model_fitting

lisa liubovich

2024-06-13

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.
```

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

```
## Warning in model.matrix.default(mt, mf, contrasts): the response appeared on
## the right-hand side and was dropped
```

```
## Warning in model.matrix.default(mt, mf, contrasts): problem with term 6 in
## model.matrix: no columns are assigned
```

```
vif_values_train <- vif(lm_train)
```

```
## Warning in model.matrix.default(mod, data = structure(list(NETINC = c(885, :
## the response appeared on the right-hand side and was dropped
```

```
## Warning in model.matrix.default(mod, data = structure(list(NETINC = c(885, :
## problem with term 6 in model.matrix: no columns are assigned
```

```
print(vif_values_train)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## ASSET   1.924570e+02  1       13.872888
## BKCLASS 1.059256e+00  7        1.004120
## DEP     1.599229e+02  1       12.646063
## DEPDOM  3.377787e+01  1        5.811873
## EQ      4.006568e+01  1        6.329746
## NETINC  2.108450e+06  0             Inf
## ROA     9.030026e+01  1        9.502645
## ROAPTX  8.746630e+01  1        9.352342
## ROAPTXQ 7.508461e+01  1        8.665138
## ROAQ    7.682353e+01  1        8.764903
```

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

```
## Warning in model.matrix.default(mt, mf, contrasts): the response appeared on
## the right-hand side and was dropped
```

```
## Warning in model.matrix.default(mt, mf, contrasts): problem with term 6 in
## model.matrix: no columns are assigned
```

```
vif_values_test <- vif(lm_test)
```

```
## Warning in model.matrix.default(mod, data = structure(list(NETINC = c(343560, :
## the response appeared on the right-hand side and was dropped
```

```
## Warning in model.matrix.default(mod, data = structure(list(NETINC = c(343560, :
## problem with term 6 in model.matrix: no columns are assigned
```

```
print(vif_values_test)
```

```
##                 GVIF Df GVIF^(1/(2*Df))
## ASSET   2.052970e+03  1       45.309710
## BKCLASS 1.128698e+00  7        1.008685
## DEP     3.601127e+03  1       60.009394
## DEPDOM  6.557843e+02  1       25.608287
## EQ      2.144865e+02  1       14.645356
## NETINC  1.050030e+08  0             Inf
## ROA     8.286899e+01  1        9.103241
## ROAPTX  8.018192e+01  1        8.954436
## ROAPTXQ 6.994068e+01  1        8.363055
## ROAQ    7.249624e+01  1        8.514472
```

Since a lot of our variables are correlated, we can use ridge regression.

Silvy: KNN

Ryan: linear and ridge regression

Lisa: bootstrap and logistic

Method Selection

The concept of the bias-variance trade-off has been central to our learning so far, so we really wanted that to be the focus of our project. As a result, we selected methods over a range of flexibility and restrictiveness as well as disadvantages/advantages that accompany certain methods regarding our ratio of n to p. Here is why we chose each model:

1. Linear Regression:
    1. advantages:
        1. low variance → simple model, makes it less sensitive to fluctuations in the training data
        2. interpret-ability is simple
    2. disadvantages:
        1. high bias: if the true relationship between predictors and response is not linear, linear regression will underfit the data
        2. the assumptions of linear regression (linearity, independence of errors, constant variance, and normally distributed errors) may not hold in practice

2. Logistic Regression:
    1. advantages:
        1. low variance for large samples → especially useful given how large our sample size is
        2. effective for binary outcomes,
    2. disadvantages:
        1. high bias (like linear reg)
        2. modeling limitations → may struggle with complex boundaries in the data, which may lead to underfitting
3. KNN
    1. advantages:
        1. low bias → KNN is nonparametric and can fit very complex decision boundaries
        2. flexibility → can model complex patterns in the data more accurately
    2. disadvantages:
        1. high variance → very sensitive to the specific training data and prone to over fitting when k is small
        2. computational cost is high for large datasets
4. ridge regression
    1. advantages
        1. penalty for large coefficients manages the bias-variance trade-off by reducing variance without significantly increasing bias
        2. handles multicollinearity, which is important in our data that very much suffers from multi-collinearity
    2. disadvantages:
        1. moderate bias might lead to underfitting
        2. linear assumptions might also lead to bias if the true relationship is not linear
5. boostrap
    1. advantages:
        1. low variance estimates
        2. allows for better understanding and estimation of the bias and variance of model estimates → which helps in model selection and tuning
    2. disadvantages:
        1. no direct impact on bias or variance → helps assess these properties
        2. computationally complex, especially given our large dataset

By using this specific set of methods, we can properly see the advantages and disadvantages of each particular method on our large dataset and which provides the most accurate and consistent predictions as a result.

Fitting a Logistic Regression

We're gonna convert BKCLASS into a binary variable just for this specific equation.

A classification code assigned by the FDIC based on the institution's charter type (commercial bank or savings institution), charter agent (state or federal), Federal Reserve membership status (Fed member, Fed non-member) and its primary federal regulator (state chartered institutions are subject to both federal and state supervision). **N** - Commercial bank, national (federal) charter, Fed member, and supervised by the Office of the Comptroller of the Currency (OCC); **NM** - Commercial bank, state charter, Fed non-member, and supervised by the Federal Deposit Insurance Corporation (FDIC); **OI** - Insured U.S. branch of a foreign chartered institution (IBA) and supervised by the OCC or FDIC; SB Federal savings banks, federal charter, supervised by the OCC or before July 21,2011 the Office of Thrift Supervision (OTS); **SI** - State chartered stock savings banks, supervised by the FDIC; **SL** - State chartered stock savings and loan associations, supervised by the FDIC or before July 21,2011 the OTS; **SM** - Commercial bank, state charter, Fed member, and supervised by the Federal Reserve Bank (FRB); **NC** Noninsured non-deposit commercial banks and/or trust companies regulated by the OCC, a state, or a territory; **NS** - Noninsured stock savings bank supervised by a state or territory; **CU** - state or federally chartered credit unions supervised by the National Credit Union Association (NCUA).

Our binary variable will split these categories into two categories: commercial and non commericial

```
commercial_bank_categories <- c("N", "NM", "SM", "NC")

# Create binary variable
inst.training$CommercialBank <- ifelse(inst.training$BKCLASS %in% commercial_bank_categories, 1, 0)
print(inst.training)
```

```
## # A tibble: 19,024 x 15
##       ASSET BKCLASS    DEP DEPDOM      EQ MUTUAL NETINC      ROA  ROAPTX ROAPTXQ
##       <dbl> <chr>    <dbl>  <dbl>   <dbl>  <dbl>  <dbl>    <dbl>   <dbl>   <dbl>
##  1    56759 N       5.05e4 5.05e4    5360      0    885   1.63     1.88     0.44
##  2   811533 SI      7.20e5 7.20e5   65486      0    458   0.0760   0.104    0.08
##  3    16458 NM      1.35e4 1.35e4    2790      0    -46  -0.528   -0.551   -1.35
##  4   380737 N       2.81e5 2.81e5   29198      0   4099   1.10     1.75     1.61
##  5   108825 SL      1.03e5 1.03e5   -9813      0 -10024 -34.9    -34.9    -34.9
##  6   538091 N       5.24e5 5.24e5    6973      0 -10899  -3.87    -3.87    -3.26
##  7    25071 SI      2.26e4 2.26e4    2461      1    -49  -0.199   -0.199   -1.22
##  8    16460 NM      1.40e4 1.40e4    2258      0     52   1.21     1.39     1.39
##  9    38118 N       3.64e4 3.64e4    1288      0    316   0.882    1.38     2.13
## 10 14667000 N       1.09e7 1.09e7 2257000      0  24000   0.638    0.877    0.88
## # i 19,014 more rows
## # i 5 more variables: ROAQ <dbl>, ROE <dbl>, STNAME <chr>, TRUST <dbl>,
## #   CommercialBank <dbl>
```

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

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

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(logit_inst)
```

```
##
```

```
## Call:
## glm(formula = CommercialBank ~ ASSET + DEP + DEPDOM + EQ + NETINC +
##     ROA + ROAPTX + ROAPTXQ + ROAQ, family = "binomial", data = inst.training)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.543e+00  2.051e-02  75.254  < 2e-16 ***
## ASSET       -1.375e-07  2.384e-08  -5.767 8.07e-09 ***
## DEP          2.239e-05  9.054e-06   2.473 0.013398 *
## DEPDOM      -2.235e-05  9.053e-06  -2.469 0.013555 *
## EQ           1.108e-06  2.070e-07   5.353 8.66e-08 ***
## NETINC       3.389e-06  9.075e-07   3.735 0.000188 ***
## ROA          1.528e-01  3.683e-02   4.148 3.35e-05 ***
## ROAPTX      -6.433e-02  3.475e-02  -1.851 0.064111 .
## ROAPTXQ      2.618e-02  2.038e-02   1.285 0.198954
## ROAQ        -3.717e-02  2.139e-02  -1.738 0.082205 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18530  on 19023  degrees of freedom
## Residual deviance: 17622  on 19014  degrees of freedom
## AIC: 17642
##
## Number of Fisher Scoring iterations: 21
```

```
library(MASS)
```

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

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

```
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-8
```

```r
X <- model.matrix(logit_inst)[, -1]
y <- as.numeric(inst.training$MUTUAL)

dim(X)
```

```
## [1] 19024     9
```

```r
length(y)
```

```
## [1] 19024
```

```r
lasso_model <- cv.glmnet(X, y, family = "binomial", alpha = 1)
print(lasso_model)
```

```
##
## Call:  cv.glmnet(x = X, y = y, family = "binomial", alpha = 1)
##
## Measure: Binomial Deviance
##
##        Lambda Index Measure       SE Nonzero
## min 0.0000706    67  0.4894 0.007860       6
## 1se 0.0005464    45  0.4968 0.007095       4
```

```r
# minimum lambda = 0.0000706. More tuning will be done next week.
```

Fitting a linear regression with boostrap

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

```
##
## Call:
## lm(formula = ASSET ~ BKCLASS + DEP + DEPDOM + EQ + NETINC + ROA +
##     ROAPTX + ROAPTXQ + ROAQ, data = inst.training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -51970762    -92792    -32041      9903 150895518
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.679e+04  3.060e+04   2.836  0.00457 **
## BKCLASSNC   -1.538e+05  2.194e+05  -0.701  0.48336
## BKCLASSNM   -9.863e+04  3.755e+04  -2.627  0.00863 **
## BKCLASSNS   -1.342e+05  4.993e+05  -0.269  0.78817
## BKCLASSSB    8.797e+04  6.358e+04   1.384  0.16651
## BKCLASSSI   -8.088e+04  8.528e+04  -0.948  0.34290
```

```
## BKCLASSSL    8.234e+04  6.102e+04   1.349  0.17723
## BKCLASSSM   -2.521e+04  5.498e+04  -0.459  0.64658
## DEP          9.062e-01  5.146e-03 176.087  < 2e-16 ***
## DEPDOM      -2.968e-02  4.296e-03  -6.909 5.05e-12 ***
## EQ           3.232e+00  2.752e-02 117.431  < 2e-16 ***
## NETINC      -2.509e+00  1.990e-01 -12.612  < 2e-16 ***
## ROA         -4.016e+03  1.569e+04  -0.256  0.79802
## ROAPTX       3.183e+03  1.489e+04   0.214  0.83071
## ROAPTXQ     -7.138e+02  1.170e+04  -0.061  0.95136
## ROAQ         2.975e+02  1.212e+04   0.025  0.98043
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2055000 on 19008 degrees of freedom
## Multiple R-squared:  0.9948, Adjusted R-squared:  0.9948
## F-statistic: 2.447e+05 on 15 and 19008 DF,  p-value: < 2.2e-16
```

```
bootstrap_lm <- function(data, formula = NULL, model = NULL, B = 1000) {
  if (!is.null(model)) {
    formula <- formula(model)
  }

  n <- nrow(data)
  coefficients <- matrix(NA, ncol = length(coef(lm(formula, data = data))), nrow = B)

  for (i in 1:B) {

    idx <- sample(1:n, replace = TRUE)
    bootstrap_sample <- data[idx, ]


    model_bootstrap <- lm(formula, data = bootstrap_sample)

    coefficients[i, ] <- coef(model_bootstrap)
  }

  return(coefficients)
}
```

```
set.seed(123)
boot_results <- bootstrap_lm(inst.training, model = lm_full, B = 1000)

head(boot_results)
```

```
##           [,1]      [,2]       [,3]       [,4]       [,5]       [,6]      [,7]
## [1,] 112034.42 -166272.5 -114672.83 -203960.14 125798.653 -100071.68  56648.02
## [2,]  66441.61 -124094.6  -79212.33  -85360.67  40059.023  -25875.58  62110.32
## [3,]  66364.89 -159237.7  -68520.06  -79400.85 157131.490  -85140.64 154313.20
## [4,]  79948.81 -160111.1 -108101.81 -103823.90 194696.748  -76469.30 271874.17
## [5,]  89028.02 -119225.6 -112188.85  -92792.16   7346.665  -75402.38  94406.71
## [6,] 118589.49 -185621.5 -116815.23 -119729.51 -67434.226 -103932.01  66994.32
##           [,8]      [,9]         [,10]    [,11]      [,12]       [,13]
## [1,] -76646.30 1.6404002 -0.8297299609 3.541487 -3.5289832 -10976.8476
```

```
## [2,] -40440.06 1.0053577 -0.0266370658 2.261186  0.5677236   -625.1104
## [3,] -59993.37 1.0154793 -0.2103640296 3.739480 -0.2918425  -4637.0613
## [4,] -66635.13 0.6108311 -0.0001167199 5.468161 -0.6489180  -5097.4688
## [5,]  44123.79 0.8980175 -0.0570679068 3.303500  2.2205626 -27710.7686
## [6,] -42165.51 0.9424168 -0.0641463442 2.899239  3.3758296  -4532.4219
##           [,14]      [,15]      [,16]
## [1,] 10053.1807 -3822.7027  4218.9479
## [2,]   762.6287  3011.3766 -4581.6168
## [3,]  1502.6516  2813.7998 -3386.1681
## [4,]  1730.2937 -5867.3997  2991.4527
## [5,] 12867.7874   559.9029  -285.7965
## [6,] -1344.2959  4324.4101 -4375.6968
```

```
# will tune next week
```

Fitting a Linear Model

```
Assets_linear <- lm(ASSET ~ BKCLASS + DEP + DEPDOM + EQ + MUTUAL + NETINC + ROA + ROAPTX + ROAPTXQ + RO
# Equity_linear <- lm(EQ ~ ASSET + BKCLASS + DEP + DEPDOM + MUTUAL + NETINC + ROA + #ROAPTX + ROAPTXQ +
# ROA_linear <- lm(ROA ~ ASSET + BKCLASS + DEP + DEPDOM + EQ + MUTUAL + NETINC + # ROAPTX + ROAPTXQ + R
# ROE_linear <- lm(ROE ~ ASSET + BKCLASS + DEP + DEPDOM + EQ + MUTUAL + NETINC + ROA + # ROAPTX + ROAPT

summary(Assets_linear)
```

```
##
## Call:
## lm(formula = ASSET ~ BKCLASS + DEP + DEPDOM + EQ + MUTUAL + NETINC +
##     ROA + ROAPTX + ROAPTXQ + ROAQ + ROE + STNAME + TRUST, data = inst.training)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -51438288   -100775    -19005     48705 150347790
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    -4.265e+04  1.195e+05  -0.357 0.721203
## BKCLASSNC                      -2.067e+05  2.223e+05  -0.930 0.352346
## BKCLASSNM                      -8.347e+04  3.893e+04  -2.144 0.032012 *
## BKCLASSNS                      -1.027e+05  5.001e+05  -0.205 0.837253
## BKCLASSSB                       9.705e+04  6.811e+04   1.425 0.154234
## BKCLASSSI                      -1.123e+05  9.612e+04  -1.169 0.242522
## BKCLASSSL                       8.689e+04  7.281e+04   1.193 0.232732
## BKCLASSSM                      -1.710e+04  5.630e+04  -0.304 0.761384
## DEP                             9.066e-01  5.169e-03 175.396  < 2e-16 ***
## DEPDOM                         -3.028e-02  4.300e-03  -7.042 1.96e-12 ***
## EQ                              3.228e+00  2.771e-02 116.498  < 2e-16 ***
## MUTUAL                          1.640e+04  7.212e+04   0.227 0.820149
## NETINC                         -2.448e+00  1.992e-01 -12.293  < 2e-16 ***
## ROA                            -5.195e+03  1.572e+04  -0.330 0.741055
## ROAPTX                          3.851e+03  1.491e+04   0.258 0.796205
## ROAPTXQ                        -3.219e+03  1.171e+04  -0.275 0.783389
```

```
## ROAQ                              2.974e+03  1.213e+04   0.245 0.806370
## ROE                               1.609e+00  9.056e+00   0.178 0.858958
## STNAMEAlaska                      4.575e+03  4.627e+05   0.010 0.992110
## STNAMEAmerican Samoa              1.031e+05  2.056e+06   0.050 0.959988
## STNAMEArizona                     3.079e+05  2.150e+05   1.432 0.152103
## STNAMEArkansas                    1.820e+02  1.692e+05   0.001 0.999142
## STNAMECalifornia                  3.601e+05  1.343e+05   2.681 0.007349 **
## STNAMEColorado                    1.451e+05  1.455e+05   0.997 0.318863
## STNAMEConnecticut                -3.919e+03  1.971e+05  -0.020 0.984133
## STNAMEDelaware                   -1.855e+05  2.687e+05  -0.691 0.489829
## STNAMEDistrict Of Columbia        3.660e+04  4.737e+05   0.077 0.938416
## STNAMEFlorida                     7.082e+04  1.338e+05   0.529 0.596634
## STNAMEGeorgia                     8.058e+04  1.403e+05   0.574 0.565654
## STNAMEGuam                       -6.844e+03  1.033e+06  -0.007 0.994713
## STNAMEHawaii                      5.301e+04  4.278e+05   0.124 0.901371
## STNAMEIdaho                       6.177e+04  3.177e+05   0.194 0.845847
## STNAMEIllinois                    4.608e+04  1.278e+05   0.361 0.718414
## STNAMEIndiana                     1.751e+03  1.513e+05   0.012 0.990766
## STNAMEIowa                        9.116e+03  1.422e+05   0.064 0.948883
## STNAMEKansas                      6.613e+04  1.413e+05   0.468 0.639786
## STNAMEKentucky                    2.749e+04  1.544e+05   0.178 0.858739
## STNAMELouisiana                   4.054e+04  1.535e+05   0.264 0.791774
## STNAMEMaine                       3.944e+04  3.008e+05   0.131 0.895670
## STNAMEMaryland                    4.498e+04  1.805e+05   0.249 0.803193
## STNAMEMassachusetts               1.536e+05  1.631e+05   0.942 0.346335
## STNAMEMichigan                    6.106e+04  1.548e+05   0.395 0.693194
## STNAMEMinnesota                   1.127e+05  1.385e+05   0.813 0.415946
## STNAMEMississippi                -2.856e+03  1.845e+05  -0.015 0.987652
## STNAMEMissouri                    7.967e+04  1.376e+05   0.579 0.562496
## STNAMEMontana                     6.060e+04  1.935e+05   0.313 0.754120
## STNAMENebraska                    7.286e+04  1.507e+05   0.484 0.628688
## STNAMENevada                      1.437e+06  2.760e+05   5.207 1.94e-07 ***
## STNAMENew Hampshire               4.177e+04  2.257e+05   0.185 0.853195
## STNAMENew Jersey                  8.334e+04  1.584e+05   0.526 0.598723
## STNAMENew Mexico                  4.071e+04  2.155e+05   0.189 0.850135
## STNAMENew York                    5.685e+05  1.547e+05   3.676 0.000238 ***
## STNAMENorth Carolina              4.611e+05  1.714e+05   2.690 0.007160 **
## STNAMENorth Dakota                9.335e+04  2.041e+05   0.457 0.647428
## STNAMEOhio                        1.063e+05  1.450e+05   0.733 0.463444
## STNAMEOklahoma                    5.805e+04  1.456e+05   0.399 0.690071
## STNAMEOregon                      1.342e+05  2.262e+05   0.593 0.552976
## STNAMEPennsylvania                2.845e+04  1.461e+05   0.195 0.845548
## STNAMEPuerto Rico                 2.216e+05  4.049e+05   0.547 0.584231
## STNAMERhode Island               -3.564e+05  3.538e+05  -1.007 0.313747
## STNAMESouth Carolina              1.318e+04  1.933e+05   0.068 0.945612
## STNAMESouth Dakota                3.173e+04  2.037e+05   0.156 0.876243
## STNAMETennessee                  -2.847e+04  1.527e+05  -0.186 0.852110
## STNAMETexas                       7.665e+04  1.238e+05   0.619 0.535897
## STNAMEUtah                       -3.199e+05  2.118e+05  -1.510 0.130981
## STNAMEVermont                     1.791e+04  4.050e+05   0.044 0.964735
## STNAMEVirgin Islands Of The U.S.  6.022e+04  1.191e+06   0.051 0.959663
## STNAMEVirginia                    2.480e+04  1.639e+05   0.151 0.879691
## STNAMEWashington                  1.123e+05  1.823e+05   0.616 0.537981
## STNAMEWest Virginia               3.106e+04  1.792e+05   0.173 0.862413
```

```
## STNAMEWisconsin                   5.969e+04  1.424e+05   0.419 0.675065
## STNAMEWyoming                     3.561e+04  2.162e+05   0.165 0.869202
## TRUST                             1.047e+05  3.674e+04   2.849 0.004385 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2052000 on 18951 degrees of freedom
## Multiple R-squared:  0.9949, Adjusted R-squared:  0.9949
## F-statistic: 5.108e+04 on 72 and 18951 DF,  p-value: < 2.2e-16
```

```
# summary(Equity_linear)
# summary(ROA_linear)
# summary(ROE_linear)
```

Fitting a Ridge Regression Model

```
asset_x <- model.matrix(ASSET ~ .-1, data = inst.training)
asset_y <- inst.training$ASSET
Assets_ridge <- glmnet(asset_x, asset_y, alpha = 0, lambda = 0.1)
print(coef(Assets_ridge))
```

```
## 75 x 1 sparse Matrix of class "dgCMatrix"
##                                         s0
## (Intercept)                   4.684592e+05
## BKCLASSN                      3.712566e+05
## BKCLASSNC                     1.566326e+05
## BKCLASSNM                     2.851969e+05
## BKCLASSNS                    -6.066966e+05
## BKCLASSSB                    -4.038826e+05
## BKCLASSSI                    -6.126480e+05
## BKCLASSSL                    -4.176063e+05
## BKCLASSSM                     3.519365e+05
## DEP                           9.180319e-01
## DEPDOM                       -3.277025e-02
## EQ                            3.151063e+00
## MUTUAL                        1.412476e+04
## NETINC                       -2.332531e+00
## ROA                          -2.869225e+03
## ROAPTX                        1.689011e+03
## ROAPTXQ                      -1.308689e+03
## ROAQ                          9.961095e+02
## ROE                           1.584246e+00
## STNAMEAlaska                 -3.259045e+03
## STNAMEAmerican Samoa          9.613922e+04
## STNAMEArizona                 2.898276e+05
## STNAMEArkansas               -9.457402e+03
## STNAMECalifornia              3.529385e+05
## STNAMEColorado                1.358343e+05
## STNAMEConnecticut            -1.284465e+04
## STNAMEDelaware               -1.321732e+05
```

```
## STNAMEDistrict Of Columbia        2.678498e+04
## STNAMEFlorida                     6.065243e+04
## STNAMEGeorgia                     7.209342e+04
## STNAMEGuam                       -1.381287e+04
## STNAMEHawaii                      4.108337e+04
## STNAMEIdaho                       5.256706e+04
## STNAMEIllinois                    3.826746e+04
## STNAMEIndiana                    -8.211161e+03
## STNAMEIowa                       -9.769831e+02
## STNAMEKansas                      5.649141e+04
## STNAMEKentucky                    1.766552e+04
## STNAMELouisiana                   3.121164e+04
## STNAMEMaine                       2.971952e+04
## STNAMEMaryland                    3.655533e+04
## STNAMEMassachusetts               1.423193e+05
## STNAMEMichigan                    5.170983e+04
## STNAMEMinnesota                   1.031595e+05
## STNAMEMississippi                -1.225844e+04
## STNAMEMissouri                    7.019244e+04
## STNAMEMontana                     5.159926e+04
## STNAMENebraska                    6.348954e+04
## STNAMENevada                      1.450075e+06
## STNAMENew Hampshire               3.293109e+04
## STNAMENew Jersey                  7.317935e+04
## STNAMENew Mexico                  3.115327e+04
## STNAMENew York                    5.607303e+05
## STNAMENorth Carolina              4.578305e+05
## STNAMENorth Dakota                8.378025e+04
## STNAMEOhio                        9.871195e+04
## STNAMEOklahoma                    4.883977e+04
## STNAMEOregon                      1.248990e+05
## STNAMEPennsylvania                1.919731e+04
## STNAMEPuerto Rico                 2.021718e+05
## STNAMERhode Island               -3.084407e+05
## STNAMESouth Carolina              3.787422e+03
## STNAMESouth Dakota                2.184557e+04
## STNAMETennessee                  -3.557836e+04
## STNAMETexas                       6.626556e+04
## STNAMEUtah                       -3.125778e+05
## STNAMEVermont                     7.899030e+03
## STNAMEVirgin Islands Of The U.S.  5.147656e+04
## STNAMEVirginia                    1.552982e+04
## STNAMEWashington                  1.033319e+05
## STNAMEWest Virginia               2.116382e+04
## STNAMEWisconsin                   5.088058e+04
## STNAMEWyoming                     2.598314e+04
## TRUST                             1.057337e+05
## CommercialBank                   -8.711973e+05
```

```
# equity_x <- model.matrix(EQ ~ .-1, data = inst.training)
# equity_y <- inst.training$EQ
# Equity_ridge <- glmnet(equity_x, equity_y, alpha = 0, lambda = 0.1)
# print(coef(Equity_ridge))
```

```
# ROA_x <- model.matrix(ROA ~ .-1, data = inst.training)
# ROA_y <- inst.training$ROA
# ROA_ridge <- glmnet(ROA_x, ROA_y, alpha = 0, lambda = 0.1)
# print(coef(ROA_ridge))

# ROE_x <- model.matrix(ROE ~ .-1, data = inst.training)
# ROE_y <- inst.training$ROE
# ROE_ridge <- glmnet(ROE_x, ROE_y, alpha = 0, lambda = 0.1)
# print(coef(ROE_ridge))
```

Fitting a model with KNN

- 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", "ROAPTX", "ROAPTXQ", "ROAQ")]
Y.train <- inst.training$ASSET

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

- Select Predictors

- 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
```

```
k <- 3
asset.knn <- knn(train=X.train, test=X.test, cl=Y.train, k)

Y.test <- as.numeric(as.character(Y.test))
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): 690649389311649"
```

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

```
## [1] "Mean Absolute Error (MAE): 583553.895522394"
```

KNN Model 2 - BKCLASS

```
# Define your mapping
bkclass.map <- c(
  "N" = 1,
  "NM" = 2,
  "OI" = 3,
  "SB" = 4,
  "SI" = 5,
  "SL" = 6,
  "SM" = 7,
  "NC" = 8,
  "NS" = 9,
  "CU" = 10
)

# Ensure inst.training$BKCLASS is a factor or character vector
# Convert to factor if necessary
inst.training$BKCLASS <- as.factor(inst.training$BKCLASS)

# Check for NA values before mapping
if (any(is.na(inst.training$BKCLASS))) {
  stop("NA values found in inst.training$BKCLASS. Please handle missing values first.")
}
```

```r
# Map BKCLASS to integers based on bkclass.map
inst.training$BKCLASS <- as.integer(bkclass.map[as.character(inst.training$BKCLASS)])

# Print table to verify the result
print(table(inst.training$BKCLASS))
```

```
## < table of extent 0 >
```

```r
# Handle NA values by assigning 0
inst.training[is.na(inst.training)] <- 0
inst.testing[is.na(inst.testing)] <- 0
```

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

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


X.train <- inst.training[, c("ASSET", "DEP", "DEPDOM", "EQ", "NETINC", "ROAPTX", "ROAPTXQ", "ROAQ")]
Y.train <- inst.training$BKCLASS


X.test <- inst.testing[, c("ASSET", "DEP", "DEPDOM", "EQ", "NETINC", "ROAPTX", "ROAPTXQ", "ROAQ")]
Y.test <- inst.testing$BKCLASS


k <- 3
bkclass.knn <- class::knn(train=X.train, test=X.test, cl=Y.train, k)


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

```
##       bkclass.knn
## Y.test    0
##      0 4757
```

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

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

Updated 6/20 to reflect comments from professor (made sure response variables are consistent and there are no unnecessary extra splits