```
# Jonothan Meyer
# Data Science - MTH 3270
# 05/12/21
# Final Project - Midterm Project 3

# 1. Every machine learning procedure has at least one tuning parameter, whose
value you
# choose, that controls the model complexity, that is, how closely the fitted model
is able to
# conform to the data:
#   -Decision tree. The tuning parameters are: 1) The minimum size of a node in
order
#    for a split to be attempted; 2) The complexity parameter, for which a split is
only
#    performed if it decreases the misclassification rate by this percent or more.
# - Random forest: The tuning parameter is the number of variables to use in each
tree.
# - K nearest neighbor: The tuning parameter is the number of neighbors, k.
# - Artificial neural network: The tuning parameter is the number of hidden units,
k.

# Your first task is to separate the small business data set randomly into 75%
training and
# 25% testing sets, then fit one of the above machine learning models (your choice)
to the
# training set using using at least three different values of the tuning parameter,
and
# compare the effectiveness of each model for classifying individuals in the test
set. Your
# model should predict one of the following categorical variables (your choice).
You may
# use any explanatory (X) variables, but they must be numerical (not categorical).

# -The hours per week spent managing or working the business by the first business
# owner HOURS1. You'll need to convert the 1, 2, ..., 7 values to "character" (so
# they won't be treated as numerical responses by the model-fitting function in R)
#  using as.character() with mutate().

small_business <- read.csv("C:\\Users\\Jonothan\\Desktop\\MSU-Spring2021\\Data
Science\\Midterm\\icesiv_contest.csv")
names(small_business)
head(small_business)
#sb <- small_business %>% select(AGE1, PAYROLL_NOISY, EMPLOYMENT_NOISY,
                                #RECEIPTS_NOISY, PRMINC1, EDUC1,HOURS1,PCT1)

require(timeR)
#install.packages("timeR")
library(timeR)
library(dplyr)
library(ggplot2)
library(rpart)
library(randomForest)
library(partykit)
library(class)
library(caret)

#------------------------------------------#1 KNN w/differnt values of
k-----------------------------------------------------
small_business <- read.csv("C:\\Users\\Jonothan\\Desktop\\MSU-Spring2021\\Data
```

```
Science\\Midterm\\icesiv_contest.csv") #Pull Data

# Separating into training/testing
sb_knn <- small_business %>% select(AGE1, PAYROLL_NOISY, EMPLOYMENT_NOISY, #
Pulling Relevant Columns
                                    RECEIPTS_NOISY, PCT1, EDUC1)
rm(small_business) # This variable uses up a ton of memory. So after I have what I
need it is removed from the global enviroment
#sb_knn <- small_business %>% select(AGE1, PAYROLL_NOISY, EDUC1)
sb_knn <- sb_knn[complete.cases(sb_knn),] #Removing all NA's

set.seed(1) # Making sure sample draws are consistent
data_set_size <- round(floor(nrow(sb_knn))*.75) # Getting size for traininig data
(75%)
indexes <- sample(1:nrow(sb_knn), size = data_set_size) # Getting Indicies for
training data
training <- sb_knn[indexes,] # Pulling training data
testing <- sb_knn[-indexes,] # Pulling testing data

nrow(training)/(nrow(training)+nrow(testing)) # Making sure training data
proportions are correcct
nrow(testing)/(nrow(training)+nrow(testing)) # Making sure testing data proportions
are correct

#BUILDING AND ASSESSING KNN W/VARIOUS VALUES OF K
timer1 <- createTimer() #Want to time how long all the predictions take
timer1$start("Making y_pred, diff k's") # Starting timer

# Knn w/ k = 1 has a 69.88% accuracy rate
y_pred1 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
1)
conf1 <- confusionMatrix(y_pred1, factor(testing$EDUC1))
c1 <- conf1$overall[1]
c1

# Knn w/ k = 2 has a 67.68% accuracy rate
y_pred2 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
2)
conf2 <- confusionMatrix(y_pred2, factor(testing$EDUC1))
c2 <- conf2$overall[1]


# Knn w/ k = 3 has 67.44% accuracy rate
y_pred3 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
3)
conf3 <- confusionMatrix(y_pred3, factor(testing$EDUC1))
c3 <- conf3$overall[1]


# Knn w/ k = 5 has 66.83 accuracy rate
y_pred4 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
5)
conf4 <- confusionMatrix(y_pred4, factor(testing$EDUC1))
c4 <- conf4$overall[1]

# Knn w/ k = 9 has 66.83 accuracy rate
y_pred5 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
9)
conf5 <- confusionMatrix(y_pred5, factor(testing$EDUC1))
```

```
c5 <- conf5$overall[1]

# Knn w/ k = 15 has 65.27 accuracy rate
y_pred6 <- knn(train = training, test = testing, cl = factor(training$EDUC1), k =
15)
conf6 <- confusionMatrix(y_pred6, factor(testing$EDUC1))
c6 <- conf6$overall[1]

timer1$stop("Making y_pred, diff k's") # Stopping timer
tot_time_min <- toString(timer1$eventTable[4]/60) # Converting time to minutes)
cat("Total Time (min) to build all KNN's: ", tot_time_min, "min") # print time

accuracies <- data.frame(k = c(1, 2, 3, 5, 9, 15), accuracy = c(c1, c2, c3, c4, c5,
c6))
accuracies



ggplot(data = accuracies, aes(x = k, y = accuracy)) +
  geom_point() + geom_line() + ylab("Accuracy / 100") + ggtitle("Accuracy for
Different Values of k")


# Looking at the distribution for how observations were classified based off
educaton
plot(y_pred1, main = "k = 1")
plot(y_pred2, main = "k = 2")
plot(y_pred3, main = "k = 3")
plot(y_pred4, main = "k = 5")
plot(y_pred5, main = "k = 9")
plot(y_pred6, main = "k = 15")


#1 = Less than High School, 2 = High School, 3 = Technical School, 4 = Some
College,
#5 = Associate's, 6 = Bachelor's, 7 = Master's or higher


#-----------------------------------------------------------------------------------
---------------------------------------

#Your second task is to carry out a cluster analysis (hierarchical or k means, your
choice) to
#group the businesses into k clusters, where k is in the range 2-5 (your choice).
You must use
#four or more explanatory (X) variables in the cluster analysis, and they must be
numerical
#(not categorical). It's your choice which ones to use.
#Then inspect whether the clusters seem to correspond to whether a business is the
primary
#source of income for the first owner. To decide, look for whether businesses
within clusters
#largely are or aren't the primary source for the first owner (use the variable
PRMINC1). This
#can be an informal inspection or something more formal (e.g. computing a measure
of "purity"
#                                                         for each cluster) { your
choice.
#  (It's okay if the businesses don't cluster according to whether they're the
```

```
primary source of
#    income for the first owner.)
#    You're allowed to use only a subset of rows (observations) because clustering
procedures
#    are memory hogs and are computationally intensive. For example, to use just
businesses from
#    Sector 54 (of the North American Industry Classification System) that are
franchises, (if your
#                data set is named small_business) you might type:

library(mclust)

small_business <- read.csv("C:\\Users\\Jonothan\\Desktop\\MSU-Spring2021\\Data
Science\\Midterm\\icesiv_contest.csv")
#sb <- small_business[complete.cases(small_business),]
sb1 <- small_business %>% filter(SECTOR == 54 & FRANCHISE == 1) %>%
  select(EMPLOYMENT_NOISY, PAYROLL_NOISY, RECEIPTS_NOISY, PCT1, PRMINC1)
sb1 <- sb1[complete.cases(sb1),]

nrow(sb1) # Checking how many observations are left
center <- 3 # Choosing the 'amount' of clusters
sb1_clust <- sb1 %>% kmeans(centers = center) %>% fitted("classes") %>%
as.character() # Conducting clusters
clust <- kmeans(sb1, centers = center)
sb1 <- sb1 %>% mutate(cluster = sb1_clust) # Adding cluster classification column
to data
sb1 %>% group_by(cluster) %>% summarize (n = n()) # Summarizing clustering
sb2 <- sb1 %>% select(-cluster)
pairs(sb2,
      col = clust$cluster,
      main = "Scatterplot Matrix of Small Business",
      pch = 19) # Looking for clustering classification in terms of PRMINC1


length(levels(sb1$PRMINC1))
sb1$PRMINC1
#-----------------------Cluster
Plots-----------------------------------------------------
sb1 %>% ggplot(aes(x = EMPLOYMENT_NOISY, y = PAYROLL_NOISY)) +
  geom_point(aes(size = factor(PRMINC1), color = cluster), alpha = 0.5) + xlim(0,
50) + ylim(0, 500)

sb1 %>% ggplot(aes(x = EMPLOYMENT_NOISY, y = RECEIPTS_NOISY)) +
  geom_point(aes(color = factor(PRMINC1), shape = cluster), alpha = 0.5, size = 2)

sb1 %>% ggplot(aes(x = EMPLOYMENT_NOISY, y = PCT1)) +
  geom_point(aes(color = factor(PRMINC1), shape = cluster), alpha = 0.5, size = 2)

sb1 %>% ggplot(aes(x = PAYROLL_NOISY, y = PCT1)) +
  geom_point(aes(color = factor(PRMINC1), shape = cluster), alpha = 0.5, size = 2)

sb1 %>% ggplot(aes(x = RECEIPTS_NOISY, y = PCT1)) +
  geom_point(aes(color = factor(PRMINC1), shape = cluster), alpha = 0.5, size = 2)

biz_dist <- dist(sb1, method = "euclidean")
biz_hclust <- hclust(biz_dist)
plot(biz_hclust, cex = 0.7)
#---------------------------------------------------------------------------------
----------
```