Real-Time Indoor Location Positioning

Justin Howard, Paul Adams, and Stuart Miller

September 8, 2020

1 Introduction

Real Time Location Systems (RTLS) that are capable of tracking business assets and people in special circumstances, are a popular area of research. Warehouse distribution and delivery services have grown more rapidly due to the COVID-19 pandemic, and with that growth, we can expect the relative importance of tracking assets at scale to increase. In this case study, we assess whether it is possible to predict locations based on the measurement of WiFi signals from fixed access points (AP)[1]. Additionally, two APs are placed in close proximity, we assess the impact of using both of these APs. The location sensing system analyzed in this case study is located in an indoor office environment on a single floor of a building.

2 Methods

2.1 Data

2.1.1 Data Collection

The data was collected in an indoor office environment on a single floor with 7 fixed WiFi APs. The WiFi APs are identifiable by their media-access-control (MAC) addresses. It is shown by Nolan and Lang (2015) that two of these APs (`00:0f:a3:39:e1:c0` and `00:0f:a3:39:dd:cd`) are positioned in close proximity. The experiment area was split into a grid 166 measurement points, which were spaced 1 meter apart. The points of measurement are referred to as `posX` and `posY`. Using a handheld WiFi scanning device, WiFi signal strength measurements were collected at each of the 166 locations. Additionally, the handheld WiFi scanning device was rotated to 8 angles (45°increments from 0) at each measurement location. The resulting dataset contains approximately one million measurements.

2.1.2 Data Cleaning

The data are provided as measurement logs with the MAC address of the scanning device, measurement position, measurement orientation, AP MAC address, AP signal strength, and timestamp on a single line. We extracted the data by reading each line and tokenizing each unit of information. The data was initially

[1]This analysis was conducted with the R programming language (R Core Team (2020)).

organized in a long format data frame with time, position X, Y, and Z, orientation, AP MAC address, signal strength, and channel as the columns. Once the data was organized, we were able to determine that several columns were not necessary. Specifically, we removed position Z and the scan MAC address because these were fixed throughout the experiment and channel because each AP used a single channel throughout the experiment. We also found that several MAC addresses were coded as *adhoc* (`type=1`). These were removed since we are only interested in positioning based on the fixed APs (`type=3`). Additionally, we determined that there some noise present in the orientation measurement. The experiment specified that only 8 angles were considered (each 45°increment from 0). All orientation values were rouned to the nearest 45°increment. Any missing signal values were filled with -100, imputing it as a very low strength signal.

2.2 Modeling Method

We chose to model the position as a function of the WiFi singal strength with k-nearest neighbors (k-NN). Intuitively, a k-NN model will predict a new position as the average positions of the closest k data points in the training data based on the WiFi signals. Since the features are microwaves, which can pass through physical barriers, we used euclidean distance to determine the closest signals. To use a k-NN, the value of k most be estimated based on the training data. We used 5-fold cross validation to determine the value of k, selecting the value of k that produced the lowest error based on root meean squared error (RMSE).

An advantage of k-NN is that it is non-parametric, thus it not necessary to assess whether the data exhibit certain distributional properties. However, a potential disadvantage of k-NN is the computational complexity, which is given by (1) for large N. This time complexity means a k-NN would not be suitable for modeling position in large areas as the number measurement points would be large.

$$O\left(n\right) \sim \frac{n^2}{2} \tag{1}$$

2.3 Model Validation

Predicting the real-time X- and Y-coordinate positions using K-Nearest Neighbors required the constructing an ensemble model. The first model we built contained all devices. The second model we built included all but the device excluding MAC address 00:0f:a3:39:dd:cd. Finally, we developed a third model which did not include the device identified as 00:0f:a3:39:e1:c0. We made this comparison because both of those explicitly mentioned devices were positioned at the same location within the study. We modeled along this process for both X- and Y-coordinate location prediction sets.

In order to simulate a measure of error that could be expected after applying to the data once the system is online (online data set), we applied cross-validation to a holdout data set, comprised of 25% of the offline data set. For this analysis, we trained the offline data set on 75%.

3 Results

Fig. 1 and Fig. 2 show the RMSE of k-NN models for X and Y positions, respectively. We found found that k of 5 minimized the error of the position X model and k of 4 minimized the error of the position Y model by cross-validation. We also found that using all APs provided the lowest overall error for both X position and Y position prediction. From the plots, we can also see that there is more error in the X position than the Y position prediction.

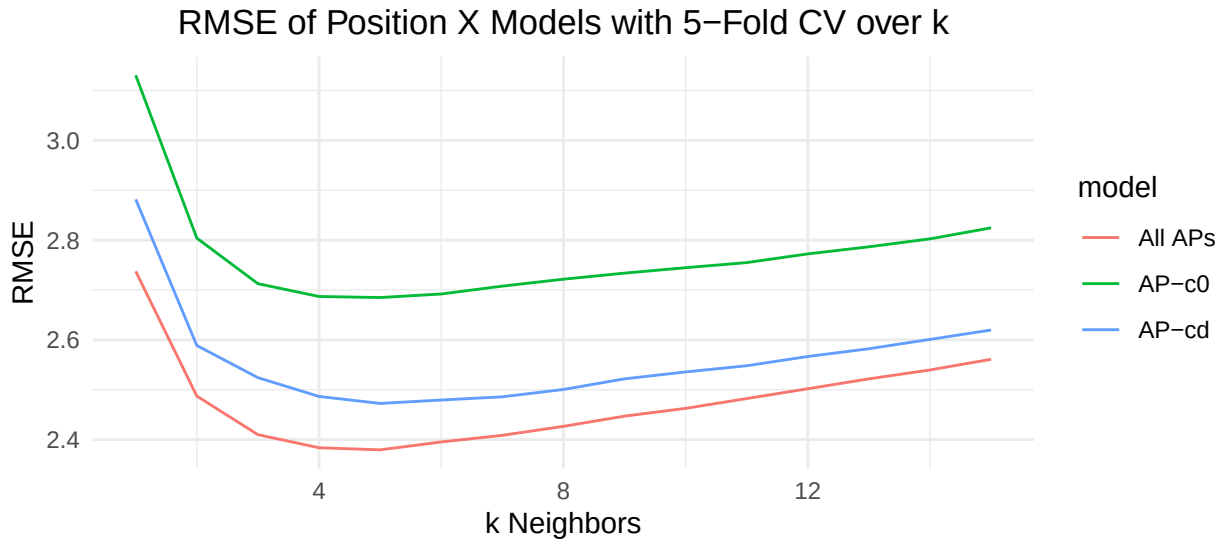

Figure 1:
The mean RMSE of 5−fold cross validation for k−NN models using all APs (All APs), all APs excluding 00:0f:a3:39:dd:cd (AP−c0),and all APs excluding 00:0f:a3:39:e1:c0 (AP−cd) for k values 1 through 15. This includes position X values only.

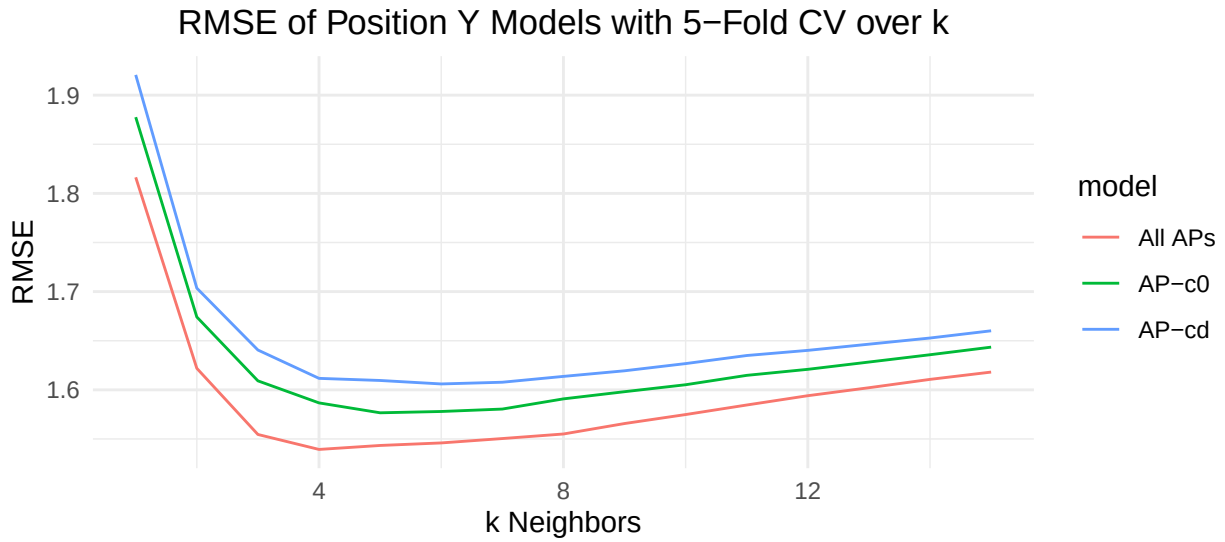

Figure 2:
The mean RMSE of 5−fold cross validation for k−NN models using all APs (All APs), all APs excluding 00:0f:a3:39:dd:cd (AP−c0),and all APs excluding 00:0f:a3:39:e1:c0 (AP−cd) for k values 1 through 15. This includes position Y values only.

Fig. 3 visualizes the RMSE of the combination of X and Y models. These results are consistent with the results for the individual X and Y position predictions. The model using all the APs performs best based on RMSE.

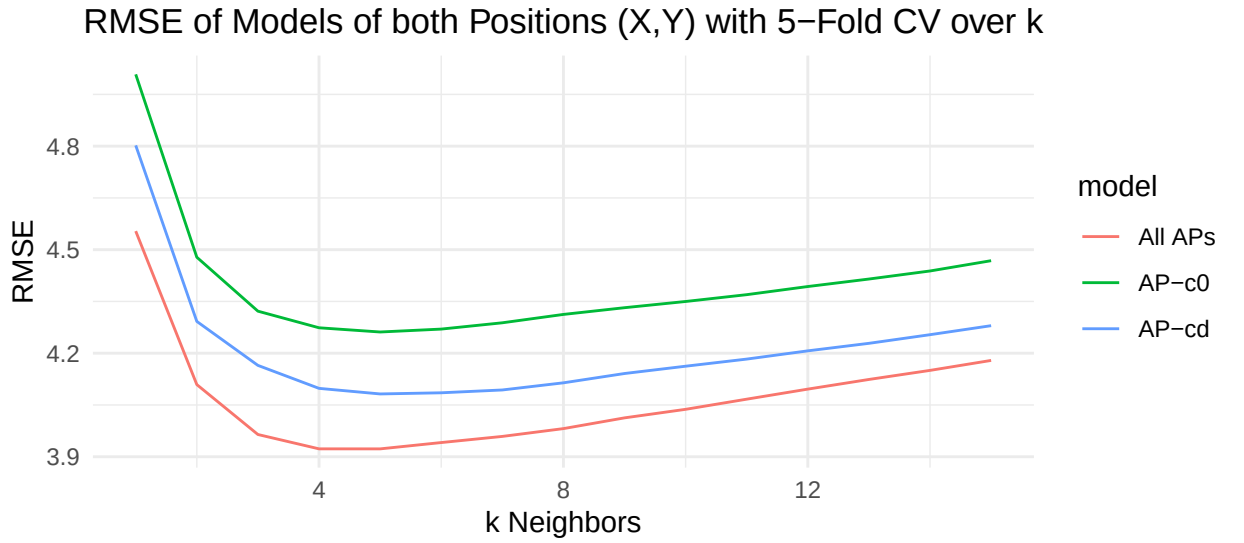

Figure 3:
The mean RMSE of 5−fold cross validation for k−NN models using all APs (All APs), all APs excluding 00:0f:a3:39:dd:cd (AP−c0),and all APs excluding 00:0f:a3:39:e1:c0 (AP−cd) for k values 1 through 15.

The estimated RMSE for the model of both positions applied to the online data set is roughly 7.042. We derived this value by taking the mean squared error for the residual values on the holdout sets for both the predicted X positions and predicted Y positions. A summary of the results for the holdout dataset is shown in Table 1. We see a small inflation of the error for model predictions on the online data. A summary of the results for the online dataset is shown in Table 2.

Table 1. Simulated Online Model Errors

Positional Predictions	Root Mean Squared Error
Predicted X	2.294
Predicted Y	1.521
Online X-Y Predictions	2.753

Table 2. Online Model Errors

Positional Predictions	Root Mean Squared Error
Predicted X	2.948
Predicted Y	2.194
Online X-Y Predictions	3.675

Fig. 4 shows location prediction (red) and actual positions (black) of measurement in the online dataset. There appears to be a large amount of error in the predictions. This figure shows that this model is not useful for indoor positioning.

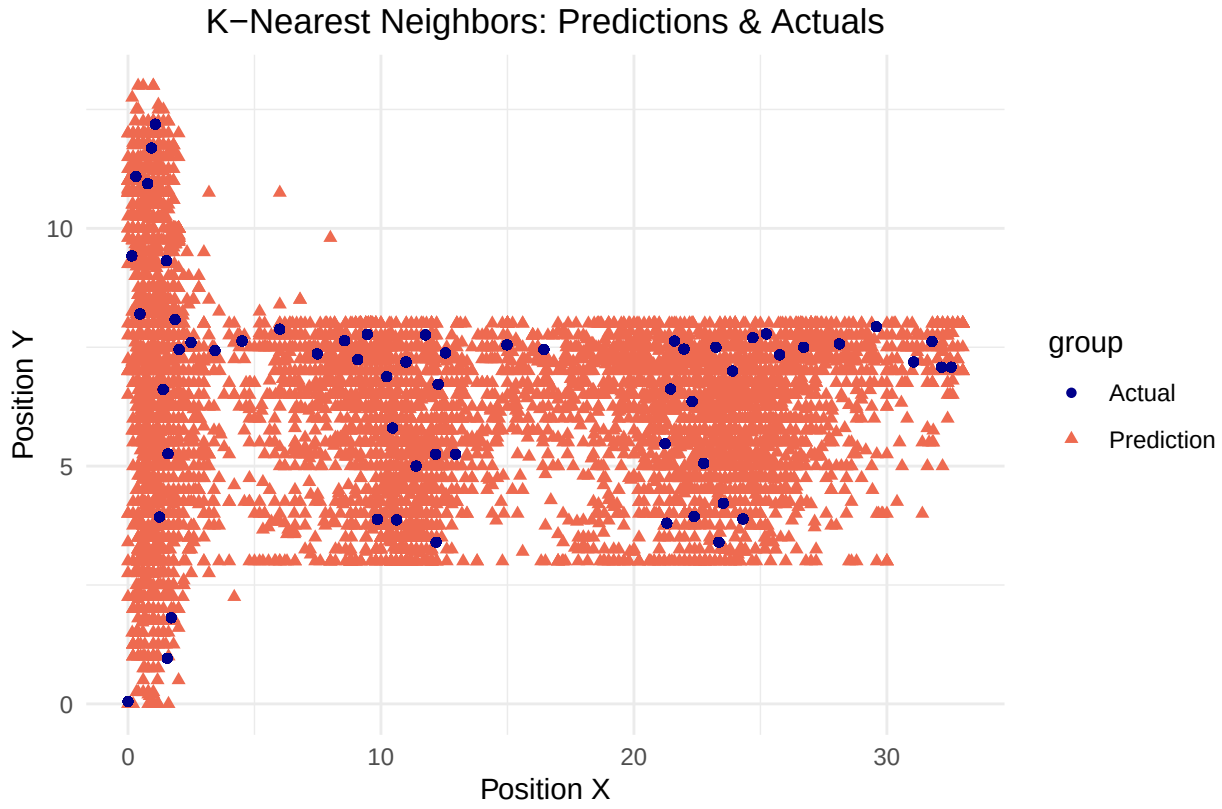

Figure 4: Positions of predicted X−Y values (blue) and actual values (red)

4 Conclusion

Modeling position with k-NN based on RSSI signal strength does not appear to yield useful results. Measurements were taking every 0.5 meters in the indoor environment, but the error from the model is much larger than 0.5 meters as shown in Fig. 3. Based on Fig. 4, we also can see that predictions of the online dataset do not appear to be very close to the actual positions. Overall, the k-NN does not appear to be very useful for predicting position based on RSSI values.

In addition, k-NN is a slow prediction algorithm because it computes distance across the entire data set every time it is executed. As noted earlier, the algorithm scales as shown in (1). This calculation time is not practical for a real-time location as the overhead time consumption for computation is too great. Given that electromagnetic signal strength decreases by $1/d^2$ per unit distance d as described by Hayt and Buck (2006), polynomial regression may be a better suited alternative to the K-Nearest Neighbors model used. In addition, polynomial regression has a much smaller time complexity than k-NN since it does not have to re-evaluate the entire data set for each new prediction.

The following assumptions would need to be met for a regression approach to be efficable:

- Variance of the residuals is constant
- Normally distributed residuals
- No outliers with high leverage

A Code

The code used to produce the analysis in shown below.

```
#' Process a single line of the data files.
#'
#' @description
#' Produces a list of matricies from a data line.
#' The columns are as follows:
#' time, scanMAC, positionX, positionY, positionZ,
#' orientation, MAC, signalRSSI, channel, router_type
#'
#' @param x A file line
#'
processLine = function(x)
{
  # tokenize line on delimiters ;=,
  tokens = strsplit(x, "[;=,]")[[1]]
  # return null when there are no measurements
  if (length(tokens) == 10)
    return(NULL)
  # get matrix of measured RSSI
  tmp = matrix(tokens[ - (1:10) ], , 4, byrow = TRUE)
  # add handheld device data and return resulting matrix
  cbind(matrix(tokens[c(2, 4, 6:8, 10)], nrow(tmp), 6,
               byrow = TRUE), tmp)
}

#' Round the measurement angle to the nearest 45 deg.
#'
#' @param angles a vector of angles; expected 0 - 360
#'
roundOrientation = function(angles)
  {
  # create a sequence of reference angles
  # 0, 45, 90, ... , 315
  refs = seq(0, by = 45, length  = 9)
  # round angles to the closest reference value
  q = sapply(angles, function(o) which.min(abs(o - refs)))
  c(refs[1:8], 0)[q]
}
```

```
#' Load data files for this case study.
#'
#' @description
#' Produces a data.frame with the following columns
#' "time", "posX", "posY", "orientation", "mac", "signal",
#' "rawTime", "angle"
#'
#' Only regular access points are kept (router_type==3).
#' Time is converted from milliseconds to seconds.
#'
#'
#' @param filename The path to the data file
#' @param subMacs A vector of MAC addresses to use in the data
#'
readData =
  function(filename = './offline.data.txt',
           subMacs = c("00:0f:a3:39:e1:c0", "00:0f:a3:39:dd:cd", "00:14:bf:b1:97:8a",
                       "00:14:bf:3b:c7:c6", "00:14:bf:b1:97:90", "00:14:bf:b1:97:8d",
                       "00:14:bf:b1:97:81"))
  {
    txt <- readLines(filename)
    lines <- txt[ substr(txt, 1, 1) != "#" ]
    tmp <- lapply(lines, processLine)
    offline <- as.data.frame(do.call("rbind", tmp),
                             stringsAsFactors= FALSE)

    names(offline) <- c("time", "scanMac",
                        "posX", "posY", "posZ", "orientation",
                        "mac", "signal", "channel", "type")

    # keep only signals from access points
    offline <- offline[ offline$type == "3", ]

    # drop scanMac, posZ, channel, and type - no info in them
    dropVars <- c("scanMac", "posZ", "channel", "type")
    offline <- offline[ , !( names(offline) %in% dropVars ) ]

    # drop more unwanted access points
    offline <- offline[ offline$mac %in% subMacs, ]

    # convert numeric values
    numVars <- c("time", "posX", "posY", "orientation", "signal")
    offline[ numVars ] <- lapply(offline[ numVars ], as.numeric)
```

```
    # convert time to POSIX
    offline$rawTime <- offline$time
    offline$time <- offline$time/1000
    class(offline$time) <- c("POSIXt", "POSIXct")

    # round orientations to nearest 45
    offline$angle = roundOrientation(offline$orientation)

    return(offline)
  }



# libraries
library(pacman)
p_load(tidyverse, caret, dplyr, magrittr, knitr, tidyr)
source('./functions.R')

# set a random seed for repodicibility
set.seed(42)

# load the data
offline <- readData()
online <- readData(filename = './online.data.txt',
                   subMacs = unique(offline$mac))

macAddresses <- sort(unique(offline$mac))

# create a unique position identifier
offline$posXY <- paste(offline$posX, offline$posY, sep = '-')
online$posXY <- paste(online$posX, online$posY, sep = '-')



# pivot the dataset to have mac addresses as features
# and RSSI values as their features
offline_pivot <- offline %>%
   tidyr::pivot_wider(names_from = mac,
               values_from = 'signal',
               values_fn = list(signal=mean))
# impute any NAs with -100
offline_pivot[is.na(offline_pivot)] <- -100
```

```
# pivot the dataset to have mac addresses as features
# and RSSI values as their features
online_pivot <- online %>%
   tidyr::pivot_wider(names_from = mac,
               values_from = 'signal',
               values_fn = list(signal=mean))
# impute any NAs with -100
online_pivot[is.na(online_pivot)] <- -100

# subset to the features of interest
offline_subset <- select(offline_pivot, c("posX", "posY", 'posXY', "angle",
                                    "00:14:bf:b1:97:8a", "00:14:bf:b1:97:90", "00:0f:a3:39:e1:c0",
                                    "00:14:bf:b1:97:8d", "00:14:bf:b1:97:81", "00:14:bf:3b:c7:c6",
                                    "00:0f:a3:39:dd:cd"))

# shuffle the dataset before training.
rows <- sample(nrow(offline_subset))
offline_shuffed_subset <- offline_subset[rows,]

# split out a 25% holdout set.
offline_train_75 = offline_shuffed_subset[1:floor((dim(offline_shuffed_subset)[[1]]*0.75)),] #686,213 t
offline_test_25 = offline_shuffed_subset[ceiling((dim(offline_shuffed_subset)[[1]]*0.75)):dim(offline_s



offline_train_subset <- offline_train_75 %>% filter(angle < 135-2 & angle < 135+2)
offline_test_subset <- offline_test_25 %>% filter(angle < 135-2 & angle < 135+2)

# create a tuning grid for k: 1 through 15
knn.grid <- expand.grid(k = seq(1:15))

# create a train controller for 5-fold cv
train.control <- trainControl(method = 'cv', number = 5)

# fit a model for posX with AP c0
model.c0.x <- train(
   posX ~ . ,
   data = offline_train_subset %>% select(-c("posY","posXY", "00:0f:a3:39:dd:cd")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)

# fit a model for posY with AP c0
```

```
model.c0.y <- train(
   posY ~ . ,
   data = offline_train_subset %>% select(-c("posX","posXY", "00:0f:a3:39:dd:cd")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)

# fit a model for posX with AP cd
model.cd.x <- train(
   posX ~ . ,
   data = offline_train_subset %>% select(-c("posY","posXY", "00:0f:a3:39:e1:c0")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)

# fit a model for posY with AP cd
model.cd.y <- train(
   posY ~ . ,
   data = offline_train_subset %>% select(-c("posX","posXY", "00:0f:a3:39:e1:c0")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)

# fit a model for posX with all APs
model.all.x <- train(
   posX ~ . ,
   data = offline_train_subset %>% select(-c("posY","posXY")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)

# fit a model for posY with all APs
model.all.y <- train(
   posY ~ . ,
   data = offline_train_subset %>% select(-c("posX","posXY")),
   method = 'knn',
   trControl = train.control,
   tuneGrid = knn.grid
)
```

```
# predict on holdout set
X_preds.allMAC <- predict(model.all.x, newdata = subset(offline_test_subset, select = -c(`posY`, `posXY
Y_preds.allMAC <- predict(model.all.y, newdata = subset(offline_test_subset, select = -c(`posX`, `posXY

# create data frame of predictions and actuals for comparison
X_res = data.frame(X_preds.allMAC, offline_test_subset$posX)
Y_res = data.frame(Y_preds.allMAC, offline_test_subset$posY)
names(X_res) <- c('preds','actuals')
names(Y_res) <- c('preds','actuals')
X_res$residuals = X_res$preds - X_res$actuals
Y_res$residuals = Y_res$preds - Y_res$actuals

# calculate X error
X_RSS <- c(crossprod(X_res$residuals))
X_MSE <- X_RSS / length(X_res$residuals)
X_RMSE <- sqrt(X_MSE)

# calculate Y error
Y_RSS <- c(crossprod(Y_res$residuals))
Y_MSE <- Y_RSS / length(Y_res$residuals)
Y_RMSE <- sqrt(Y_MSE)

# combine error from both models
comboRMSE <- sqrt(X_MSE + Y_MSE)

# predict on online set
preds_x<- predict(model.all.x, online_pivot[,c(6,8:14)])
preds_y<- predict(model.all.y, online_pivot[,c(6,8:14)])

# create data frame of predictions and actuals for comparison
X_res = data.frame(preds_x, online_pivot$posX)
Y_res = data.frame(preds_y, online_pivot$posY)
names(X_res) <- c('preds','actuals')
names(Y_res) <- c('preds','actuals')
X_res$residuals = X_res$preds - X_res$actuals
Y_res$residuals = Y_res$preds - Y_res$actuals

# calculate X error
X_RSS <- c(crossprod(X_res$residuals))
X_MSE <- X_RSS / length(X_res$residuals)
X_RMSE <- sqrt(X_MSE)
```

```
# calculate Y error
Y_RSS <- c(crossprod(Y_res$residuals))
Y_MSE <- Y_RSS / length(Y_res$residuals)
Y_RMSE <- sqrt(Y_MSE)

# combine error from both models
comboRMSE <- sqrt(X_MSE + Y_MSE)

# create a data frame of the X model errors
errorsX <- data.frame(
   k = rep(model.all.y$results$k, 3),
   model = c(
      rep('All APs', length(model.all.x$results$k)),
      rep('AP-c0', length(model.all.x$results$k)),
      rep('AP-cd', length(model.all.x$results$k))),
   errorX = c(model.all.x$results$RMSE,
               model.c0.x$results$RMSE,
               model.cd.x$results$RMSE)
)

# plot the X model errors
errorsX %>%
   ggplot(aes(x = k, y = errorX, color = model)) +
   geom_line() +
   xlab('k Neighbors') +
   ylab('RMSE') +
   ggtitle('RMSE of Position X Models with 5-Fold CV over k') +
   labs(caption = paste(
      'Figure 1:\n',
      'The mean RMSE of 5-fold cross validation for k-NN models using all APs (All APs),\n',
      'all APs excluding 00:0f:a3:39:dd:cd (AP-c0),',
      'and all APs excluding 00:0f:a3:39:e1:c0 (AP-cd)\nfor k values 1 through 15. This includes positi
      ) +
  theme_minimal() +
   theme(
      plot.title = element_text(hjust = 0.5),
      plot.caption = element_text(hjust = 0)
  )

# create a data frame of the X model errors
errorsY <- data.frame(
   k = rep(model.all.y$results$k, 3),
   model = c(
```

```
      rep('All APs', length(model.all.y$results$k)),
      rep('AP-c0', length(model.all.y$results$k)),
      rep('AP-cd', length(model.all.y$results$k))),
   errorY = c(model.all.y$results$RMSE,
               model.c0.y$results$RMSE,
               model.cd.y$results$RMSE)
)

# plot the Y model errors
errorsY %>%
   ggplot(aes(x = k, y = errorY, color = model)) +
   geom_line() +
   xlab('k Neighbors') +
   ylab('RMSE') +
   ggtitle('RMSE of Position Y Models with 5-Fold CV over k') +
   labs(caption = paste(
      'Figure 2:\n',
      'The mean RMSE of 5-fold cross validation for k-NN models using all APs (All APs),\n',
      'all APs excluding 00:0f:a3:39:dd:cd (AP-c0),',
      'and all APs excluding 00:0f:a3:39:e1:c0 (AP-cd)\nfor k values 1 through 15. This includes positi
      ) +
  theme_minimal() +
   theme(
      plot.title = element_text(hjust = 0.5),
      plot.caption = element_text(hjust = 0)
  )

# create a data frame of the error for both models
errorsALL <- data.frame(
   k = rep(model.all.y$results$k, 3),
   model = c(
      rep('All APs', length(model.all.y$results$k)),
      rep('AP-c0', length(model.all.y$results$k)),
      rep('AP-cd', length(model.all.y$results$k))),
   errorALL = c(model.all.y$results$RMSE + model.all.x$results$RMSE,
               model.c0.y$results$RMSE + model.c0.x$results$RMSE,
               model.cd.y$results$RMSE + model.cd.x$results$RMSE)
)

# plot the errors for both models
errorsALL %>%
   ggplot(aes(x = k, y = errorALL, color = model)) +
   geom_line() +
```

```
   xlab('k Neighbors') +
   ylab('RMSE') +
   ggtitle('RMSE of Models of both Positions (X,Y) with 5-Fold CV over k') +
   labs(caption = paste(
      'Figure 3:\n',
      'The mean RMSE of 5-fold cross validation for k-NN models using all APs (All APs),\n',
      'all APs excluding 00:0f:a3:39:dd:cd (AP-c0),',
      'and all APs excluding 00:0f:a3:39:e1:c0 (AP-cd)\nfor k values 1 through 15.', sep='')
      ) +
  theme_minimal() +
   theme(
      plot.title = element_text(hjust = 0.5),
      plot.caption = element_text(hjust = 0)
  )

# predict positions from model
preds_x<- predict(model.all.x, online_pivot[,c(6,8:14)])
preds_y<- predict(model.all.y, online_pivot[,c(6,8:14)])

# graph the position predictions vs the actual positions
data.frame(
  x=c(preds_x, online_pivot$posX),
  y=c(preds_y, online_pivot$posY),
  group=c(rep('Prediction', length(preds_y)), rep('Actual', length(online_pivot$posY)))
) %>%
  ggplot(aes(x=x, y=y, color = group)) + geom_point(aes(shape=group)) +
   ggtitle('K-Nearest Neighbors: Predictions & Actuals') +
  ylab("Position Y") +
  xlab("Position X") +
   labs(caption = paste(
      'Figure 4:',
      'Positions of predicted X-Y values (blue) and actual values (red)')
      ) +
  theme_minimal() +
  theme(
      plot.title = element_text(hjust = 0.5),
      plot.caption = element_text(hjust = 0.5)
  ) + scale_color_manual(values=c("blue4", "coral2")
)
```

References

Hayt, W. H. and Buck, J. A. (2006.). *Engineering electromagnetics*. Tata McGraw-Hill„ 7th ed. edition.

Nolan, D. and Lang, D. T. (2015). *Data Science in R A Case Studies Approach to Computational Reasoning and Problem Solving*. CRC Press.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.