model_Tuning_KNN

Silvy S.

2024-06-25

A bit more EDA...

We have a high number of n relative to p.

Let's test for multicollinearity:

```
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)
library(car)
```

```
## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some
```

```
inst_clean <- read_csv("./inst_clean.csv")
```

```
## Rows: 23781 Columns: 14
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr  (2): BKCLASS, STNAME
## dbl (12): ASSET, DEP, DEPDOM, EQ, MUTUAL, NETINC, ROA, ROAPTX, ROAPTXQ, ROAQ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Create binary variable
commercial_bank_categories <- c("N", "NM", "SM", "NC")
inst_clean$CommercialBank <- ifelse(inst_clean$BKCLASS %in% commercial_bank_categories, 1, 0)
print(inst_clean)
```

```
## # A tibble: 23,781 x 15
##     ASSET BKCLASS    DEP DEPDOM     EQ MUTUAL NETINC    ROA ROAPTX ROAPTXQ  ROAQ
##     <dbl> <chr>    <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>   <dbl> <dbl>
##  1 7.09e4 NM      6.03e4 6.03e4 7.24e3      0   -226 -0.633 -0.879   -2.28 -1.71
##  2 4.40e5 N       3.02e5 3.02e5 5.37e4      0    818  0.749  0.944    0.94  0.75
##  3 4.47e5 N       3.88e5 3.88e5 3.09e4      0   3732  0.840  1.28     1.15  0.75
##  4 4.78e6 NM      4.08e6 4.08e6 4.42e5      0 -13573 -0.570 -0.770   -2.25 -1.7
##  5 3.63e5 NM      3.04e5 3.04e5 3.67e4      0    368  0.407  0.515    0.52  0.41
##  6 2.68e6 SM      2.43e6 2.43e6 2.22e5      0   3278  0.490  0.555    0.55  0.49
##  7 1.11e8 NM      8.99e7 8.99e7 9.18e6      0 343560  1.22   1.64     1.64  1.22
##  8 1.33e7 NM      1.18e7 1.18e7 1.30e6      0 209990  1.61   2.08     1.62  1.28
##  9 3.20e5 NM      2.96e5 2.96e5 1.89e4      0    540  0.667  0.667    0.67  0.67
## 10 2.42e6 N       2.18e6 2.18e6 2.10e5      0   7331  1.21   1.21     1.21  1.21
## # i 23,771 more rows
## # i 4 more variables: ROE <dbl>, STNAME <chr>, TRUST <dbl>,
## #   CommercialBank <dbl>
```

```
## Splitting
set.seed(12345)
training_pct <- 0.8
Z <- sample(nrow(inst_clean), floor(training_pct*nrow(inst_clean)))
inst.training <- inst_clean[Z, ]
inst.testing <- inst_clean[-Z, ]
c(nrow(inst_clean), nrow(inst.training), nrow(inst.testing))
```

```
## [1] 23781 19024  4757
```

```
lm_full <- lm(ASSET ~ BKCLASS + DEP + DEPDOM + EQ + NETINC + ROA + ROAPTX + ROAPTXQ + ROAQ,
              data = inst_clean)
summary(lm_full)
```

```
##
## Call:
## lm(formula = ASSET ~ BKCLASS + DEP + DEPDOM + EQ + NETINC + ROA +
##     ROAPTX + ROAPTXQ + ROAQ, data = inst_clean)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -127836179     -41429      -3591      19187  168737669
##
## Coefficients:
```

```
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.506e+03  3.229e+04   0.294  0.76847
## BKCLASSNC   -6.012e+04  2.262e+05  -0.266  0.79041
## BKCLASSNM   -3.027e+04  3.967e+04  -0.763  0.44538
## BKCLASSNS   -2.560e+04  4.862e+05  -0.053  0.95801
## BKCLASSSB    1.982e+05  6.717e+04   2.950  0.00318 **
## BKCLASSSI    1.937e+02  9.055e+04   0.002  0.99829
## BKCLASSSL    1.873e+05  6.420e+04   2.918  0.00353 **
## BKCLASSSM   -2.004e+04  5.799e+04  -0.346  0.72959
## DEP          1.126e+00  5.482e-03 205.315  < 2e-16 ***
## DEPDOM      -1.870e-01  4.725e-03 -39.580  < 2e-16 ***
## EQ           2.718e+00  3.045e-02  89.281  < 2e-16 ***
## NETINC       4.096e+00  2.127e-01  19.260  < 2e-16 ***
## ROA         -4.765e+03  1.606e+04  -0.297  0.76670
## ROAPTX       1.724e+02  1.494e+04   0.012  0.99080
## ROAPTXQ      1.402e+03  1.127e+04   0.124  0.90100
## ROAQ        -2.574e+03  1.204e+04  -0.214  0.83074
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2426000 on 23765 degrees of freedom
## Multiple R-squared:  0.9951, Adjusted R-squared:  0.9951
## F-statistic: 3.229e+05 on 15 and 23765 DF,  p-value: < 2.2e-16
```

```
vif(lm_full)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## BKCLASS  1.068870  7        1.004769
## DEP     84.832008  1        9.210429
## DEPDOM  47.706835  1        6.907013
## EQ      39.239187  1        6.264119
## NETINC   3.032791  1        1.741491
## ROA     86.556345  1        9.303566
## ROAPTX  83.352104  1        9.129737
## ROAPTXQ 70.612458  1        8.403122
## ROAQ    73.298891  1        8.561477
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
# Perform stepwise selection
stepwise_model <- stepAIC(lm_full, direction = "both", trace = FALSE)

# Display the summary of the final model
summary(stepwise_model)
```

```
##
## Call:
## lm(formula = ASSET ~ BKCLASS + DEP + DEPDOM + EQ + NETINC + ROA,
##     data = inst_clean)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -127837748     -41467      -3768      18871  168739673
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.041e+04  3.213e+04   0.324  0.74598
## BKCLASSNC   -5.406e+04  2.248e+05  -0.240  0.80995
## BKCLASSNM   -3.044e+04  3.965e+04  -0.768  0.44267
## BKCLASSNS   -2.571e+04  4.862e+05  -0.053  0.95782
## BKCLASSSB    1.982e+05  6.716e+04   2.951  0.00317 **
## BKCLASSSI   -8.407e+01  9.053e+04  -0.001  0.99926
## BKCLASSSL    1.875e+05  6.419e+04   2.922  0.00348 **
## BKCLASSSM   -2.019e+04  5.798e+04  -0.348  0.72772
## DEP          1.126e+00  5.482e-03 205.328  < 2e-16 ***
## DEPDOM      -1.870e-01  4.725e-03 -39.582  < 2e-16 ***
## EQ           2.718e+00  3.045e-02  89.288  < 2e-16 ***
## NETINC       4.095e+00  2.126e-01  19.262  < 2e-16 ***
## ROA         -5.794e+03  1.773e+03  -3.267  0.00109 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2426000 on 23768 degrees of freedom
## Multiple R-squared:  0.9951, Adjusted R-squared:  0.9951
## F-statistic: 4.037e+05 on 12 and 23768 DF,  p-value: < 2.2e-16
```

```
vif(stepwise_model)
```

```
##              GVIF Df GVIF^(1/(2*Df))
## BKCLASS  1.053190  7        1.003709
## DEP     84.831277  1        9.210390
## DEPDOM  47.706795  1        6.907011
## EQ      39.238731  1        6.264083
## NETINC   3.031857  1        1.741223
## ROA      1.055135  1        1.027198
```

- Ordinal Encoding - BKCLASS variable

- Missing values check

```
bkclass.levels <- unique(inst.training$BKCLASS)
bkclass.map <- setNames(as.integer(bkclass.levels), bkclass.levels)
```

```
## Warning in setNames(as.integer(bkclass.levels), bkclass.levels): NAs introduced
## by coercion
```

```
inst.training$BKCLASS <- as.integer(bkclass.map[inst.training$BKCLASS])
inst.testing$BKCLASS <- as.integer(bkclass.map[inst.testing$BKCLASS])

inst.training[is.na(inst.training)] <- 0
inst.testing[is.na(inst.testing)] <- 0
```

```
# Select predictors and response variable for training set
X.train <- inst.training[, c("BKCLASS", "DEP", "DEPDOM", "EQ", "NETINC", "ROA")]
Y.train <- inst.training$ASSET

# Select predictors and response variable for testing set
X.test <- inst.testing[, c("BKCLASS", "DEP", "DEPDOM", "EQ", "NETINC", "ROA")]
Y.test <- inst.testing$ASSET
```

- Select Predictors from VIF test ~ predictors best for asset

- Fit the KNN Model

- had to convert y.test to numeric,
- calculated MSE

```
library(class)
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
set.seed(12345)
k <- 3
Y.test <- as.numeric(as.character(Y.test))

asset.knn <- knn(train=X.train, test=X.test, cl=Y.train, k)

asset.knn <- as.numeric(as.character(asset.knn))


mse <- mean((Y.test - asset.knn)^2)
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 696108089028790"
```

```r
mae <- mean(abs(Y.test - asset.knn))
print(paste("Mean Absolute Error (MAE):", mae))
```

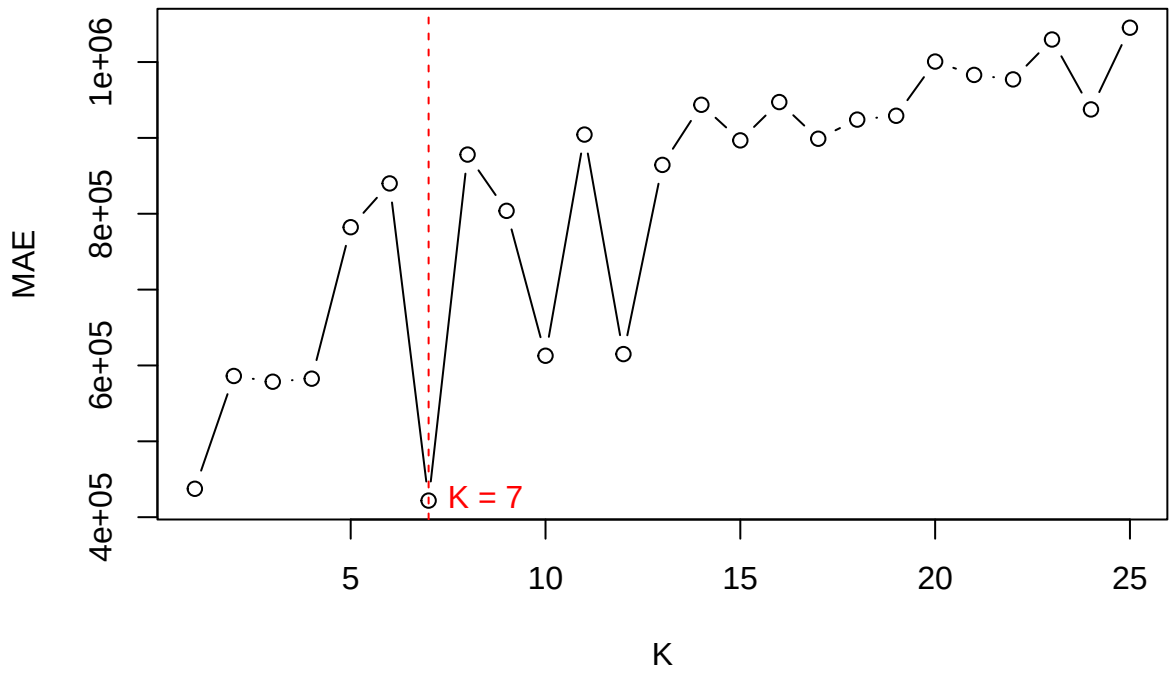
```
## [1] "Mean Absolute Error (MAE): 588126.133487511"
```

Tuning - tuned the model with optimal K

```r
library(class)  # Make sure the class package is loaded for the knn function
set.seed(12345)
Kmax <- 25  # Set the largest K to consider for this study.
mae <- rep(0, Kmax)  # Initialize MAE vector
for (i in 1:Kmax) {   knn.out <- knn(train = X.train, test = X.test, cl = Y.train, k = i)

mae[i] <- mean(abs(as.numeric(as.character(Y.test)) - as.numeric(as.character(knn.out)))) } # Find the

optimal_K <- which.min(mae)
optimal_MAE <- mae[optimal_K] # Plot the MAE values
plot(1:Kmax, mae, xlab = "K", ylab = "MAE", type = "b")
abline(v = optimal_K, col = "red", lty = 2)  # Add a vertical line for the optimal K
text(optimal_K, optimal_MAE, labels = paste("K =", optimal_K), pos = 4, col = "red") # Print the optima
```



```r
cat("The optimal K is:", optimal_K, "with a minimum MAE of:", optimal_MAE, "\n")
```

```
## The optimal K is: 7 with a minimum MAE of: 421864.2
```

```r
library(class)
library(caret)
set.seed(12345)
k <- 7
Y.test <- as.numeric(as.character(Y.test))

asset.knn <- knn(train=X.train, test=X.test, cl=Y.train, k)

asset.knn <- as.numeric(as.character(asset.knn))


mse <- mean((Y.test - asset.knn)^2)
print(paste("Mean Squared Error (MSE):", mse))
```

```
## [1] "Mean Squared Error (MSE): 213502885486152"
```

```r
mae <- mean(abs(Y.test - asset.knn))
print(paste("Mean Absolute Error (MAE):", mae))
```

```
## [1] "Mean Absolute Error (MAE): 487179.270548663"
```

KNN Model 2 - BKCLASS

```r
# predicting whether or not the bank is a commericial bank or not

logit_inst <- lm(CommercialBank ~ ASSET + DEP + DEPDOM + EQ + NETINC + ROA + ROAPTX + ROAPTXQ + ROAQ, da
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

```r
summary(logit_inst)
```

```
##
## Call:
## lm(formula = CommercialBank ~ ASSET + DEP + DEPDOM + EQ + NETINC +
##     ROA + ROAPTX + ROAPTXQ + ROAQ, data = inst.training, family = "binomial")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6810  0.1718  0.1783  0.1826  5.2366
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.154e-01  2.872e-03 283.867  < 2e-16 ***
```

```
## ASSET       -4.002e-09  1.367e-09  -2.927 0.003422 **
## DEP          2.680e-09  1.573e-09   1.704 0.088489 .
## DEPDOM      -6.885e-10  8.110e-10  -0.849 0.395927
## EQ           2.071e-08  6.813e-09   3.040 0.002370 **
## NETINC       1.912e-07  3.764e-08   5.080 3.82e-07 ***
## ROA          1.099e-02  2.956e-03   3.719 0.000201 ***
## ROAPTX      -4.838e-03  2.805e-03  -1.725 0.084584 .
## ROAPTXQ      3.127e-03  2.205e-03   1.418 0.156291
## ROAQ        -2.653e-03  2.285e-03  -1.161 0.245520
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3874 on 19014 degrees of freedom
## Multiple R-squared:  0.02734,    Adjusted R-squared:  0.02688
## F-statistic: 59.37 on 9 and 19014 DF,  p-value: < 2.2e-16
```

```
# Perform stepwise selection
stepwise_model_class <- stepAIC(logit_inst, direction = "both", trace = FALSE)
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
##  extra argument 'family' will be disregarded
```

```
# Display the summary of the final model
summary(stepwise_model_class)
```

```
##
## Call:
## lm(formula = CommercialBank ~ ASSET + DEP + EQ + NETINC + ROA,
##     data = inst.training, family = "binomial")
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8055  0.1718  0.1784  0.1830  5.2104
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.146e-01  2.825e-03 288.402  < 2e-16 ***
## ASSET       -3.951e-09  1.365e-09  -2.894  0.00381 **
## DEP          1.988e-09  1.340e-09   1.484  0.13791
## EQ           2.098e-08  6.806e-09   3.083  0.00205 **
## NETINC       1.871e-07  3.731e-08   5.013 5.39e-07 ***
## ROA          6.594e-03  3.129e-04  21.074  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3874 on 19018 degrees of freedom
## Multiple R-squared:  0.0271, Adjusted R-squared:  0.02684
## F-statistic: 105.9 on 5 and 19018 DF,  p-value: < 2.2e-16
```

```
vif(stepwise_model_class)
```

```
##      ASSET        DEP         EQ     NETINC        ROA
## 193.381216 116.294052  45.314158   2.192892   1.009736
```

```
library(caret)
library(class)

inst.training <- na.omit(inst.training)
inst.testing <- na.omit(inst.testing)


X.train <- inst.training[, c("ASSET", "DEP","EQ", "NETINC", "ROA")]
Y.train <- inst.training$CommercialBank


X.test <- inst.testing[,c("ASSET", "DEP","EQ", "NETINC", "ROA")]
Y.test <- inst.testing$CommercialBank


k <- 5
bkclass.knn <- class::knn(train=X.train, test=X.test, cl=Y.train, k)


table(Y.test, bkclass.knn)
```

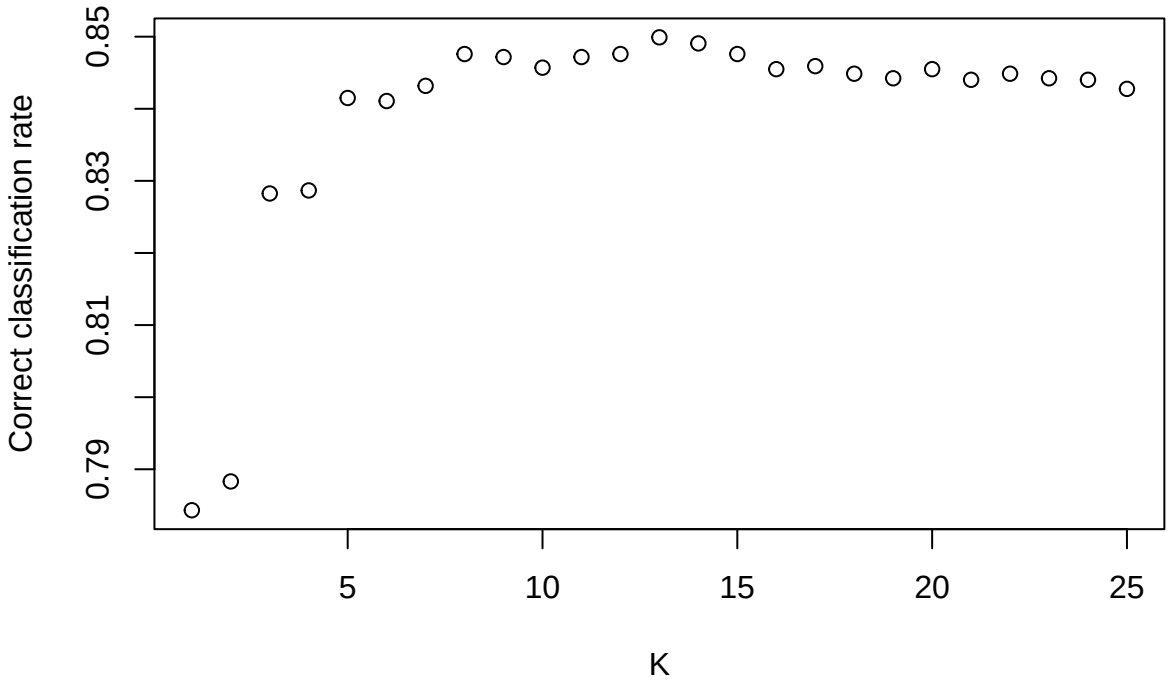
```
##       bkclass.knn
## Y.test    0    1
##      0  317  602
##      1  152 3686
```

```
mean(Y.test == bkclass.knn)
```

```
## [1] 0.8414967
```

Tuning

```
Kmax <- 25 # Set the largest K I would consider for this study.
class.rate <- rep(0, Kmax)
for (i in 1:Kmax) {
knn.out <- knn(train=X.train, test=X.test, cl=Y.train, k = i)
class.rate[i] <- mean(Y.test == knn.out)
}
plot(c(1:Kmax), class.rate, xlab="K", ylab="Correct classification rate")
```




```
k.opt <- which.max(class.rate)
c(k.opt, class.rate[which.max(class.rate)]) # Optimal K
```

```
## [1] 13.0000000  0.8499054
```

Tuning- with optimal K

```
library(caret)
library(class)

inst.training <- na.omit(inst.training)
inst.testing <- na.omit(inst.testing)


X.train <- inst.training[, c("ASSET", "DEP","EQ", "NETINC", "ROA")]
Y.train <- inst.training$CommercialBank


X.test <- inst.testing[,c("ASSET", "DEP","EQ", "NETINC", "ROA")]
Y.test <- inst.testing$CommercialBank


k <- 13 # Optimal K
```

```r
bkclass.knn <- class::knn(train=X.train, test=X.test, cl=Y.train, k)


table(Y.test, bkclass.knn)
```

```
##       bkclass.knn
## Y.test    0    1
##      0  265  654
##      1   60 3778
```

```r
mean(Y.test == bkclass.knn)
```

```
## [1] 0.8499054
```