Machine Learning 1

Lindley Slipetz

7/4/2021

For this homework I will be using my own dataset (psychology-related because I'm a quantitative psychology PhD student). We will be predicting suicides per 1000 from handguns per 1000, percentage of males in the population, percentage white in the population, and unemployment percentage in the population of Pennsylvania counties in 2018 (the data is actually longitudinal; but, for the sake of this assignment, I'll stick to one time point).

Let's load our packages!

```
#install.packages("tidyverse")
library(tidyverse)
#install.packages("tidymodels")
library(tidymodels)
library(caret)
#install.packages("RANN")
library(RANN)
```

Now, let's load the data!

```
sui_full_data <- read.csv("C:\\Users\\Owner\\Documents\\ICPSR\\ML\\HW 1\\ML_HW1.csv", header = TRUE)
```

Let's subset our data to the relevant columns and year.

```
sui_data <- sui_full_data %>%
   filter(year == 2018) %>%
  select(suicides_per1000, handguns_per1000, male_pct, white_pct, unemployment_pct)
```

Here, I standardize the predictors.

```
sui_stand <- as.data.frame(scale(sui_data))
```

2 Creating training and testing sets.

```
set.seed(2874187)
trainIndex <- createDataPartition(sui_stand$suicides_per1000, p=0.7, list = FALSE)
train <- sui_stand[trainIndex,]
test <- sui_stand[-trainIndex,]
```

OLS with training data

```
set.seed(164891264)
train_sui = lm(suicides_per1000 ~ ., data=train)
summary(train_sui)
```

```
##
## Call:
## lm(formula = suicides_per1000 ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5192 -0.4141 -0.0541  0.3106  4.6235
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.12018    0.14374   0.836  0.40773
## handguns_per1000 -0.27327    0.14443  -1.892  0.06522 .
## male_pct         -0.04764    0.12721  -0.375  0.70986
## white_pct         0.45989    0.13907   3.307  0.00191 **
## unemployment_pct  0.40553    0.14595   2.778  0.00806 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9805 on 43 degrees of freedom
## Multiple R-squared:  0.2753, Adjusted R-squared:  0.2079
## F-statistic: 4.085 on 4 and 43 DF,  p-value: 0.006806
```

The R^2 for the training model is 0.275, meaning 27.5% of the variance in the suicide variable is predicted by the other variables. The white percentage and unemployment percentage coefficient are positive and significant, meaning that as percent of white and unemployment increase, chance of suicide increases.

Comparing R^2

```
predictions <- train_sui %>% predict(test)
R2(predictions, test$suicides_per1000)
```

```
## [1] 0.01512037
```

The R^2 of the testing data is 0.015, meaning 1.5% of the variance is explained by the predictors. This is much less the R^2 of the training model. Perhaps this means the training model was overfit to the training data and, hence, was not flexible enough to fit the testing data.

5 Create missing data

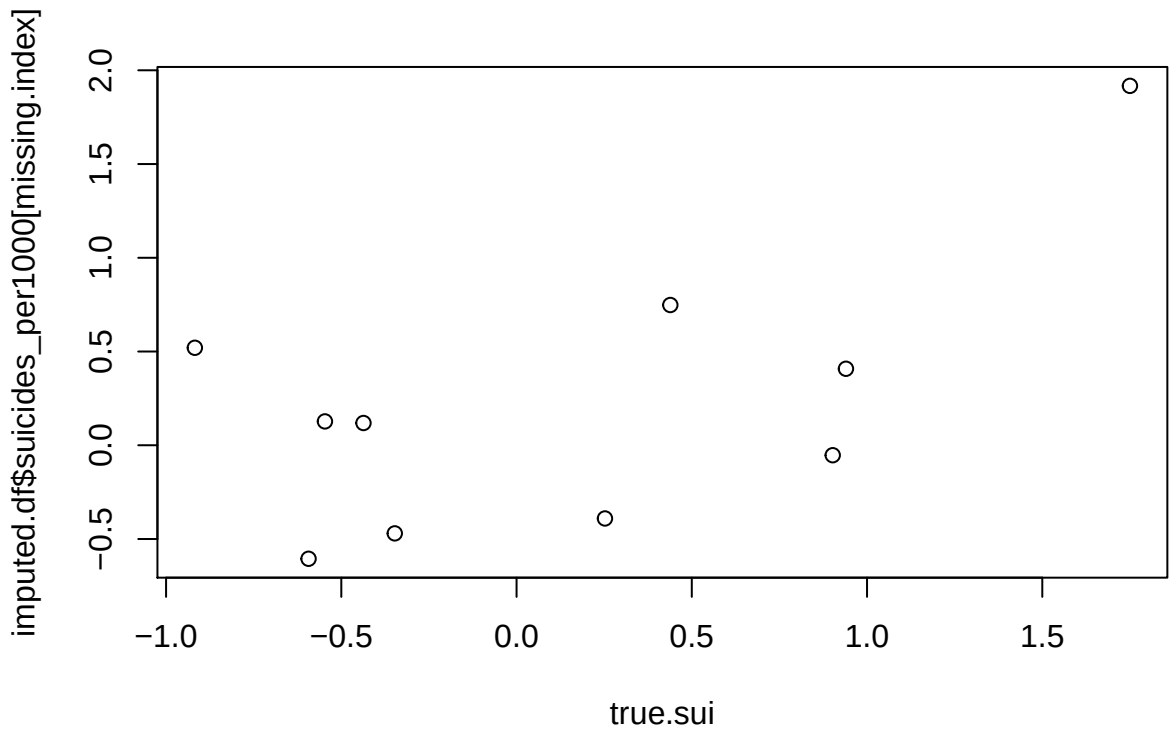
Since my data has way fewer observations, I thought I would take the same proportion of missing data from my data. That left me with 4 missing values, which I thought would be too few. So, I increased it to 10.

```
sui.missing <- sui_stand
set.seed(4725)
missing.index <- sample(1:67, 10)
true.sui <- sui.missing$suicides_per1000[missing.index]
sui.missing$suicides_per1000[missing.index] <- NA
```

6 k-nearest neighbors imputation

```
set.seed(345)
imputeValues <- preProcess(sui.missing, method="knnImpute", k=3)
imputed.df <- predict(imputeValues, sui.missing)
```

```
plot(true.sui, imputed.df$suicides_per1000[missing.index])
```




```
cor(true.sui, imputed.df$suicides_per1000[missing.index])
```

```
## [1] 0.6161882
```

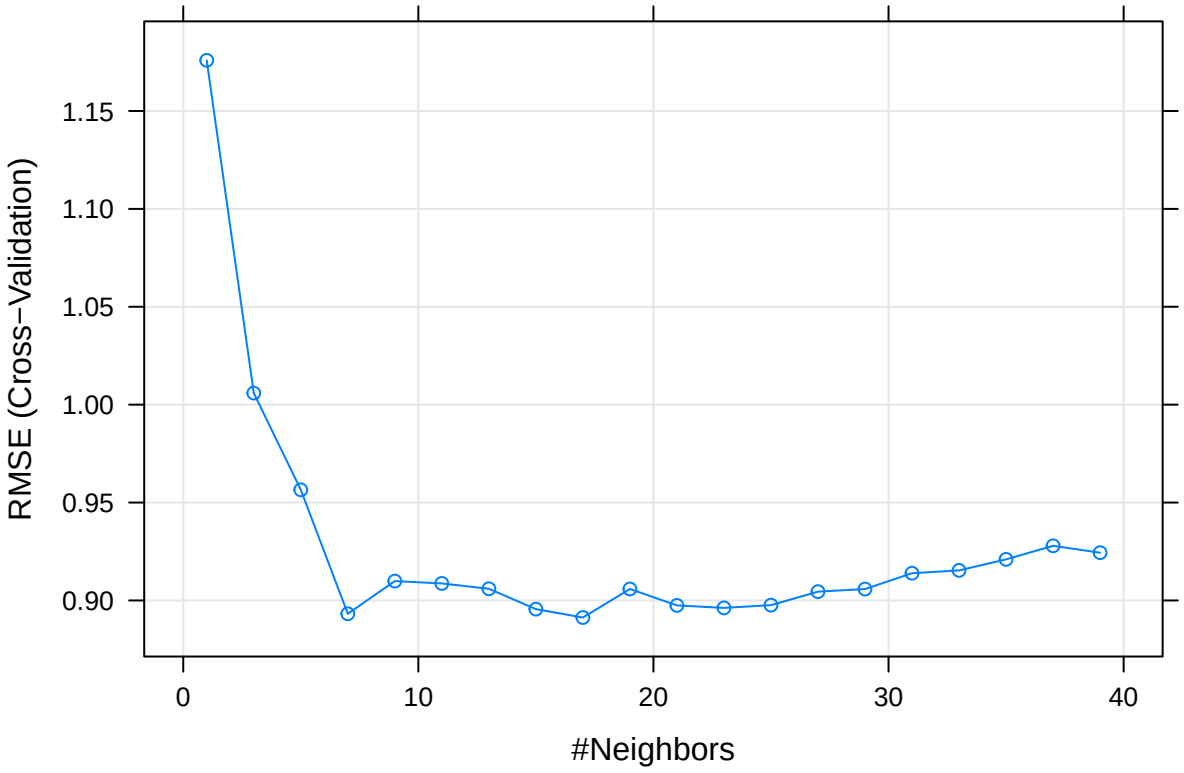
The correlation between the true values and the imputed values is 0.616, which isn't very good. Let's try again with an optimal k.

```
knnGrid <- expand.grid(k=seq(1,40,by=2))
nFolds <- 5  ## The number of folds * number of repeats
nTune <- nrow(knnGrid)  ## The number of tuning configurations tested
fitCtrl <- trainControl(method = "cv",
                        number = nFolds,
                        ## Search "grid" or "random"
                        search = "random")
#
knn.res <- train(suicides_per1000 ~ .,
                 data=na.omit(sui.missing),
                 method="knn",
                 trControl=fitCtrl,
                 tuneGrid=knnGrid,
                 #tuneLength=10,
                 metric="RMSE")
#
knn.res
```

```
## k-Nearest Neighbors
```

```
##
## 57 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 45, 45, 46, 47, 45
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared    MAE
##    1  1.1758735  0.08568105  0.8434441
##    3  1.0059161  0.01885856  0.7193673
##    5  0.9565153  0.06574970  0.6638717
##    7  0.8931915  0.11406387  0.6190829
##    9  0.9098921  0.07929895  0.6403129
##   11  0.9086873  0.10900261  0.6321589
##   13  0.9059636  0.11094873  0.6335382
##   15  0.8955617  0.17236017  0.6328048
##   17  0.8912808  0.14395187  0.6306063
##   19  0.9058590  0.10702904  0.6434417
##   21  0.8974655  0.15139655  0.6404055
##   23  0.8962030  0.17708571  0.6505419
##   25  0.8976072  0.13798298  0.6450614
##   27  0.9044951  0.12361056  0.6522707
##   29  0.9058098  0.12696665  0.6434875
##   31  0.9139051  0.09961204  0.6507233
##   33  0.9153531  0.11277659  0.6512119
##   35  0.9209794  0.12247441  0.6526524
##   37  0.9279091  0.07121114  0.6588522
##   39  0.9243996  0.14643018  0.6554930
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 17.
```

```
#
plot(knn.res)
```

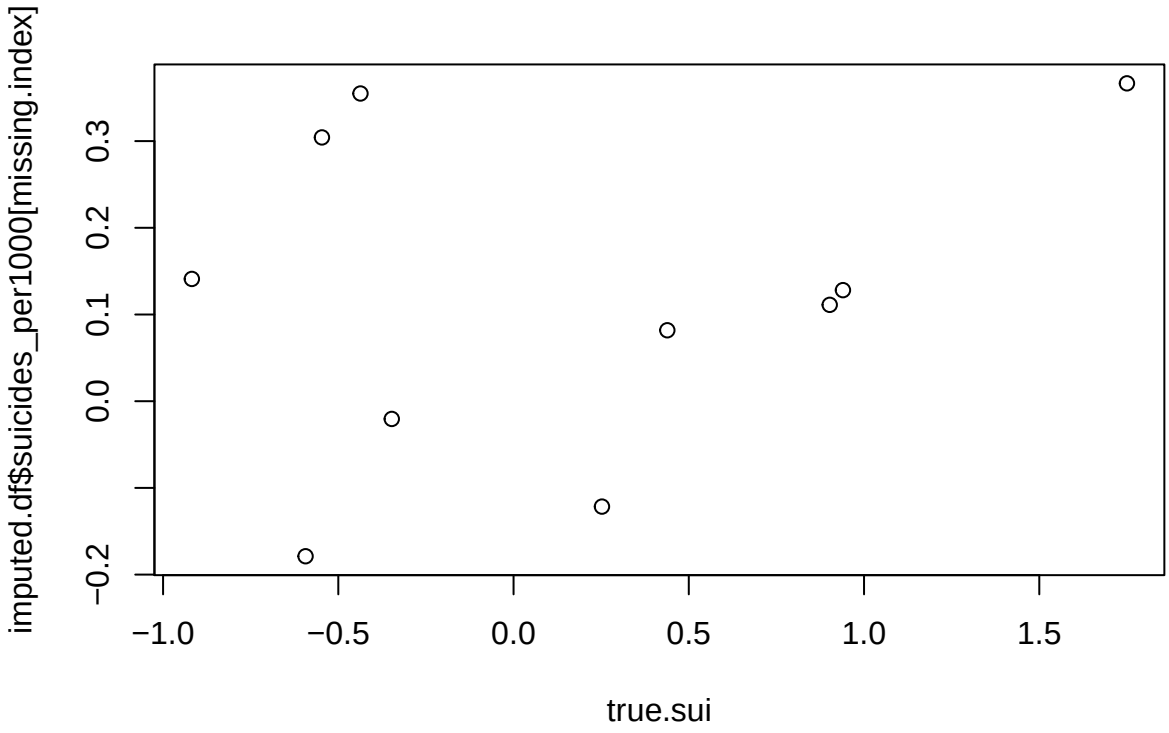



```
#
#
```

From that, we got k=21. Let's try the correlations again with that.

```
set.seed(47)
imputeValues <- preProcess(sui.missing, method="knnImpute", k=21)
imputed.df <- predict(imputeValues, sui.missing)
```

```
plot(true.sui, imputed.df$suicides_per1000[missing.index])
```




```
cor(true.sui, imputed.df$suicides_per1000[missing.index])
```

```
## [1] 0.2489866
```

Hmm. . . we got an even worse fit with a correlation of 0.245. Maybe k-nearest neighbors is not an appropriate imputation method for this data.