 variables that have near 0 variance
determine which atributes have zero variance
trdata_0var[trdata_0var[, "zeroVar"] > 0,]

```
trdata_0var[trdata_0var[, "zeroVar"] > 0,]
       freqRatio percentUnique zeroVar  nzv
WAP003         0     0.0050158    TRUE TRUE
WAP004         0     0.0050158    TRUE TRUE
WAP092         0     0.0050158    TRUE TRUE
WAP093         0     0.0050158    TRUE TRUE
WAP094         0     0.0050158    TRUE TRUE
WAP095         0     0.0050158    TRUE TRUE
WAP152         0     0.0050158    TRUE TRUE
WAP158         0     0.0050158    TRUE TRUE
WAP159         0     0.0050158    TRUE TRUE
WAP160         0     0.0050158    TRUE TRUE
WAP215         0     0.0050158    TRUE TRUE
WAP217         0     0.0050158    TRUE TRUE
WAP226         0     0.0050158    TRUE TRUE
WAP227         0     0.0050158    TRUE TRUE
WAP238         0     0.0050158    TRUE TRUE
WAP239         0     0.0050158    TRUE TRUE
WAP240         0     0.0050158    TRUE TRUE
WAP241         0     0.0050158    TRUE TRUE
WAP242         0     0.0050158    TRUE TRUE
WAP243         0     0.0050158    TRUE TRUE
WAP244         0     0.0050158    TRUE TRUE
WAP245         0     0.0050158    TRUE TRUE
WAP246         0     0.0050158    TRUE TRUE
WAP247         0     0.0050158    TRUE TRUE
WAP254         0     0.0050158    TRUE TRUE
WAP293         0     0.0050158    TRUE TRUE
WAP296         0     0.0050158    TRUE TRUE
WAP301         0     0.0050158    TRUE TRUE
```

```
WAP303          0      0.0050158    TRUE TRUE
WAP304          0      0.0050158    TRUE TRUE
WAP307          0      0.0050158    TRUE TRUE
WAP333          0      0.0050158    TRUE TRUE
WAP349          0      0.0050158    TRUE TRUE
WAP353          0      0.0050158    TRUE TRUE
WAP360          0      0.0050158    TRUE TRUE
WAP365          0      0.0050158    TRUE TRUE
WAP416          0      0.0050158    TRUE TRUE
WAP419          0      0.0050158    TRUE TRUE
WAP423          0      0.0050158    TRUE TRUE
WAP429          0      0.0050158    TRUE TRUE
WAP433          0      0.0050158    TRUE TRUE
WAP438          0      0.0050158    TRUE TRUE
WAP441          0      0.0050158    TRUE TRUE
WAP442          0      0.0050158    TRUE TRUE
WAP444          0      0.0050158    TRUE TRUE
WAP445          0      0.0050158    TRUE TRUE
WAP451          0      0.0050158    TRUE TRUE
WAP458          0      0.0050158    TRUE TRUE
WAP482          0      0.0050158    TRUE TRUE
WAP485          0      0.0050158    TRUE TRUE
WAP487          0      0.0050158    TRUE TRUE
WAP488          0      0.0050158    TRUE TRUE
WAP491          0      0.0050158    TRUE TRUE
WAP497          0      0.0050158    TRUE TRUE
WAP520          0      0.0050158    TRUE TRUE
```

remove attributes with 0 variance & other unecessary attrubutes by name

```
trdata_cleaned = subset(trdata, select = -c(WAP003, WAP004, WAP092, WAP093,
                        WAP094, WAP095, WAP152, WAP158,
                        WAP159, WAP160, WAP215, WAP217,
                        WAP226, WAP227, WAP238, WAP239,
                        WAP240, WAP241, WAP242, WAP243,
                        WAP244, WAP245, WAP246, WAP247,
                        WAP254, WAP293, WAP296, WAP301,
                        WAP303, WAP304, WAP307, WAP333,
                        WAP349, WAP353, WAP360, WAP365,
                        WAP416, WAP419, WAP423, WAP429,
                        WAP433, WAP438, WAP441, WAP442,
                        WAP444, WAP445, WAP451, WAP458,
                        WAP482, WAP485, WAP487, WAP488,
                        WAP491, WAP497, WAP520,USERID,
                        PHONEID, TIMESTAMP, LATITUDE,
                        LONGITUDE ))
```

#another way to find and remove zerovariance variables:
 # nearzeroVar () with saveMetrics = FALSE retruns a vector which can then # be used to easily and quckily remove the zero variance features.

```
#create nzv vector
nvz <- nearZeroVar(trdata, saveMetrics = FALSE)
nvz
  # create new data set and remove nearzero variace features and create new dataset

trdataNZV <- trdata[,-nvz]
str(trdataNZV)
```

Seperate Data by building using filter function from dplyr

```
trData_buidgID_0 <- filter(trdata_cleaned, BUILDINGID == 0)
trData_buidgID_1 <- filter(trdata_cleaned, BUILDINGID == 1)
trData_buidgID_2 <- filter(trdata_cleaned, BUILDINGID == 2)
```

Unite location attributes (Building, Floor, SpaceID, Relativeposition) into one LOCATION attribute using tidyr

```
library(tidyr)
trData_buidgID_0_combLoc <- unite(trData_buidgID_0, BUILDINGID, FLOOR, SPACEID,
RELATIVEPOSITION, col = LOCATION)
trData_buidgID_1_combLoc <- unite(trData_buidgID_1, BUILDINGID, FLOOR, SPACEID,
RELATIVEPOSITION, col = LOCATION)
trData_buidgID_2_combLoc <- unite(trData_buidgID_2, BUILDINGID, FLOOR, SPACEID,
RELATIVEPOSITION, col = LOCATION)
```

 # check data type for new LOCATION attribute

```
str(trData_buidgID_0_combLoc$LOCATION)
str(trData_buidgID_1_combLoc$LOCATION)
str(trData_buidgID_2_combLoc$LOCATION)
```

 # convert new Location attribute from chr data type to factor for classification

```
trData_buidgID_0_combLoc$LOCATION <- as.factor(trData_buidgID_0_combLoc$LOCATION)
trData_buidgID_1_combLoc$LOCATION <- as.factor(trData_buidgID_1_combLoc$LOCATION)
trData_buidgID_2_combLoc$LOCATION <- as.factor(trData_buidgID_2_combLoc$LOCATION)
```

 # confirm data conversion to factor

```
str(trData_buidgID_0_combLoc$LOCATION)
str(trData_buidgID_1_combLoc$LOCATION)
str(trData_buidgID_2_combLoc$LOCATION)
```

CREATE AND TEST CLASSIFICATION MODELS
The task is to predict the detailed location based of WAPs. The location is not a consecutive value thus a classification model is in order for this task. The Tree classifiers selected are C5.0, Random Forest and KNN. 10 fold cross validation is applied to avoid overfitting.

C5.0 and Random Forest

C5.0 and Random Forest algorithms belong to the tree model family. Tree models are a flowchart-like model that works by splitting the sample based on the maximum informative variable, named nodes. Each nodes will then split again, the process repeats until the subsamples cannot be split any further.

C5.0 is strong at processing a large number of variables, It is relatively fast in the creation of the decision tree and automatic post pruning of branches and nodes that have little or no effect on the classification errors.

Random Forest usually requires a longer time to be trained, depending on the numbers of trees. The algorithm constructs multiple decision trees and makes the output the mode of the classes for classification problem. That is , the trees sort of vote on the classification classes. This provides the advantage of Random forests avoids overfitting over simple decision tree model.

KNN

K-Nearest Neighbor algorithm is based on the assumption that similar things exist near each other. It captures similarity by calculating the distance between points. KNN is simple and relatively easy to implement, but it could become time consuming in larger datasets.

CREATE CLASSIFICATION MODELS

```
# set seed and create 10-fold cross validation fit control
set.seed(123)
fitControl_0 <- trainControl(method= "repeatedcv", number=10,repeats = 1)

#Building 0

# create training and testing sets
inTrain_0 <- createDataPartition( y = trDataa_buidgID_0_combLoc$LOCATION,p=0.75, list =  FALSE)
str(inTrain_0)
training_0 <- trData_buidgID_0_combLoc [inTrain_0,]
testing_0 <- trData_buidgID_0_combLoc [-inTrain_0,]
nrow(training_0)
nrow(testing_0)

## create C5.0 model for classification for trData_buidgID_0combLoc using LOCATION as dependent variable
library(C50)
C5.0model_0<- train (LOCATION~.,data = training_0, method = "C5.0", trControl=fitControl_0)
C5.0model_0
varImp(C5.0model_0)
```

 # prediction C5.0model_0
predictions_C50_0 <- predict(C5.0model_0,testing_0)

Note:
```
#note the system time wrapper. system.time()
#this is used to measure process execution time
system.time(C5.0model_0<- train (LOCATION~.,data = training_0, method = "C5.0", trControl=fitControl_0))
```

 # Random Forest for building 0
rfGrid_0 <- expand.grid(mtry=c(1,2,3,4,5))

RF_model_0B <- train(LOCATION~., data = training_0, method = "rf",trControl=fitControl_0, tunelength = rfGrid_0)
RF_model_0B

RF_model_0 <- train(LOCATION~., data = training_0, method = "rf",trControl=fitControl_0, tunelength =5)
RF_model_0
 # make predictions using testing data and RF_model_0
predictions_RF_0 <- predict(RF_model_0,testing_0)

#knn model for building 0
Knn_0 <- train(LOCATION~., data = training_0, method = "knn", trControl=fitControl_0)
Knn_0
predictions_knn_0 <- predict(Knn_0,testing_0)

 # Building #1

# create training and testing sets
inTrain_1 <- createDataPartition(y = trData_buidgID_1_combLoc$LOCATION,p=0.75, list = FALSE)
training_1 <- trData_buidgID_1_combLoc [inTrain_1,]
testing_1 <- trData_buidgID_1_combLoc [-inTrain_1,]
nrow(training_1)
nrow(testing_1)

create C5.0 model for classification for trData_buidgID_1_combLoc using LOCATION as dependent variable

C5.0model_1<- train (LOCATION~.,data = training_1, method = "C5.0", trControl=fitControl_0)
C5.0model_1
varImp(C5.0model_1)

 # prediction C5.0model_1
predictions_C50_1 <- predict(C5.0model_1,testing_1)

 # Random Forest for building 1
rfGrid_0 <- expand.grid(mtry=c(1,2,3,4,5))
RF_model_1b <- train(LOCATION~., data = training_1, method = "rf",trControl=fitControl_0, tunelength = rfGrid_0)
RF_model_1b

RF_model_1 <- train(LOCATION~., data = training_1, method = "rf",trControl=fitControl_0)
RF_model_1
 # make predictions usign testing data and RF_model_0
predictions_RF_1 <- predict(RF_model_1,testing_1)

#knn model for building 1
Knn_1 <- train(LOCATION~., data = training_1, method = "knn", trControl=fitControl_0)
Knn_1
 # Make predictions using knn_0 model and testing data
predictions_knn_1 <- predict(Knn_1,testing_1)

#Building #2
create C5.0 model for classification for trData_buidgID_2_combLoc using LOCATION as dependent variable

C5.0model_2<- train (LOCATION~.,data = training_2, method = "C5.0", trControl=fitControl_0)

C5.0model_2
varImp(C5.0model_2)

 # prediction C5.0model_2
predictions_C50_2 <- predict(C5.0model_2,testing_2)

 # Random Forest for building 2
RF_model_2 <- train(LOCATION~., data = training_2, method = "rf",trControl=fitControl_0)
RF_model_2
# make predictions usign testing data and RF_model_2
predictions_RF_2 <- predict(RF_model_2,testing_2)

#knn model for building 2
Knn_2 <- train(LOCATION~., data = training_2, method = "knn", trControl=fitControl_0)
Knn_2
# Make predictions using knn_2 model and testing data
predictions_knn_2 <- predict(Knn_2,testing_2)

Evaluating the Machine learning Models
There are different ways to compare machine learning model performance. In this case confusion matrix, postResample, resample methods will be applied. Both Accuracy and Kappa Score will be looled at to evaluate our models.
Kappa Score: Kappa Score compares an Observed Accuracy with an Expected Accuracy. Observed Accuracy is simply the number of instances that were classified correctly while Expected Accuracy is defined as the accuracy that any random classifier would be expected to achieve. In general Kappa Score is less misleading than simply using accuracy.

 # Evaluating the models
 # Building 0

#evaluate C5.0model_0
cm_C5.0model_0 <- confusionMatrix(predictions_C50_0, testing_0$LOCATION)
postResample(predictions_C50_0, testing_0$LOCATION)

```
Accuracy     Kappa
0.7222666 0.7211349
```

#evaluate rf_model_0
cm_rf_model_0 <- confusionMatrix(predictions_RF_0, testing_0$LOCATION)
postResample(predictions_RF_0, testing_0$LOCATION)

```
Accuracy     Kappa
0.7701516 0.7692116
```

#evaluate knn_0
cm_knn_0 <- confusionMatrix(predictions_knn_0, testing_0$LOCATION)
postResample(predictions_knn_0, testing_0$LOCATION)

```
Accuracy     Kappa
0.5466879 0.5448494
```

 # resample to obtain accuracy and kappa for all three models
resample_0 <- resamples(list(C5.0 = C5.0model_0, RF = RF_model_0, Knn = Knn_0))
summary(resample_0)

Evaluation of models for building_0							
Acurracy							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.66	0.68	0.69	0.7	0.72	0.74	0
RF	0.72	0.75	0.76	0.75	0.77	0.79	0
Knn	0.049	0.53	0.56	0.55	0.53	0.58	0
kappa							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.66	0.68	0.69	0.7	0.71	0.74	0
RF	0.71	0.75	0.76	0.76	0.77	0.79	0
Knn	0.49	0.53	0.56	0.56	0.57	0.58	0

 # Building 1
#evaluate C5.0model_1
cm_C5.0model_1 <- confusionMatrix(predictions_C50_1, testing_1$LOCATION)
postResample(predictions_C50_1, testing_1$LOCATION)

```
Accuracy     Kappa
0.7962662 0.7950263
```

#evaluate rf_model_1
cm_rf_model_1 <- confusionMatrix(predictions_RF_1, testing_1$LOCATION)
postResample(predictions_RF_1, testing_1$LOCATION)

```
Accuracy     Kappa
0.8603896 0.8595299
```

#evaluate knn_1
cm_knn_1 <- confusionMatrix(predictions_knn_1, testing_1$LOCATION)
postResample(predictions_knn_1, testing_1$LOCATION)

```
Accuracy     Kappa
0.6501623 0.6480264
```

# resample to obtain accuracy and kappa for all three models
resample_1 <- resamples(list(C5.0 = C5.0model_1, RF = RF_model_1, Knn = Knn_1))
summary(resample_1)

Evaluation of models for building_1							
Acurracy							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.76	0.79	0.8	0.8	0.82	0.83	0
RF	0.83	0.84	0.85	0.85	0.86	0.86	0
Knn	0.6	0.62	0.64	0.63	0.64	0.66	0
kappa							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.76	0.79	0.8	0.8	0.82	0.82	0
RF	0.83	0.84	0.85	0.85	0.86	0.86	0
Knn	0.6	0.61	0.64	0.63	0.64	0.66	0

 # Building 2
#evaluate C5.0model_2
cm_C5.0model_2 <- confusionMatrix(predictions_C50_2, testing_2$LOCATION)
postResample(predictions_C50_2, testing_2$LOCATION)

```
Accuracy     Kappa
0.7358907 0.7350637
```

#evaluate rf_model_2
cm_rf_model_2 <- confusionMatrix(predictions_RF_2, testing_2$LOCATION)
postResample(predictions_RF_2, testing_2$LOCATION)

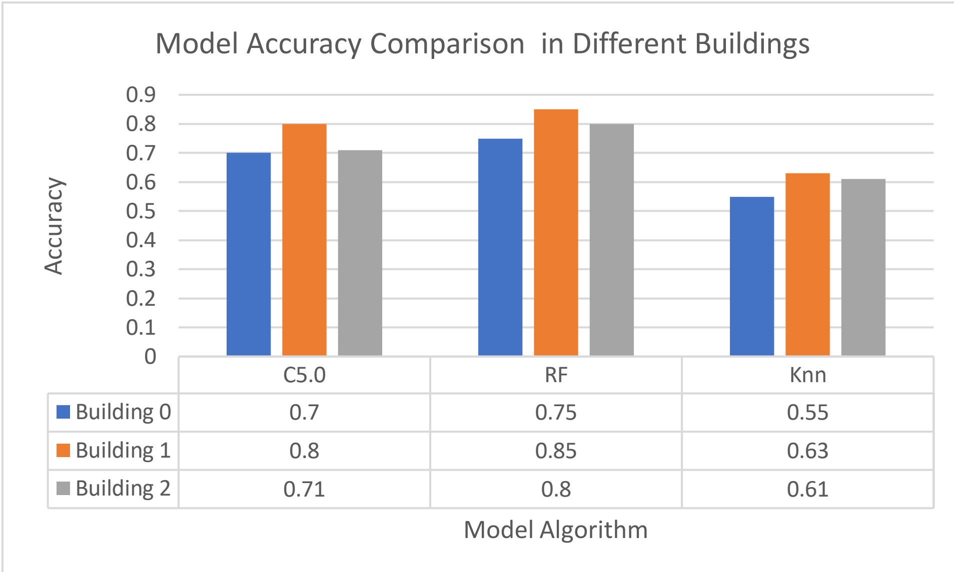
```
Accuracy     Kappa
0.8196649 0.8190941
```

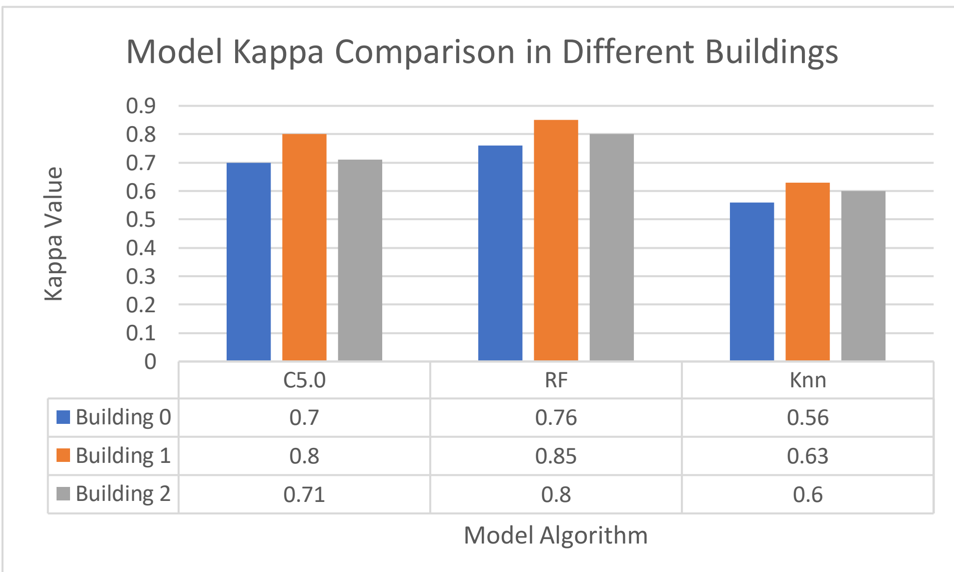
#evaluate knn_2
cm_knn_2 <- confusionMatrix(predictions_knn_2, testing_2$LOCATION)
postResample(predictions_knn_2, testing_2$LOCATION)

```
Accuracy     Kappa
0.6499118 0.6488031
```

# resample to obtain accuracy and kappa for all three models
resample_2 <- resamples(list(C5.0 = C5.0model_2, RF = RF_model_2, Knn = Knn_2))
summary(resample_2)

Evaluation of models for building_2							
Acurracy							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.69	0.69	0.71	0.71	0.72	0.74	0
RF	0.78	0.79	0.8	0.8	0.81	0.83	0
Knn	0.58	0.61	0.61	0.61	0.62	0.63	0
kappa							
	Min.	**1st Qu.**	**Median**	**Mean**	**3rd Qu.**	**Max**	**NA's**
C5.0	0.69	0.69	0.71	0.71	0.72	0.74	0
RF	0.78	0.79	0.8	0.8	0.81	0.83	0
Knn	0.57	0.59	0.61	0.6	0.62	0.63	0





Model Training time Comparison:

Model Name	Average Training Time (seconds)
C5.0	1800
RF	11000
KNN	500

Model Selection:

Upon evaluation of our models it was determined that the KNN algorithm does not perform well for this application. Both C5.0 and Random Forest algorithms are compatible at the accuracy and kappa values. In this case it is beneficial to apply the C5.0 model based on the training time of the model approximately 0.5 hours vs 3.0 hours.

Further Business recommendations: Combine Wi-Fi and Bluetooth Beacon technology

A Wi-Fi-based system, consists Wi-Fi transmitters which send information to multiple Wi-Fi access points (WAPs). These WAPs report the time and strength of the information reading to a computer, which uses algorithms to determine position. The calculated position information is then stored in a cloud. The use of time difference of arrival (TDOA) measurement allows for Wi-Fi indoor positioning systems to have an accuracy of three to five meters. A major drawback of this system is that to achieve this accuracy least three access points need to “hear” each transmitted information.

Additionally, a second system can be used in conjunction to the Wi-Fi Based system. Bluetooth Beacon technology is on the rise. This technology can be used for locationing and due to its space and distance limitations the accuracy for locationing is increased. Furthermore, Bluetooth technology is supported by main stream personal devices such as Android and iOS and respective devises can be installed without additional internal building wiring.