How Much Did It Rain – Kaggle Competition

Tyler Byers

December 6, 2015

```
##-----Load Libraries-----
library(dplyr); library(randomForest); library(rpart);
library(caret); library(ggplot2); library(MESS); library(randomForest)
library(gbm)
```

Summary

```
getwd()
```

```
## [1] "/Users/tybyers/UW_DataScience/DataAtScale/final_project"
```

Competition Description

Data

Scoring

Modeling

Data Clean-Up

Flattening Observations

Changing Target Variable Distribution

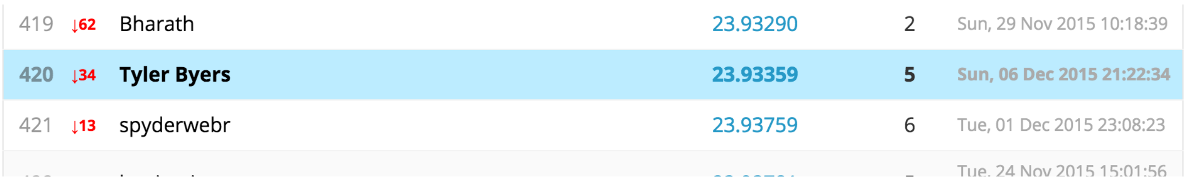
Cross Validation

CART Model

GBM Model

Competition Submission, Results

Public Leaderboard



419	↓62	Bharath	23.93290	2	Sun, 29 Nov 2015 10:18:39
420	**↓34**	**Tyler Byers**	**23.93359**	**5**	**Sun, 06 Dec 2015 21:22:34**
421	↓13	spyderwebr	23.93759	6	Tue, 01 Dec 2015 23:08:23
					Tue, 24 Nov 2015 15:01:56

Conclusions

Appendix – R Script

Below is the entirety of the script I built as I was completing this project. It includes comments that I put as I was going along.

```
##--------------------------------------------
##
## Kaggle Final Project Part #2 of 2
##
## Class: Methods for Data Science at Scale
## UW Data Science Certificate Class #3 of 3
##
## Tyler Byers
## Nov 30, 2015
##--------------------------------------------

## This documents my work on the Kaggle Project for the UW Methods for
##  Data Science at Scale course.  This is part #2 of my project.

## This is the only document I am turning in.  You may read my
##  code and look at my notes to see what I have tried.  I tried to document
## my local CV scores as well as my Kaggle scores when submitted.

## Suggest you do NOT run the following code, as it may take hours to run!!

##-----Set working directory-----
setwd('~/UW_DataScience/DataAtScale/kaggle_howmuchdiditrain/')

##-----Load Libraries-----
library(dplyr); library(randomForest); library(rpart);
library(caret); library(ggplot2); library(MESS); library(randomForest)
library(gbm)

##-----Load Data Sets --------------
train <- read.csv('./data/train.csv')
test <- read.csv('./data/test.csv')

## Filter out train$Expected
#train <- train %>% filter(Expected < 350)
## Realized later that doing this messes up the scoring, so need to leave all values in.
## Maybe when training we can ignore it.

## This is a mostly a feature engineering project.
##  First thing I'm going to do is just take the mean and
##  median of each radar reading and then model with that.

train_1perhr <- train %>% group_by(Id) %>%
  summarise(n_obs = n(), radardist_km = first(radardist_km),
            Ref_mean = mean(Ref, na.rm = TRUE), #Ref_med = median(Ref, na.rm = TRUE),
            Ref_5x5_10th_mean = mean(Ref_5x5_10th, na.rm = TRUE),
            Ref_5x5_50th_mean = mean(Ref_5x5_50th, na.rm = TRUE),
            Ref_5x5_90th_mean = mean(Ref_5x5_90th, na.rm = TRUE),
            RefComposite_mean = mean(RefComposite, na.rm = TRUE),
            RefComposite_5x5_10th_mean = mean(RefComposite_5x5_10th, na.rm = TRUE),
            RefComposite_5x5_50th_mean = mean(RefComposite_5x5_50th, na.rm = TRUE),
            RefComposite_5x5_90th_mean = mean(RefComposite_5x5_90th, na.rm = TRUE),
            RhoHV_mean = mean(RhoHV, na.rm = TRUE),
            RhoHV_5x5_10th_mean = mean(RhoHV_5x5_10th, na.rm = TRUE),
            RhoHV_5x5_50th_mean = mean(RhoHV_5x5_50th, na.rm = TRUE),
            RhoHV_5x5_90th_mean = mean(RhoHV_5x5_90th, na.rm = TRUE),
            Zdr_mean = mean(Zdr, na.rm = TRUE),
            Zdr_5x5_10th_mean = mean(Zdr_5x5_10th, na.rm = TRUE),
            Zdr_5x5_50th_mean = mean(Zdr_5x5_50th, na.rm = TRUE),
            Zdr_5x5_90th_mean = mean(Zdr_5x5_90th, na.rm = TRUE),
            Kdp_mean = mean(Kdp, na.rm = TRUE),
            Kdp_5x5_10th_mean = mean(Kdp_5x5_10th, na.rm = TRUE),
            Kdp_5x5_50th_mean = mean(Kdp_5x5_50th, na.rm = TRUE),
            Kdp_5x5_90th_mean = mean(Kdp_5x5_90th, na.rm = TRUE),
            Expected = first(Expected))

## Function to check error
mae <- function(actual, pred) {
  mae <-  mean(abs(actual - pred))
  print(paste('MAE: ', mae))
  mae
}

write_output <- function(id, pred) {
  output <- data.frame(Id = id, Expected = pred)
  curdatetime <- strftime(Sys.time(), '%Y%m%d_%H%M%S')
  write.csv(output, paste0('output/', curdatetime, '.csv'), row.names = FALSE)
  print(paste0('Wrote file: ',  paste0('/output/', curdatetime, '.csv')))
}

## Going to do a baseline test using the mean rainfall for all observations
mean_all <- mean((train_1perhr %>% filter(Expected < 350))$Expected)  # mean of Expected with obs > 350
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)], mean_all)
# MAE: 11.77 -- hmm, doesn't seem right (later -- was with filtered Expected vals)
#  With ugly bad Expeced vals, score is 27.2388, much more in line w/ Kaggle score.
write_output(unique(test$Id), mean_all)
# kaggle score: 27.82422

## What if I take the mean of Expected, filtering out over 150, what then is my score?
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)],
    mean((train_1perhr %>% filter(Expected < 150))$Expected))
## Score: 24.5667

median(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)])
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)], 1.27)
## Score: 23.53 --- would be top of leaderboard, seems too good to be true though
write_output(unique(test$Id), 1.27)
## Kaggle score: 24.1578, so it was too good to be true :)

## Now need to figure out how to format the data so that we know how many minutes
##  each reading is good for.
```

```
# Try the Murray-Palmer transformation, using the mean ref (nothing fancy)
#  Still not doing real machine learning, just calculation, so don't need
#  to do CV or anything like that.
mp_pred <- (10^(train_1perhr$Ref_mean/10)/200)^(5/8)
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)],
    mp_pred[!is.na (mp_pred)])
## Local score: 23.361129

# Now do this calculation for the test set
test_1perhr <- test %>% group_by(Id) %>%
  summarise(n_obs = n(), radardist_km = first(radardist_km),
            Ref_mean = mean(Ref, na.rm = TRUE))
mp_pred_test <- (10^(test_1perhr$Ref_mean/10)/200)^(5/8)
## Fill in the NAs with 9999
mp_pred_test[is.na (mp_pred_test)] <- 9999
write_output(unique(test$Id), mp_pred_test)
## Kaggle score: 24.01167, moved up to 377th place


## Want to do a spline-fit under the curve.
## Can do a spline fit for each and every id
# system.time(Ref_spline <- sapply(unique(train$Id)[1:250], function(id) {
#   if(id %% 1000 == 1) {
#     print(paste(Sys.time(), ': Id = ', id))
#   }
#   #id_df <- train %>% filter(Id == id)
#   id_rows <- which(train$Id == id)
#   if(sum(!(is.na (train$Ref[id_rows]))) > 0) {
#     # do the calculation. Sort of a simple integration
#     sum(spline(x = train$minutes_past[id_rows],
#                y = train$Ref[id_rows], xout = 0:60)$y)/60
#   } else {
#     NA
#   }
# })
# )
# The above is going to take around 26 hours to do. Need a different way.


# Maybe take advantage of the group_by in dplyr
# do auc from MESS package....but cannot have NAs on the ends
valid_ids <- train_1perhr$Id[which(!is.na (train_1perhr$Ref_mean))]
system.time(Ref_spline <- train %>% filter(Id %in% valid_ids) %>%
              group_by(Id) %>%
              summarise(Ref_spline_periodic = sum(spline(x = minutes_past,
                                                         y = Ref, xout = 0:60,
                                                         method = 'periodic')$y)/60))

# This took 177 seconds! Sweet!  (with method = 'periodic')
# However, it led to some horribly unbounded spline fits, leading to Inf
#  results on the Murray Palmar fit.
sfit_pred <-  (10^(Ref_spline$Ref_spline_periodic/10)/200)^(5/8)
sfit_pred[sfit_pred > 200] <- 150
```

```r
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)],
    sfit_pred)
# Local MAE: 23.4652

# Try with a natural spline fit:
Ref_spline_natural <- train %>% filter(Id %in% valid_ids) %>%
  group_by(Id) %>%
  summarise(Ref_spline_natural = sum(spline(x = minutes_past,
                                           y = Ref, xout = 0:60,
                                           method = 'natural')$y)/60)
sfit_pred_natural <-  (10^(Ref_spline_natural$Ref_spline_natural/10)/200)^(5/8)
sfit_pred_natural[sfit_pred_natural > 150] <- 150
mae(train_1perhr$Expected[!is.na (train_1perhr$Ref_mean)],
    sfit_pred_natural)
## Local score: 24.727.  Maybe not worth submitting.  That method isn't so great.

## Realized after looking at 4th posting on this page:
##  https://www.kaggle.com/c/how-much-did-it-rain-ii/forums/t/16572/38-missing-data/
## that I need to first convert Ref column to rain rate.  Maybe then I can find
## area under the curve and go from there.

# Remove those dfs to reclaim space
rm(Ref_spline); rm(Ref_spline_natural); rm(mp_pred); rm(mp_pred_test);
rm(sfit_pred_natural); rm(sfit_pred); rm(valid_ids)

# Calculate instantaneous rain rate (mm/hr):
train$Ref_irr <- (10^(train$Ref/10)/200)^(5/8)
## all the NAs should be 0 now, for 0 instant rain rate.
train$Ref_irr[is.na (train$Ref_irr)] <- 0
summary(train$Ref_irr)
# Now do the auc calc, with a 0 anchor just before 0 and after 60
system.time(train_Ref_irr_auc <- train %>% group_by(Id) %>%
              summarise(n_obs = n(),
                        Est_rain_linear = auc(x = c(-0.001, minutes_past, 60.001),
                                              y = c(0, Ref_irr, 0), type = 'linear')/60#,
                        #Est_rain_spline = auc(x = c(-0.001, minutes_past, 60.001),
                        #               y = c(0, Ref_irr, 0), type = 'spline')/60
              )
)
train_Ref_irr_auc$Est_rain_spline[train_Ref_irr_auc$Est_rain_spline < 0] <- 0
## only problem is that for the 1-observation hours we're getting NA,
##  so added anchors on either side of 0 and 60.
scorerows <- !is.na (train_1perhr$Ref_mean)
mae(train_1perhr$Expected[scorerows],
    train_Ref_irr_auc$Est_rain_linear[scorerows])
# Calc MAE: 23.3873.  Slightly lower than my "mean" method, but same area.
mae(train_1perhr$Expected[scorerows],
    train_Ref_irr_auc$Est_rain_spline[scorerows])
# Spline calc MAE: 36.883.  This gets pretty unbounded, so going to use 'linear'
#  from here on out.

# That MAE of 23.3873 is close to that for my "mean" calculation. Not going to
#  submit to Kaggle at this time.
# But, want to do some modeling with all the Ref values and the km from gauge.
#  Maybe can start to get better results with that.

# Want a function with the MP transformation
mp_transform <- function(Ref_vect) {
  (10^(Ref_vect/10)/200)^(5/8)
}

train_transf <- as.data.frame(lapply(train[,5:11], function(x) { mp_transform(x) }))
names(train_transf) <- paste0(names(train_transf), '_irr')
summary(train_transf)
train_transf <- as.data.frame(lapply(train_transf, function(x) {
  x[is.na (x)] <- 0
  x
}))
train_transf <- bind_cols(train[c('Id', 'minutes_past', 'radardist_km', 'Ref_irr')],
                          train_transf, train[c('Expected')])
# Now do AUC for each column.
train_allref_auc <- train_transf %>% group_by(Id) %>%
  summarise(n_obs = n(), radardist_km = first(radardist_km),
            Ref_auc = auc(x = c(-0.001, minutes_past, 60.001),
                          y = c(0, Ref_irr, 0), type = 'linear')/60,
            Ref_5x5_10th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_10th_irr, 0), type = 'linear')/60,
            Ref_5x5_50th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_50th_irr, 0), type = 'linear')/60,
            Ref_5x5_90th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_90th_irr, 0), type = 'linear')/60,
            RefComposite_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, RefComposite_irr, 0), type = 'linear')/60,
            RefComposite_5x5_10th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_10th_irr, 0),
                                            type = 'linear')/60,
            RefComposite_5x5_50th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_50th_irr, 0),
                                            type = 'linear')/60,
            RefComposite_5x5_90th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_90th_irr, 0),
                                            type = 'linear')/60
  )


# Need to know which Ids are all-NA Ref (don't train or score on these)
# Need to know which Ids have Expected > 250 (too high of readings)
allrefna_train <- train_1perhr$Id[is.na (train_1perhr$Ref_mean)]
highExpected_train <- train_1perhr$Id[train_1perhr$Expected > 250]

# Modeling (remember to filter out Expected above 200 when training), and filter
#  out the Ids with all NA Ref values when training.


run_mods <- function(data, allrefna, highExpected, fol, mod = 'rpart', k = 10) {
  print(paste(Sys.time(), ': Entering run_mods function'))
  folds <- createFolds(data$Expected, k = k)
  i <- 0
  accuracies <- lapply(folds, function(fold) {
    i <<- i + 1
    train <- data[-fold, ]
    print(paste('Fold: ', i))
    #print(paste('nrow(train):', nrow(train)))
    train_filt <- train %>% filter(!(Id %in% allrefna)) %>%
      filter(!(Id %in% highExpected))
    #print(paste('nrow(train_filt):', nrow(train_filt)))
    test <- data[fold, ]
    if(mod == 'rpart') {
      cart <- rpart(fol, data = train_filt)
      pred <- predict(cart, newdata = test)

    } else if(mod == 'rf') {
      rf <- randomForest(fol, data = train_filt)
      print(importance(rf))
      pred <- predict(rf, newdata = test)
    } else if(mod == 'gbm') {
      gbm_mod <- gbm(formula = fol, data = train_filt, n.trees = 250,
                     interaction.depth = 3, verbose = TRUE, distribution = 'laplace')
      print(summary(gbm_mod))
      pred <- predict(gbm_mod, newdata = test, n.trees = 250)
    }
    pred_df <- data.frame(Id = test$Id, pred = pred)
    #print(paste('nrow(pred_df)', nrow(pred_df)))

    pred_df$Expected <- test$Expected
    print("Prediction summary:")
    print(summary(pred_df))
    pred_df <- pred_df %>% filter(!(Id %in% allrefna))
    print(summary(pred_df))
    #print(paste('nrow(pred_df) after filter', nrow(pred_df)))
    mae_score <- mae(pred_df$Expected, pred_df$pred)
    print(paste(Sys.time(), ': MAE #',i,':', mae_score))
    mae_score
    #list(gbm_mod, train, train_filt, test, pred_df)
  })
  print(paste('Average MAE with 10-fold cv:',
              sum(unlist(accuracies))/k))
  accuracies
}


train_allref_auc$Expected <- train_1perhr$Expected

fol <- Expected ~ radardist_km + Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
  RefComposite_5x5_90th_auc

rparts <-
```

```
  run_mods(train_allref_auc, allrefna_train,
           highExpected_train, fol, mod = 'rpart', k = 10)
#[1] "Average MAE with 10-fold cv: 24.6999164195204"

# Try a randomForest
library(randomForest)
#run_mods(train_allref_auc, allrefna_train, highExpected_train, fol, mod = 'rf')
# random forest was taking HOURS to run just a few trees.  So trying something else.

fol_gbm <- fol <- Expected ~ radardist_km + Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc
gbm_maes <- run_mods(train_allref_auc, allrefna_train, highExpected_train, fol_gbm,
                     mod = 'gbm', k = 10)
#  "Average MAE with 5-fold cv: 24.8861770452619" with
#   fol <- Expected ~ radardist_km + Ref_auc + Ref_5x5_10th_auc +
#  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
#  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
#  RefComposite_5x5_90th_auc
#  Need to inspect closer to see if I'm doing it right.

# Figured it out -- was using "gaussian" distribution.
#  Changed to Laplace distribution and...
#    Average MAE with 5-fold cv: 23.4393403784286" (fol_gbm)
#  "Average MAE with 10-fold cv: 23.440215205844" (fol)
#  "Average MAE with 10-fold cv: 23.4410088231101"

## Decent-looking scores there for GBM, but I'm not ready to submit to Kaggle yet.

# Noticed that it's only predicted expected between 0.9 and 1.6.  Maybe if I take
#  the log of the Expected.  Then I can reverse it later....

# Create a new function with this change.
run_mods_Exp_log <- function(data, allrefna, highExpected, fol, mod = 'rpart', k = 10) {
  print(paste(Sys.time(), ': Entering run_mods function'))
  folds <- createFolds(data$Expected, k = k)
  i <- 0
  # Take log of Expected
  data$Expected <- log(data$Expected + 1)
  accuracies <- lapply(folds, function(fold) {
    i <<- i + 1
    train <- data[-fold, ]
    print(paste('Fold: ', i))
    #print(paste('nrow(train):', nrow(train)))
    train_filt <- train %>% filter(!(Id %in% allrefna)) %>%
      filter(!(Id %in% highExpected))
    #print(paste('nrow(train_filt):', nrow(train_filt)))
    test <- data[fold, ]
    if(mod == 'rpart') {
      cart <- rpart(fol, data = train_filt)
      pred <- predict(cart, newdata = test)
    } else if(mod == 'rf') {
      rf <- randomForest(fol, data = train_filt)
      print(importance(rf))
      pred <- predict(rf, newdata = test)
    } else if(mod == 'gbm') {
      gbm_mod <- gbm(formula = fol, data = train_filt, n.trees = 250,
                     interaction.depth = 3, verbose = TRUE, distribution = 'gaussian')
      print(summary(gbm_mod))
      pred <- predict(gbm_mod, newdata = test, n.trees = 250)
    }
    pred <- exp(pred) - 1 # un-do the log of Expected
    pred_df <- data.frame(Id = test$Id, pred = pred)
    #print(paste('nrow(pred_df)', nrow(pred_df)))

    pred_df$Expected <- exp(test$Expected) - 1  # undo the log of expected
    print("Prediction summary:")
    print(summary(pred_df))
    pred_df <- pred_df %>% filter(!(Id %in% allrefna))
    print(summary(pred_df))
    #print(paste('nrow(pred_df) after filter', nrow(pred_df)))
    mae_score <- mae(pred_df$Expected, pred_df$pred)
    print(paste(Sys.time(), ': MAE #',i,':', mae_score))
    mae_score
    #list(gbm_mod, train, train_filt, test, pred_df)
  })
  print(paste('Average MAE with',k,'-fold cv:',
              sum(unlist(accuracies))/k))
  accuracies
}

rparts_exp <-
  run_mods_Exp_log(train_allref_auc, allrefna_train,
                   highExpected_train, fol, mod = 'rpart', k = 10)
#[1] "Average MAE with 10 -fold cv: 23.3009274403245"
# This is better than my previous best score! So will have to
#  use this model and submit to Kaggle.

gbm_maes <- run_mods_Exp_log(train_allref_auc, allrefna_train, highExpected_train, fol_gbm,
                             mod = 'gbm', k = 5)
## With fol_gbm formula and laplace distribution:
# "Average MAE with 5 -fold cv: 23.4607803413511"
#  "Average MAE with 5 -fold cv: 23.5136031945773"



## As of turning in part 2 on 11/30/15, I'm still doing some modeling.  I haven't chosen a
##  final model yet, and I'd still like to get in some of the other variables. But I'm
## seeig some promising results with the rpart model and taking the log of the
## Expected values.  Will report on my modeling status for the final part of the project.

# An aside -- what is the best possible score I can expect if all readings under
#  250 are spot on?
pred_best <- train$Expected
median_under250 <- median((train %>% filter(Expected < 250))$Expected)
pred_best <- ifelse(pred_best < 250, pred_best, median_under250)
```

```r
mae(pred = pred_best[!(allrefna_train)], actual = train$Expected[!(allrefna_train)])
# MAE 17.62678 -- best possible!

# Ok, back on track.  I like what I saw from the rpart.
#  Need a test set formatted just like the training set.


test_transf <- as.data.frame(lapply(test[,4:11], function(x) { mp_transform(x) }))
names(test_transf) <- paste0(names(test_transf), '_irr')
summary(test_transf)
test_transf <- as.data.frame(lapply(test_transf, function(x) {
  x[is.na (x)] <- 0
  x
}))
test_transf <- bind_cols(test[c('Id', 'minutes_past', 'radardist_km')],
                         test_transf)
# Now do AUC for each column.
test_allref_auc <- test_transf %>% group_by(Id) %>%
  summarise(n_obs = n(), radardist_km = first(radardist_km),
            Ref_auc = auc(x = c(-0.001, minutes_past, 60.001),
                          y = c(0, Ref_irr, 0), type = 'linear')/60,
            Ref_5x5_10th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_10th_irr, 0), type = 'linear')/60,
            Ref_5x5_50th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_50th_irr, 0), type = 'linear')/60,
            Ref_5x5_90th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, Ref_5x5_90th_irr, 0), type = 'linear')/60,
            RefComposite_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                   y = c(0, RefComposite_irr, 0), type = 'linear')/60,
            RefComposite_5x5_10th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_10th_irr, 0),
                                            type = 'linear')/60,
            RefComposite_5x5_50th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_50th_irr, 0),
                                            type = 'linear')/60,
            RefComposite_5x5_90th_auc = auc(x = c(-0.001, minutes_past, 60.001),
                                            y = c(0, RefComposite_5x5_90th_irr, 0),
                                            type = 'linear')/60
  )

allrefna_test <- test_1perhr$Id[is.na (train_1perhr$Ref_mean)]

train_allref_auc$Expected_log <- log(train_allref_auc$Expected + 1)
train_allref_auc_filtered <- train_allref_auc %>% filter(!(Id %in% allrefna_train)) %>%
  filter(!(Id %in% highExpected_train))
fol_expectedlog = Expected_log ~ radardist_km + Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_5x5_10th_auc +
  RefComposite_5x5_50th_auc

cart <- rpart(fol_expectedlog, data = train_allref_auc_filtered)
summary(cart)
pred_cart <- predict(cart, newdata = test_allref_auc)
pred_cart <- exp(pred_cart) - 1
```

```r
summary(pred_cart)
# Kaggle score: 23.93418 -- moved up a few spots.

# Had a thought -- maybe if we include a km^2 factor it will work even
#  better -- distance-squared is an important thing to know in "intensity" calculations.

fol <- Expected ~ radardist_km + Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
  RefComposite_5x5_90th_auc
rparts_exp <-
  run_mods_Exp_log(train_allref_auc, allrefna_train,
                   highExpected_train, fol, mod = 'rpart', k = 10)
# with additional terms in fol:
#  [1] "Average MAE with 10 -fold cv: 23.3026793313298"

train_allref_auc$radardist_km_sq <- (train_allref_auc$radardist_km)^2
fol_kmsq <- fol <- Expected ~ radardist_km + radardist_km_sq +
  Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
  RefComposite_5x5_90th_auc
rparts_exp_kmsq <-
  run_mods_Exp_log(train_allref_auc, allrefna_train,
                   highExpected_train, fol_kmsq, mod = 'rpart', k = 10)
# [1] "Average MAE with 10 -fold cv: 23.3034193790669"
#  ^ did not help -- score basically same.


#Now going to do the auc calculations for RhoHV, Zdr and Kdp.


train_auc2 <- train %>% group_by(Id) %>%
  summarise(RhoHV_auc = auc(x = c(-0.001, minutes_past, 60.001),
                            y = c(0, RhoHV, 0), type = 'linear')/60.002,
            Zdr_auc = auc(x = c(-0.001, minutes_past, 60.001),
                          y = c(0, Zdr, 0), type = 'linear')/60,
            Kdp_auc = auc(x = c(-0.001, minutes_past, 60.001),
                          y = c(0, Kdp, 0), type = 'linear')/60)

# add above to the other train auc
train_allref_auc <- train_allref_auc %>% bind_cols(train_auc2)
# forgot to disinclude Id column from train_auc
train_allref_auc <- train_allref_auc[,c(1:14,16:18)]

# Ok now do another CART model:
train_allref_auc$radardist_km_sq_inv <- 1/(train_allref_auc$radardist_km_sq)
fol_thruKdp <- Expected ~ radardist_km + radardist_km_sq_inv +
  Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
  RefComposite_5x5_90th_auc + RhoHV_auc + Zdr_auc + Kdp_auc
rparts_thruKdp <-
```

```
  run_mods_Exp_log(train_allref_auc, allrefna_train,
                   highExpected_train, fol_thruKdp, mod = 'rpart', k = 10)
# [1] "Average MAE with 10 -fold cv: 23.3018106358255" radardist_km_sq
# basically same score
# [1] "Average MAE with 10 -fold cv: 23.2984152198045"  maybe slight better
#  but not enough to resubmit to kaggle I think


## GBM with a training grid --
gbm_n.trees <- c(100, 250, 500)
gbm_interaction.depth <- c(1, 2, 3, 4)
gbm_shrinkage <- c(0.001, 0.01, 0.05, 0.1)
gbm_grid <- expand.grid(gbm_n.trees, gbm_interaction.depth, gbm_shrinkage)
gbm_grid
# There are 48 to do.  this will take a long time to train.
names(gbm_grid) <- c('n.trees', 'interaction.depth', 'shrinkage')
gbm_grid$MAE <- NA

# Now need a 10-fold CV function
run_mods_gbm_grid <- function(data, allrefna, highExpected, fol, k = 10,
                              n.trees, interaction.depth, shrinkage) {
  print(paste(Sys.time(), ': Entering run_mods function'))
  folds <- createFolds(data$Expected, k = k)
  i <- 0
  # Take log of Expected
  data$Expected <- log(data$Expected + 1)
  accuracies <- lapply(folds, function(fold) {
    i <<- i + 1
    train <- data[-fold, ]
    print(paste('Fold: ', i))
    #print(paste('nrow(train):', nrow(train)))
    train_filt <- train %>% filter(!(Id %in% allrefna)) %>%
      filter(!(Id %in% highExpected))
    #print(paste('nrow(train_filt):', nrow(train_filt)))
    test <- data[fold, ]
    gbm_mod <- gbm(formula = fol, data = train_filt, n.trees = n.trees,
                   interaction.depth = interaction.depth,
                   shrinkage = shrinkage, verbose = TRUE, distribution = 'gaussian')
    print(summary(gbm_mod))
    pred <- predict(gbm_mod, newdata = test, n.trees = n.trees)
    pred <- exp(pred) - 1 # un-do the log of Expected
    pred_df <- data.frame(Id = test$Id, pred = pred)
    #print(paste('nrow(pred_df)', nrow(pred_df)))

    pred_df$Expected <- exp(test$Expected) - 1  # undo the log of expected
    print("Prediction summary:")
    print(summary(pred_df))
    pred_df <- pred_df %>% filter(!(Id %in% allrefna))
    print(summary(pred_df))
    #print(paste('nrow(pred_df) after filter', nrow(pred_df)))
    mae_score <- mae(pred_df$Expected, pred_df$pred)
    print(paste(Sys.time(), ': MAE #',i,':', mae_score))
    mae_score
```

```r
    #list(gbm_mod, train, train_filt, test, pred_df)
  })
  print(paste('Average MAE with',k,'-fold cv:',
              sum(unlist(accuracies))/k))
  sum(unlist(accuracies))/k
}

for(i in 1:nrow(gbm_grid)) {
  gbm_grid$MAE[i] <- run_mods_gbm_grid(train_allref_auc, allrefna_train,
                                       highExpected_train, fol_thruKdp,
                                       interaction.depth = gbm_grid$interaction.depth[i],
                                       shrinkage = gbm_grid$shrinkage[i],
                                       n.trees = gbm_grid$n.trees[i], k = 10)
  print(gbm_grid)
}

# Make the test set look like the train set:

test_allref_auc$radardist_km_sq_inv <- 1/(test_allref_auc$radardist_km)^2
test_auc2 <- test %>% group_by(Id) %>%
  summarise(RhoHV_auc = auc(x = c(-0.001, minutes_past, 60.001),
                             y = c(0, RhoHV, 0), type = 'linear')/60.002,
            Zdr_auc = auc(x = c(-0.001, minutes_past, 60.001),
                           y = c(0, Zdr, 0), type = 'linear')/60,
            Kdp_auc = auc(x = c(-0.001, minutes_past, 60.001),
                           y = c(0, Kdp, 0), type = 'linear')/60)

# add above to the other test auc
test_allref_auc <- test_allref_auc %>% bind_cols(test_auc2)
test_allref_auc <- test_allref_auc[,c(1:12,14:16)]

fol_thruKdp_gbm <- fol <- Expected_log ~ radardist_km + radardist_km_sq_inv +
  Ref_auc + Ref_5x5_10th_auc +
  Ref_5x5_50th_auc + Ref_5x5_90th_auc + RefComposite_auc +
  RefComposite_5x5_10th_auc + RefComposite_5x5_50th_auc +
  RefComposite_5x5_90th_auc + RhoHV_auc + Zdr_auc + Kdp_auc
gbm_mod_500_4_0.01 <- gbm(formula = fol_thruKdp_gbm, data = train_allref_auc,
                              n.trees = 500,
                              interaction.depth = 4,
                              shrinkage = 0.01, verbose = TRUE,
                              distribution = 'gaussian')

pred_gbm_500_4_0.01 <- predict(gbm_mod_500_4_0.01, newdata = test_allref_auc,
                               n.trees = 500)
pred_gbm_500_4_0.01 <- exp(pred_gbm_500_4_0.01) - 1
summary(pred_gbm_500_4_0.01)
write_output(unique(test$Id), pred_gbm_500_4_0.01)
# Kaggle score: 23.93359  -- no real improvement in leaderboard
```