FaceBook

Jillian Maw

4/13/2022

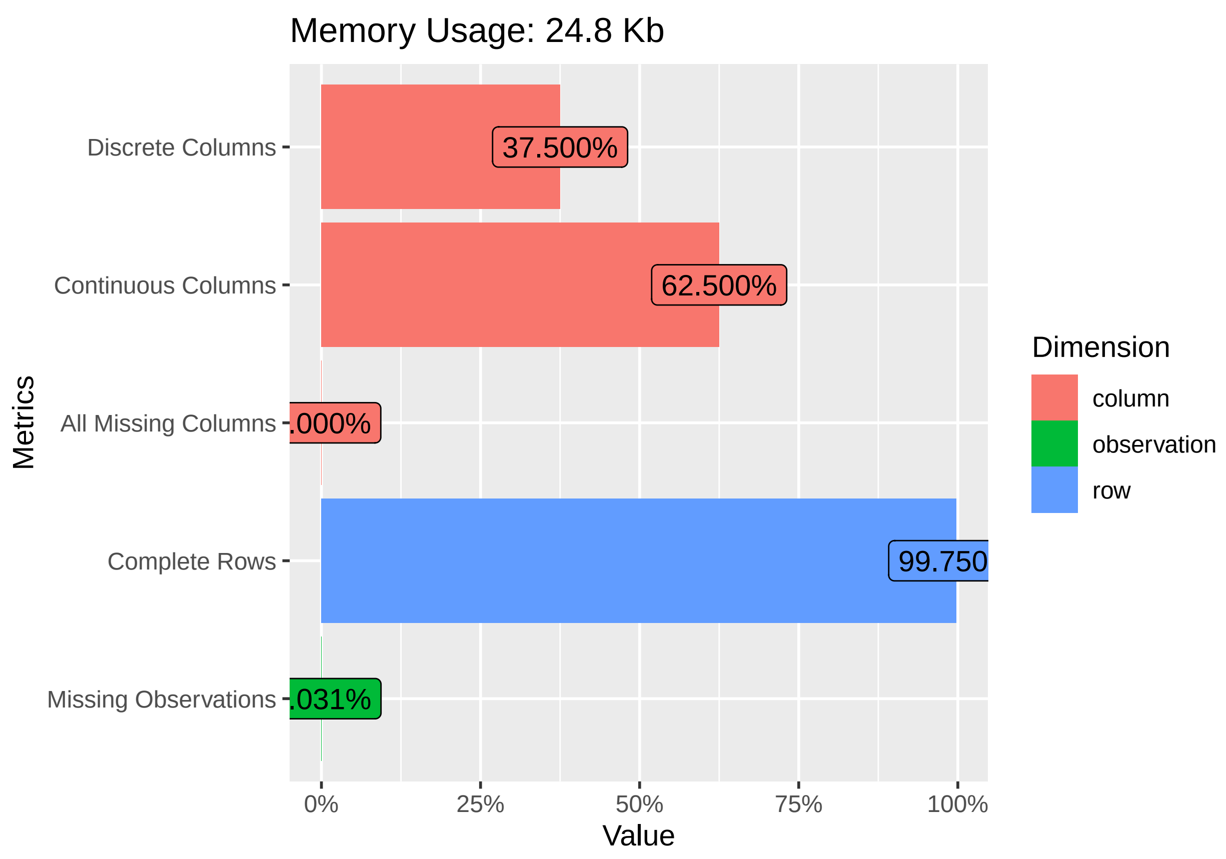
Companies often use social media to advertise their products. Measuring the impact of social media advertisements is an important piece of increasing the effectiveness of marketing. The facebook data considered here contain the following information:

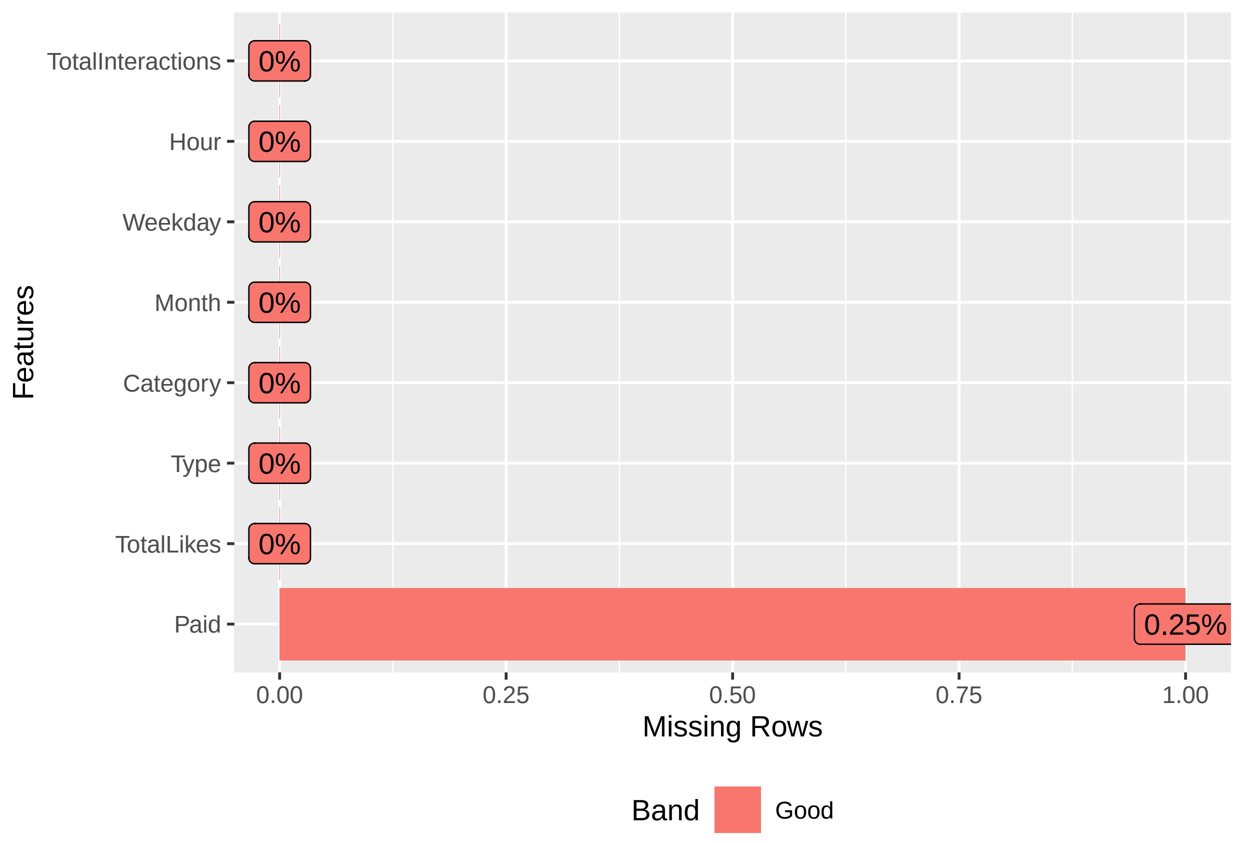
Variable Name	Description
TotalLikes	Total number of people who have liked the companies page
Category	Purpose of the advertisement
Type	Media type posted
Month	Month of posting
Weekday	Weekday of posting
Hour	Hour of posting
Paid	Did the company pay facebook to run the advertisement?
TotalInteractions	Total number of likes, comments and shares of the post

The goal of this analysis is to be able to predict the impact of the post (as measured by TotalInteractions) based on the other variables. The dataset `FacebookTraining.csv` contains all the training data used for predictive model fitting. The `FacebookAllTesting.csv` and `FacebookTesting.csv` should only be used in #4.

1. Explore the `FacebookTraining.csv` dataset and do the following:
   (a) Throw out any rows with missing values (generally you don't want to throw out missing values but since we didn't talk about ways to deal with missing values we'll throw them out for now).
   (b) Designate if you are treating the explanatory variable as categorical (nominal) or continuous.
   (c) Display a correlation plot between any continuous variables and TotalInteractions.

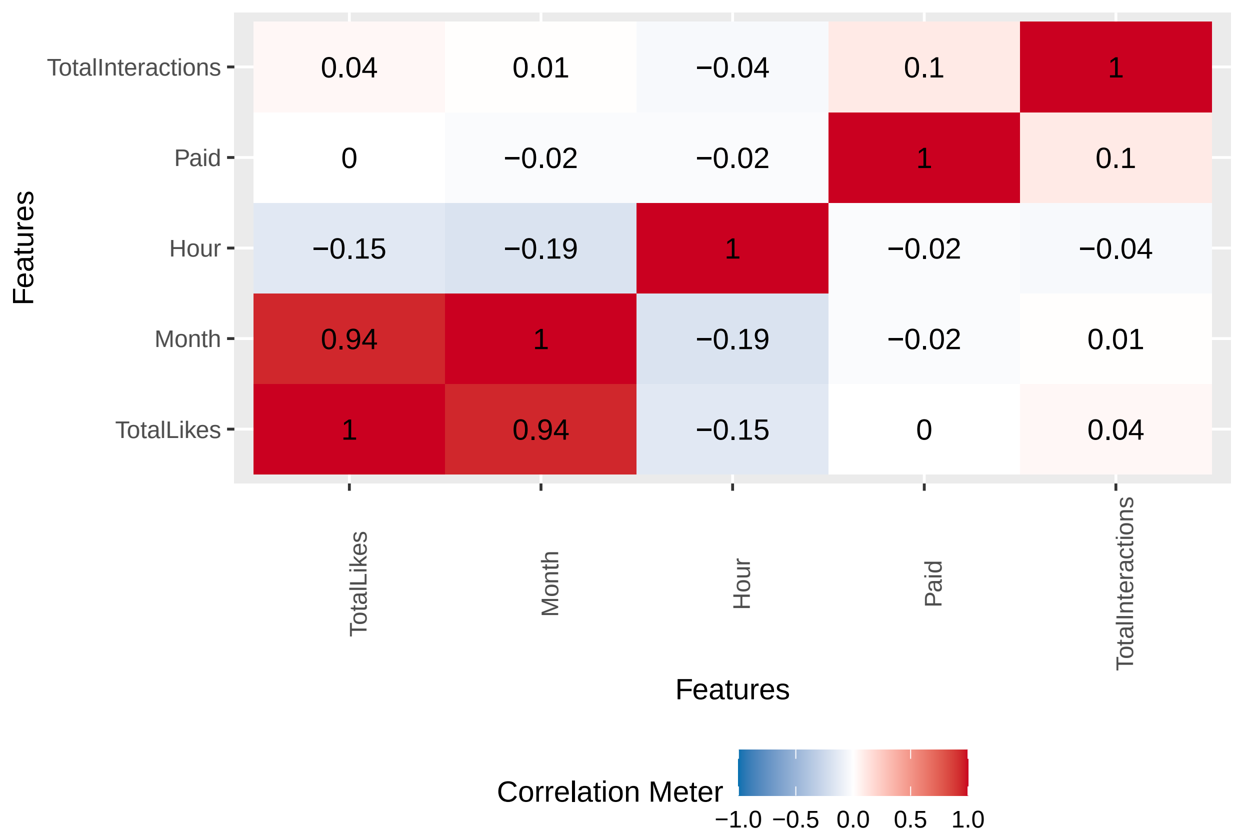
As can be seen in the graphs below, there was one missing value in the "Paid" column, so I removed that row from the data set, as requested.


Memory Usage: 24.8 Kb
Discrete Columns
Continuous Columns
All Missing Columns
Complete Rows
Missing Observations
Metrics
37.500%
62.500%
.000%
99.750
.031%
0%
25%
50%
75%
100%
Value
Dimension
column
observation
row



I will treat the explanatory variables "TotalLikes", "Month", "Hour", and "Paid" as continuous variables; "TotalLikes" since it is the numeric measure of the total number of people who have liked the companies' page, "Month" and "Hour" because they are literal measurements of time, and "Paid" because it is a binomial representation of whether a company paid FaceBook to run their advertising campaign or not. The other explanatory variables, "Category", "Type", and "Weekday", I will treat as categorical variables, because they have a limited number of outcomes, usually factors, that I can divide the variable's observations into. "Category" has 3 factors, "Inspiration", "Product", and "SpecialOffer", to describe the purpose of the advertisement. "Type" was used to describe what the media type posted was, and had 4 factors: "Link", "Photo", "Status", and "Video". "Weekday" had 7 factors, one for each day of the week.

Below is a correlation plot between the continuous variables "TotalLikes", "Month", "Hour", and "Paid" and the response variable "TotalInteractions". As can be seen, there is a strong correlation between the "Month" and "TotalLikes" variables, but everything else is fairly weak.



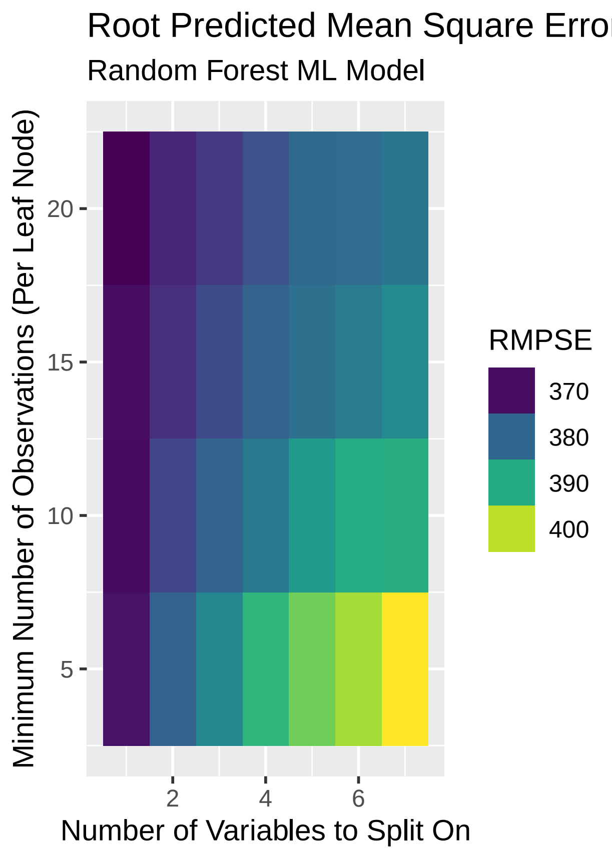
2. Pick at least 2 ML models and give a brief "birds-eye" view of how they are used to predict TotalInteractions.

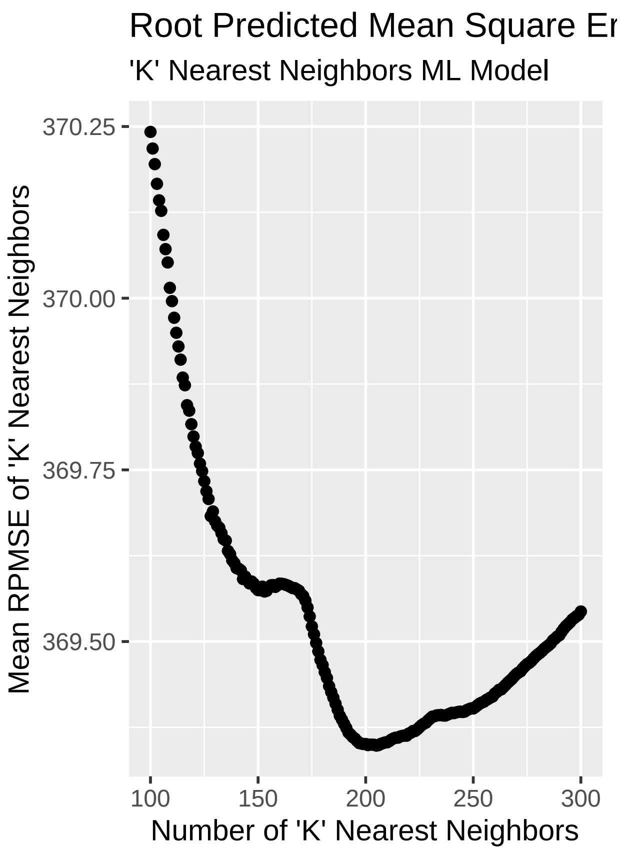
The first machine learning model I will use is Random Forests. The model will build a forest of ecologically diverse, bootstrapped, classification trees. Working backwards, a classification tree is a model that splits data into homogenous pieces (leaves) using the explanatory variables; this process of splitting data is also called recursive partitioning. The splits are determined by finding variables and values that decrease errors the most. Next, bootstrapping refers to sampling data with replacement. Finally, ecologically diverse refers to using only some of the explanatory variables on each tree. To predict the number of 'TotalInteractions' a post might have, I will be cycling through different minimum numbers of leaves and have the model's trees vote on which classification to use with which explanatory variables. Whichever parameters creates the lowest Root Predicted Mean Square Error, a metric that explains how far off my predictions for the number of 'TotalInteractions' for a FaceBook post are from the true number of 'TotalInteractions' for a FaceBook post, on average. Despite this model working like a black box (it is non-interpretable), it has several advantages. A Random Forest model captures non-linear and interaction relationships, it is fairly simple to implement, it is stable (i.e. it has a low variance), it works well for small data sets, can be used for classification and regression, and contains an automatic cross validation method.

The second machine learning model I will use is 'K' Nearest Neighbors. The model will plot all observations then compare the unknown, new observation to the 'K' number of its closest neighbors. The unknown observation's classification will be "voted" on by the majority of its neighbors. To determine the number of 'TotalInteractions', I will be cycling through different possible numbers of 'K' that could best determine the true number of 'TotalInteractions' a FaceBook post could possibly have, determined by the Root Predicted Mean Square Error. It is a very simple machine learning model to use, though it can have a few drawbacks: it can be computationally expensive for large data sets, nothing is really "close" in large dimensions, it is sensitive to irrelevant predictors, and it is sensitive to unbalanced data sets.

3. For each of your chosen ML models, choose the best values for your tuning parameters via 5-fold cross validation. Provide a plot of the RPMSE metric as a function of your tuning parameters (averaged over the values of the others if necessary). Identify the best tuning parameters.

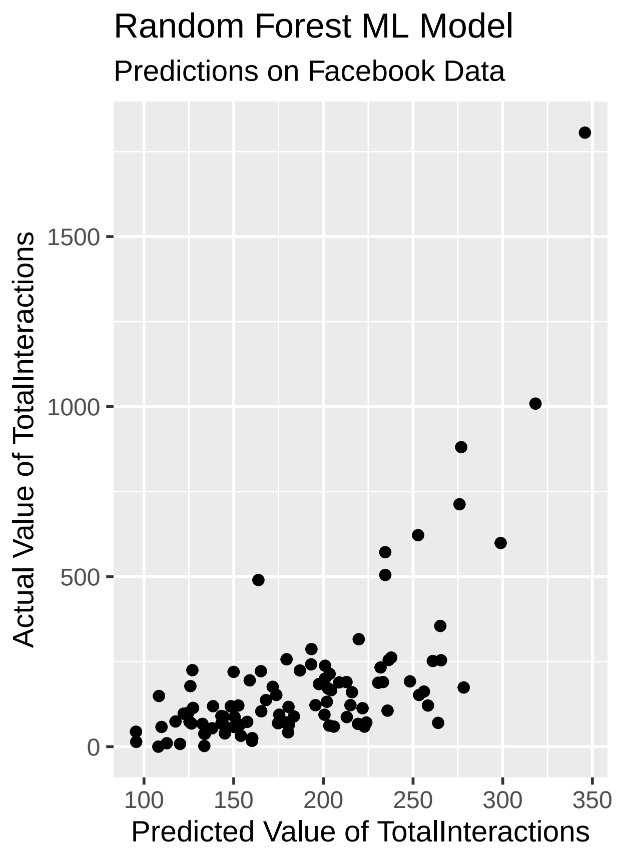
Using 5-fold cross validation, I have determined the best tuning parameters for each machine learning model, based on their Root Predicted Mean Square Errors. For the Random Forest machine learning model, the best parameters for a 500 classification tree forest are 20 minimum leaf nodes and 1 variable to split on. For the ‘K’ Nearest Neighbors machine learning model, the best parameter for a data set of 399 variables is 205 neighbors. Below are the graphs which show the Root Predicted Mean Square Error metric as a function of the tuning parameters.

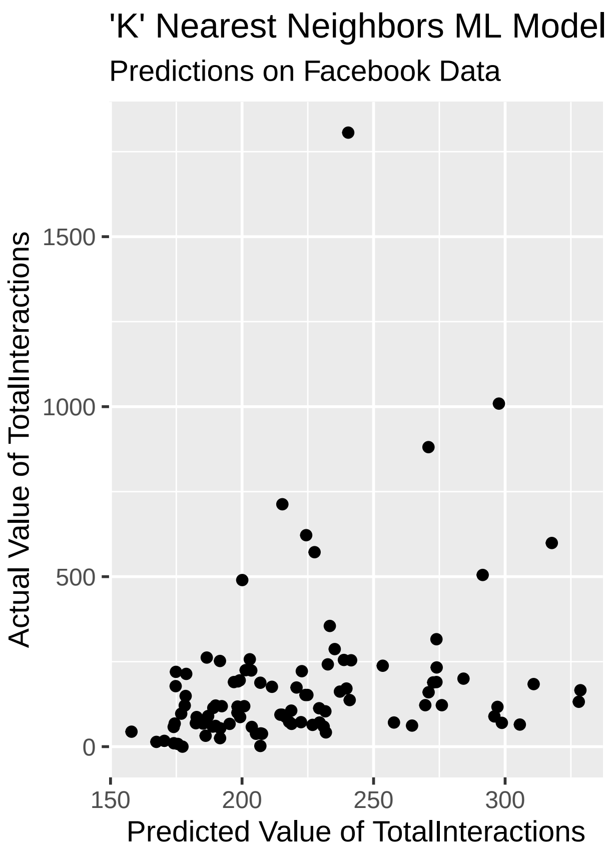




4. Using your best tuning parameters, generate predictions for each observation in `FacebookTesting.csv`. Report your RPMSE on the test data using `FacebookAllTesting.csv`.

As can be seen in the graphs below, I used the best tuning parameters for each machine learning model, as stated above, to generate predictions for each observation in `FacebookAllTesting.csv`. The Root Predicted Mean Square Error for the Random Forest machine learning model was 208.4852844. The Root Predicted Mean Square Error for the ‘K’ Nearest Neighbors machine learning model was 232.4059066. The Random Forest machine learning model had a lower Root Predicted Mean Square Error, so it performed better than the ‘K’ Nearest Neighbors machine learning model.





Appendix of Code

```
knitr::opts_chunk$set(echo = FALSE, include = FALSE)
library(vroom) #faster data reading
library(dplyr) #mutate
library(DataExplorer) #for plot explorations
library(viridis) #for color-blind friendly graphs
library(ranger) #for random forest
library(ggplot2) #for professional looking graphics
library(caret) #for k-fold cross validation
library(kknn) #for k nearest neighbors
library(knitr) #for pretty tables with kable
# library(kableExtra) #for if kable needs to be landscape
library(gridExtra) #for making grids on same row
options(scipen = 999) #for preventing scientific notation
fb = vroom("~/R programming/STAT_330/FacebookTraining.csv", show_col_types = FALSE)
#fb_test <- vroom("~/R programming/STAT_330/FacebookTesting.csv", show_col_types = FALSE)
fb_all <- vroom("~/R programming/STAT_330/FacebookAllTesting.csv", show_col_types = FALSE)
fb <- fb %>% mutate (Type = as.factor(Type), Category = as.factor(Category),
                     Weekday = as.factor(Weekday))
fb_all <- fb_all %>% mutate(Type = as.factor(Type), Category = as.factor(Category),
                     Weekday = as.factor(Weekday))

plot_intro(fb)
```

```r
plot_missing(fb)
fb <- subset(fb, !is.na(Paid))
plot_correlation(fb, type = "continuous")
## Train a Random forest
#Set a seed for reproducibility
set.seed(16)

# Set Possible Tuning Parameters
mtry.grid.rf <- 1:7 # number of variables to split on when building a tree
n_trees.rf <- 500 #how big the forest is
min_n.rf <- c(5, 10, 15, 20) #fewest obs allowed in a leaf node of any given tree
tune.grid.rf <- expand.grid(mtry = mtry.grid.rf, min_n= min_n.rf) #grid of all parameters

# Split dataset into K-pieces for K-fold cross validation
K.rf.cv <- 5
folds.rf <- createFolds(fb$TotalInteractions, k = K.rf.cv, list = FALSE)

# Vector to hold prediction error
rpmse.rf <- matrix(0, nrow = nrow(tune.grid.rf), ncol = K.rf.cv)

# Run Cross-validation
#tuning double loop
for(p in 1:nrow(tune.grid.rf)){
  for(cv in 1:K.rf.cv){

    # Fit the RF
    rf1 <- ranger(formula = TotalInteractions~.,
                  data=fb[folds.rf!=cv,],
                  num.trees=n_trees.rf,
                  mtry=tune.grid.rf$mtry[p],
                  min.node.size=tune.grid.rf$min_n[p])

    # Predict the test set
    preds.rf <- predict(rf1, data=fb[folds.rf == cv,])

    # Calculate RPMSE
    rpmse.rf[p,cv] <- sqrt(mean((fb$TotalInteractions[folds.rf == cv] -
                                   preds.rf$predictions)^2))
  }
}

rpmse.rf <- rowMeans(rpmse.rf)

rf_graph <- ggplot() +
  geom_raster(aes(x=tune.grid.rf$mtry, y=tune.grid.rf$min_n, fill=rpmse.rf)) +
  scale_fill_viridis() +
  xlab("Number of Variables to Split On") +
  ylab("Minimum Number of Observations (Per Leaf Node)") +
  ggtitle("Root Predicted Mean Square Error",
          subtitle = "Random Forest ML Model") +
  guides(fill = guide_legend(title = "RMPSE"))

# Retrain the RF using the best setting
```

```r
mtry1 <- tune.grid.rf$mtry[which.min(rpmse.rf)]
min_n1 <- tune.grid.rf$min_n[which.min(rpmse.rf)]
rf2 <- ranger(formula = TotalInteractions~., data = fb_all, num.trees = n_trees.rf,
             mtry = mtry1, min.node.size = min_n1)
#rf2["min.node.size"] = 15
#rf2["mtry"] = 1
#Number of trees: 500


## Train a K Nearest Neighbors
#Set a seed for reproducibility
set.seed(16)

# Set Possible Tuning Parameters
K.knn <- seq(100, 300) #number of neighbors

## Split dataset into K-pieces
K.knn.cv <- 5
folds.knn <- createFolds(fb$TotalInteractions, k=K.knn.cv, list=FALSE)

## Vector to hold prediction error
rpmse.knn.grid <- matrix(NA, nrow=length(K.knn), ncol=K.knn.cv)

## Tuning Double Loop
for(i in 1:length(K.knn)){
  for(k in 1:K.knn.cv){
    ## Split test/train
    testSet <- subset(fb, folds.knn == k)
    trainSet <- subset(fb, folds.knn != k)

    ## Fit K Nearest Neighbors on Training Set (specifics on this below)
    knn1 <- kknn(TotalInteractions~., train=trainSet, test=testSet, k=K.knn[i])

    # Predict the test set and save prediction error
    # preds.knn <- predict(knn, newdata=fb)
    preds.knn <- predict(knn1, data=testSet)
    rpmse.knn.grid[i,k] <- sqrt(mean((testSet$TotalInteractions - preds.knn)^2))

  }
}

rpmse.knn <- rowMeans(rpmse.knn.grid)

knn_graph <- ggplot() +
  geom_point(aes(x=K.knn, y=rpmse.knn)) +
  xlab("Number of \'K\' Nearest Neighbors") +
  ylab("Mean RPMSE of \'K\' Nearest Neighbors") +
  ggtitle("Root Predicted Mean Square Error",
          subtitle = "\'K\' Nearest Neighbors ML Model")

# Retrain the RF using the best setting
K.knn1 <- K.knn[which.min(rpmse.knn)]
knn2 <- kknn(TotalInteractions~., train=fb, test=fb_all, k=K.knn1)
```

```r
#K.knn1 =~ to 200ish
grid.arrange(rf_graph, knn_graph, nrow = 1)
#Set a seed for reproducibility
set.seed(16)

# Use the best RF model to predict
fb_preds.rf <- predict(rf2, data=fb_all)
rf.pred.plot <- qplot(fb_preds.rf$predictions, fb_all$TotalInteractions, geom="point") +
  xlab("Predicted Value of TotalInteractions") +
  ylab("Actual Value of TotalInteractions") +
  ggtitle("Random Forest ML Model",
          subtitle = "Predictions on Facebook Data")
RPMSE.rf <- sqrt(mean((fb_preds.rf$predictions-fb_all$TotalInteractions)^2))

# Use the best KNN model to predict
fb_preds.knn <- predict(knn2, data=fb_all)
knn.pred.plot <- qplot(fb_preds.knn, fb_all$TotalInteractions, geom="point") +
  xlab("Predicted Value of TotalInteractions") +
  ylab("Actual Value of TotalInteractions") +
  ggtitle("\'K\' Nearest Neighbors ML Model",
          subtitle = "Predictions on Facebook Data")
RPMSE.knn <- sqrt(mean((fb_preds.knn-fb_all$TotalInteractions)^2))
grid.arrange(rf.pred.plot, knn.pred.plot, nrow = 1)
#End of homework's code
```