hw4

2022-09-15

Team Cassidy, Minghao, Liam

The goal is to run knn model on the PUMA code and convert them back into district. To accomplish this, these are the requirements:
1. Featured categories in my knn model will be ownershp, housing_cost, inctot, race
2. I only consider adults and people who live in NYC. In other word, people living in Long Island/Westchester,etc. will not be considered. (A subset will be created to accomplish this)
3. Categorical fields will be normalized to reflect this. (Using norm_varb function)

To make sure PUMA is correctly implemented, I referred to the PUMA codebook and noticed a pattern: 1. PUMA code appears to be between 3701 - 4114
2. If I reduce all numbers by 3700, I get: 1-10 are Bronx, 101-110 are Manhattan, 201-203 are staten island, 301-318 are Brooklyn, 401-414 are Queens.
3. To rearrange things, I can rank these categories the same way it is being ranked as the borough, so that I get something like this: 1-10 are Bronx, 11-20 are Manhattan, 21-23 are Staten Island, 24-41 are Brooklyn, 42-55 are Queens.
4. After running knn, PUMA will be factored back into borough to see if this extra step can improve correctness of predicting boroughs.

With all of these being said, below are the codes.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
load('acs2017_ny_data.RData')
dat_NYC <- subset(acs2017_ny, (acs2017_ny$in_NYC == 1)&(acs2017_ny$AGE > 18))
dat_NYC <- dat_NYC %>% mutate(simple_puma = ifelse(PUMA > 3711,ifelse(PUMA>3811,ifelse(PUMA>3
904, ifelse(PUMA>4019, PUMA-4059,PUMA-3977),PUMA-3880),PUMA-3790),PUMA-3700))#to create the s
implified version of puma
attach(dat_NYC)
borough_f <- factor((in_Bronx + 2*in_Manhattan + 3*in_StatenI + 4*in_Brooklyn + 5*in_Queens),
levels=c(1,2,3,4,5),labels = c("Bronx","Manhattan","Staten Island","Brooklyn","Queens"))
puma_f <- as.factor(simple_puma)
norm_varb <- function(X_in) {
  (X_in - min(X_in, na.rm = TRUE))/( max(X_in, na.rm = TRUE) - min(X_in, na.rm = TRUE) )
}

is.na(OWNCOST) <- which(OWNCOST == 9999999) # that's how data codes NA values
housing_cost <- OWNCOST + RENT
norm_inc_tot <- norm_varb(INCTOT)
norm_housing_cost <- norm_varb(housing_cost)
norm_ownership <- norm_varb(OWNERSHP)
norm_race <- norm_varb(RACE)

data_use_prelim <- cbind(norm_inc_tot,norm_housing_cost,norm_race,norm_ownership)
data_use_prelim <- data.frame(data_use_prelim)
```

```
good_obs_data_use <- complete.cases(data_use_prelim,puma_f)
dat_use <- subset(data_use_prelim,good_obs_data_use)
y_use <- subset(puma_f,good_obs_data_use)
#For borough info as well
borough1 <- complete.cases(data_use_prelim,borough_f)
borough_use <- subset(data_use_prelim,borough1)
actual_borough <- subset(borough_f,borough1)
set.seed(35)
NN_obs <- sum(good_obs_data_use == 1)
select1 <- (runif(NN_obs) < 0.8)
train_data <- subset(dat_use,select1)
test_data <- subset(dat_use,(!select1))
cl_data <- y_use[select1]
true_data <- actual_borough[!select1]
```

```
summary(cl_data)
```

```
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
##  599  676  495  675  759  489  619  544  781  663  817  581  609  521  780  640
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31   32
##  711  549  731  756  827  708 1014  829  680  678  853  741  725  580  777 1518
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47   48
##  793  535  797  975  907  885 1196 1383  612 1197  824 1489  896 1345  860  624
##   49   50   51   52   53   54   55
##  682  768 1306 1101 1547  825  549
```

```
prop.table(summary(cl_data))
```

```
##          1          2          3          4          5          6          7
## 0.01330490 0.01501522 0.01099487 0.01499300 0.01685880 0.01086160 0.01374914
##          8          9         10         11         12         13         14
## 0.01208325 0.01734746 0.01472646 0.01814709 0.01290509 0.01352702 0.01157238
##         15         16         17         18         19         20         21
## 0.01732525 0.01421559 0.01579263 0.01219431 0.01623687 0.01679216 0.01836921
##         22         23         24         25         26         27         28
## 0.01572599 0.02252282 0.01841363 0.01510406 0.01505964 0.01894671 0.01645899
##         29         30         31         32         33         34         35
## 0.01610360 0.01288288 0.01725861 0.03371760 0.01761400 0.01188334 0.01770285
##         36         37         38         39         40         41         42
## 0.02165656 0.02014615 0.01965749 0.02656538 0.03071900 0.01359366 0.02658759
##         43         44         45         46         47         48         49
## 0.01830257 0.03307345 0.01990182 0.02987495 0.01910220 0.01386020 0.01514849
##         50         51         52         53         54         55
## 0.01705871 0.02900868 0.02445525 0.03436174 0.01832478 0.01219431
```

```
summary(train_data)
```

```
##   norm_inc_tot      norm_housing_cost   norm_race      norm_ownership
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.009871   1st Qu.:0.02139   1st Qu.:0.0000   1st Qu.:0.5000
##  Median :0.020569   Median :0.96532   Median :0.1250   Median :1.0000
##  Mean   :0.034887   Mean   :0.56502   Mean   :0.2168   Mean   :0.7389
##  3rd Qu.:0.042858   3rd Qu.:0.97688   3rd Qu.:0.3750   3rd Qu.:1.0000
##  Max.   :1.000000   Max.   :1.00000   Max.   :1.0000   Max.   :1.0000
```

```
require(class)
```

```
## Loading required package: class
```

```
m = c(11:20)
m[1] = '11'
s = c(21:23)
s[1] = '21'
b = c(24:41)
b[1] = '24'
q = c(42:55)
q[1] = '42'
for (indx in seq(1, 9, by= 2)) {
  pred_PUMA <- knn(train_data, test_data, cl_data, k = indx, l = 0, prob = FALSE, use.all = T
RUE)
  x <- as.numeric(pred_PUMA)
  x [x<11] = 'Bronx'
  x [x %in% m] = 'Manhattan'
  x [x %in% s] = 'Staten Island'
  x [x %in% b] = 'Brooklyn'
  x [x %in% q] = 'Queens'
  num_correct_labels <- sum(x == true_data)
  correct_rate <- num_correct_labels/length(true_data)
  print(c(indx,correct_rate))
}
```

```
## [1] 1.0000000 0.3709677
## [1] 3.0000000 0.3607353
## [1] 5.0000000 0.3711412
## [1] 7.000000 0.377905
## [1] 9.0000000 0.3783385
```

Comparing this to just using borough info:

```
set.seed(35)
good_obs_data_use <- complete.cases(data_use_prelim,borough_f)
dat_use <- subset(data_use_prelim,good_obs_data_use)
y_use <- subset(borough_f,good_obs_data_use)
NN_obs <- sum(good_obs_data_use == 1)
select1 <- (runif(NN_obs) < 0.8)
train_data <- subset(dat_use,select1)
test_data <- subset(dat_use,(!select1))
cl_data <- y_use[select1]
true_data <- y_use[!select1]
for (indx in seq(1, 9, by= 2)) {
  pred_borough <- knn(train_data, test_data, cl_data, k = indx, l = 0, prob = FALSE, use.all
= TRUE)
  num_correct_labels <- sum(pred_borough == true_data)
  correct_rate <- num_correct_labels/length(true_data)
  print(c(indx,correct_rate))
}
```

```
## [1] 1.0000000 0.3769511
## [1] 3.0000000 0.3844953
## [1] 5.0000000 0.3991502
## [1] 7.0000000 0.4094693
## [1] 9.0000000 0.4190947
```

It is interesting that, using a very precise PUMA code actually yielded a much worse results than just using boroughs, and improvement is also marginal when using higher index of knn.

A better result could potentially be achieved by using more factors, but I also noticed a strong correlation between factors being used and a potential of overfitting based on the trend of the entire survey. Plus, 40% accuracy is pretty good considering I am predicting a person out of 5 boroughs and the theorhetical “random guess” probability is only 20% :)

Out of curiosity, I will also show this against ols:

```
cl_data_n <- as.numeric(cl_data)

model_ols1 <- lm(cl_data_n ~ train_data$norm_inc_tot + train_data$norm_housing_cost + train_d
ata$norm_race + train_data$norm_ownership)

y_hat <- fitted.values(model_ols1)

mean(y_hat[cl_data_n == 1])
```

```
## [1] 3.451404
```

```
mean(y_hat[cl_data_n == 2])
```

```
## [1] 3.345856
```

```
mean(y_hat[cl_data_n == 3])
```

```
## [1] 3.742035
```

```
mean(y_hat[cl_data_n == 4])
```

```
## [1] 3.542451
```

```
mean(y_hat[cl_data_n == 5])
```

```
## [1] 3.62483
```

```
# maybe try classifying one at a time with OLS

cl_data_n1 <- as.numeric(cl_data_n == 1)
model_ols_v1 <- lm(cl_data_n1 ~ train_data$norm_inc_tot + train_data$norm_housing_cost+train_
data$norm_race + train_data$norm_ownership)
y_hat_v1 <- fitted.values(model_ols_v1)
mean(y_hat_v1[cl_data_n1 == 1])
```

```
## [1] 0.1715303
```

```
mean(y_hat_v1[cl_data_n1 == 0])
```

```
## [1] 0.134794
```

And OLS performs even worse than randomly guessing.