hw10.R

zhuoxun.yang001

2022-11-03

```
# import required packages
library(DAAG)
library(ggplot2)
library(mice)
library(naniar)
library(VIM)
library(kknn)
library(kernlab)

#import data & set seed
df <- read.csv("C:/Users/zhuoxun.yang001/Downloads/hw10-SP22/data 14.1/breast-cancer-wisconsin.data.txt", header=
FALSE)
# set seed
set.seed(9876)
# check the head
head(df)
```

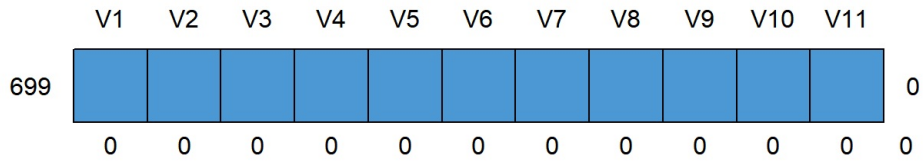
```
##        V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 1 1000025  5  1  1  1  2  1  3  1   1   2
## 2 1002945  5  4  4  5  7 10  3  2   1   2
## 3 1015425  3  1  1  1  2  2  3  1   1   2
## 4 1016277  6  8  8  1  3  4  3  7   1   2
## 5 1017023  4  1  1  3  2  1  3  1   1   2
## 6 1017122  8 10 10  8  7 10  9  7   1   4
```

```
# check the structure
str(df)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 ...
##  $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
##  $ V7 : chr  "1" "10" "2" "4" ...
##  $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
##  $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
```

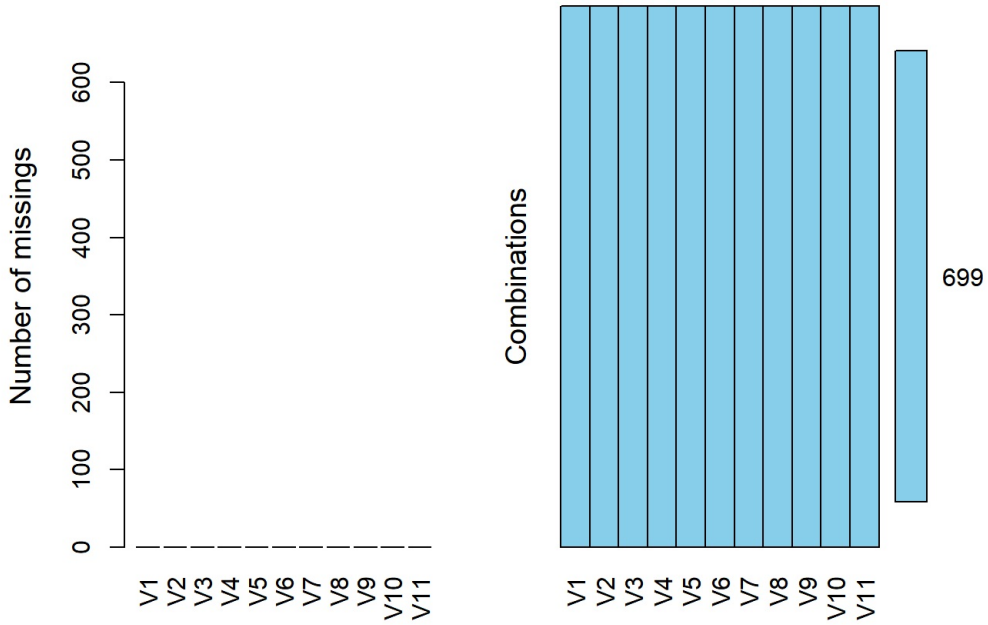
```
# check the null values
md.pattern(df) # using mice package to check the nas
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```




```
##     V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11
## 699  1  1  1  1  1  1  1  1  1   1   1 0
##      0  0  0  0  0  0  0  0  0   0   0 0
```

```
# # visualize nas,
aggr(df, prop = F, numbers = T) # we can find out there are no missing values
```




```
                          # due to left corner plot. However, we need to use
                          # another way to check it out to be more confident.
```

```
# check unique values in column v2
unique(df$V2)
```

```
##  [1]  5  3  6  4  8  1  2  7 10  9
```

```
#
unique(df$V3)
```

```
##  [1]  1  4  8 10  2  3  7  5  6  9
```

```
#
unique(df$V4)
```

```
##  [1]  1  4  8 10  2  3  5  6  7  9
```

```
#
unique(df$V5)
```

```
##  [1]  1  5  3  8 10  4  6  2  9  7
```

```
#
unique(df$V6)
```

```
##  [1]  2  7  3  1  6  4  5  8 10  9
```

```
# abnormal value
unique(df$V7) # we can find a question mark in column v7
```

```
##  [1] "1"  "10" "2"  "4"  "3"  "9"  "7"  "?"  "5"  "8"  "6"
```

```
#
unique(df$V8)
```

```
##  [1]  3  9  1  2  4  5  7  8  6 10
```

```
#
unique(df$V9)
```

```
##  [1]  1  2  7  4  5  3 10  6  9  8
```

```
#
unique(df$V10)
```

```
## [1]  1  5  4  2  3  7 10  8  6
```

```
#
unique(df$V11)
```

```
## [1] 2 4
```

```
# mice packages can find NAs but not abnormal symbols.
```

```
# print values of each columns, more straightforward.
for (i in 2:11) {
  print(table(df[,i]))
  print(paste0("V",i))
}
```

```
##
##   1   2   3   4   5   6   7   8   9  10
## 145  50 108  80 130  34  23  46  14  69
## [1] "V2"
##
##   1   2   3   4   5   6   7   8   9  10
## 384  45  52  40  30  27  19  29   6  67
## [1] "V3"
##
##   1   2   3   4   5   6   7   8   9  10
## 353  59  56  44  34  30  30  28   7  58
## [1] "V4"
##
##   1   2   3   4   5   6   7   8   9  10
## 407  58  58  33  23  22  13  25   5  55
## [1] "V5"
##
##   1   2   3   4   5   6   7   8   9  10
##  47 386  72  48  39  41  12  21   2  31
## [1] "V6"
##
##   ?   1  10   2   3   4   5   6   7   8   9
##  16 402 132  30  28  19  30   4   8  21   9
## [1] "V7"
##
##   1   2   3   4   5   6   7   8   9  10
## 152 166 165  40  34  10  73  28  11  20
## [1] "V8"
##
##   1   2   3   4   5   6   7   8   9  10
## 443  36  44  18  19  22  16  24  16  61
## [1] "V9"
##
##   1   2   3   4   5   6   7   8  10
## 579  35  33  12   6   3   9   8  14
## [1] "V10"
##
##   2   4
## 458 241
## [1] "V11"
```

```
# check the percentage to see if it is match the thumbs of rule
nrow(df[which(df$V7 == "?"),])/nrow(df) # can use imputation because < 5%
```

```
## [1] 0.02288984
```

```
# calculate mean and set column to numeric
v7_mean <- mean(as.numeric(df[-(which(df$V7 == "?", arr.ind = TRUE)),"V7"]))
v7_mean # print mean
```

```
## [1] 3.544656
```

```
index <- which(df$V7 == "?", arr.ind = TRUE) # set index of "?"
df_imp_mean <- df # replicate df
df_imp_mean[index,]$V7 <- v7_mean # mean imputation in df$v7
unique(df_imp_mean$V7) # check if imputation works
```

```
##  [1] "1"               "10"              "2"               "4"               "3"               "9"
##  [7] "7"               "3.54465592972182" "5"               "8"               "6"
```

```
# excluding response variable
df_ <- df[,-11]# if we didn't do this, the output will pop out that nas introduced by coercion
df_ind <- df_[-index,]
dim(df_ind) # check the dimension
```

```
## [1] 683  10
```

```
df_ind$V7 <- as.integer(df_ind$V7) # set v7 to be integer otherwise output will
# reach warning message that >1 explanatory variable.

# build up linear regression
lm <- lm(V7~V2+V3+V4+V5+V6+V8+V9+V10, data = df_ind)
summary(lm) # get summary
```

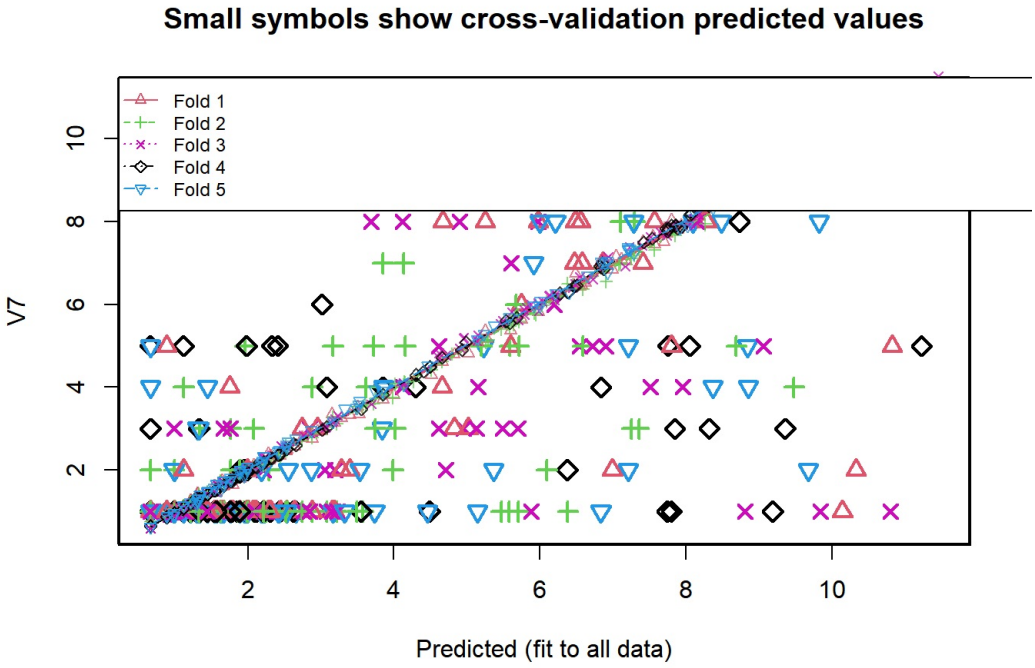
```
##
## Call:
## lm(formula = V7 ~ V2 + V3 + V4 + V5 + V6 + V8 + V9 + V10, data = df_ind)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7316 -0.9426 -0.3002  0.6725  8.6998
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.616652   0.194975  -3.163  0.00163 **
## V2           0.230156   0.041691   5.521 4.83e-08 ***
## V3          -0.067980   0.076170  -0.892  0.37246
## V4           0.340442   0.073420   4.637 4.25e-06 ***
## V5           0.339705   0.045919   7.398 4.13e-13 ***
## V6           0.090392   0.062541   1.445  0.14883
## V8           0.320577   0.059047   5.429 7.91e-08 ***
## V9           0.007293   0.044486   0.164  0.86983
## V10         -0.075230   0.059331  -1.268  0.20524
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.274 on 674 degrees of freedom
## Multiple R-squared:  0.615,  Adjusted R-squared:  0.6104
## F-statistic: 134.6 on 8 and 674 DF,  p-value: < 2.2e-16
```

```
# rebuild model due to statistically significance
lm_revised <- lm(V7~V2+V4+V5+V8, data = df_ind) # build model with only significance
# variable
# get summary
summary(lm_revised)
```

```
##
## Call:
## lm(formula = V7 ~ V2 + V4 + V5 + V8, data = df_ind)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.8115 -0.9531 -0.3111  0.6678  8.6889
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.53601    0.17514  -3.060   0.0023 **
## V2           0.22617    0.04121   5.488 5.75e-08 ***
## V4           0.31729    0.05086   6.239 7.76e-10 ***
## V5           0.33227    0.04431   7.499 2.03e-13 ***
## V8           0.32378    0.05606   5.775 1.17e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.274 on 678 degrees of freedom
## Multiple R-squared:  0.6129, Adjusted R-squared:  0.6107
## F-statistic: 268.4 on 4 and 678 DF,  p-value: < 2.2e-16
```

```
# using cross_validation to check the fitness of model
cv_regression <- cv.lm(df_ind, lm_revised, m=5)
```

```
## Warning in cv.lm(df_ind, lm_revised, m = 5):
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
##
## fold 1
## Observations in test set: 136
##                      4         6        12        21        22        29        36        37        51
55         56
## Predicted    4.6629181 10.018398  1.213452  6.623331 6.5754051  1.213452  1.213452 10.146951  5.017641 7.57220
28  5.5975534
## cvpred       4.6191932 10.058192  1.255391  6.517183 6.4654585  1.255391  1.255391 10.074662  4.821406 7.44690
69  5.3690859
## V7           4.0000000 10.000000  1.000000 10.000000 7.0000000  1.000000  1.000000  1.000000  3.000000 8.00000
00  5.0000000
## CV residual -0.6191932 -0.058192 -0.255391  3.482817 0.5345415 -0.255391 -0.255391 -9.074662 -1.821406 0.55309
31 -0.3690859
##                     57        59        64         67        74        80        81         82       87
89         91
## Predicted   5.7498291  3.497886  3.393772  1.9895750  7.703744  1.213452  3.149661  1.9980563 4.670420  1.9895
750  1.3110677
## cvpred      5.6380545  3.525040  3.324441  1.9818912  7.504717  1.255391  3.354155  1.9540254 4.751053  1.9818
912  1.4246636
## V7          6.0000000 10.000000  2.000000  1.0000000 10.000000  1.000000  1.000000  1.0000000 8.000000  1.0000
000  1.0000000
## CV residual 0.3619455  6.474960 -1.324441 -0.9818912  2.495283 -0.255391 -2.354155 -0.9540254 3.248947 -0.9818
912 -0.4246636
##                    104         109       118       123        125       127       129       130       131
138       145
## Predicted    4.823962  0.98728319  7.544252  7.182002  7.4146942  6.591389  4.512626 0.6634986  2.526532  1.11
583682  1.213452
## cvpred       4.792298  1.06964846  7.641050  7.046719  7.5240472  6.569453  4.291032 0.7146333  2.459995  1.08
611828  1.255391
## V7           3.000000  1.00000000 10.000000 10.000000  7.0000000 10.000000 10.000000 1.0000000  1.000000  1.00
000000  1.000000
## CV residual -1.792298 -0.06964846  2.358950  2.953281 -0.5240472  3.430547  5.708968 0.2853667 -1.459995 -0.08
611828 -0.255391
##                    156       160        166        169         173        179       187        194        195
197        200
## Predicted    5.146219  7.935668 1.98957497  1.7634059  0.98728319  1.9895750 6.488540  1.3110677  1.7634059 6.
4703123  1.4396214
## cvpred       5.098134  7.898157 1.98189115  1.7961486  1.06964846  1.9818912 6.746289  1.4246636  1.7961486 6.
4245843  1.4411335
## V7          10.000000 10.000000 2.00000000  1.0000000  1.00000000  1.0000000 8.000000  1.0000000  1.0000000 7.
0000000  1.0000000
## CV residual  4.901866  2.101843 0.01810885 -0.7961486 -0.06964846 -0.9818912 1.253711 -0.4246636 -0.7961486 0.
5754157 -0.4411335
##                    202       210       215        221         226       231       238       248       255
259       272
## Predicted    8.183315  2.215744 11.459052  1.6433336  0.98728319 6.8637366  6.994273 3.846111 5.975998  1.7634
059  2.215744
## cvpred       8.118688  2.167634 11.438991  1.7518131  1.06964846 6.8827567  6.840034 3.695926 5.823797  1.7961
486  2.167634
## V7          10.000000  1.000000 10.000000  1.0000000  1.00000000 7.0000000  2.000000 9.000000 8.000000  1.0000
000  1.000000
```

```
## CV residual  1.881312 -1.167634 -1.438991 -0.7518131 -0.06964846 0.1172433 -4.840034 5.304074 2.176203 -0.7961
486 -1.167634
##                    275        281        302        303        304        318       324       327      331
341      350
## Predicted    1.7634059  1.7634059  1.3110677  9.4909008  1.3110677  8.2904156  7.116354  3.994159 6.553928  5.
276300 5.248350
## cvpred       1.7961486  1.7961486  1.4246636  9.3924971  1.4246636  8.3606274  7.085435  3.806377 6.430670  5.
163783 5.357926
## V7           1.0000000  1.0000000  1.0000000 10.0000000  1.0000000  8.0000000 10.000000 10.000000 8.000000 10.
000000 8.000000
## CV residual -0.7961486 -0.7961486 -0.4246636  0.6075029 -0.4246636 -0.3606274  2.914565  6.193623 1.569330  4.
836217 2.642074
##                    354        358       364       376         377         378       381       388        397
403         408
## Predicted    7.706992  9.0385627 2.9499156 0.6634986  0.98728319  0.98728319 0.6634986  3.182583  1.7634059  2
.526532  0.98728319
## cvpred       7.822960  9.0210121 2.9250899 0.7146333  1.06964846  1.06964846 0.7146333  3.142159  1.7961486  2
.459995  1.06964846
## V7          10.000000 10.0000000 3.0000000 1.0000000  1.00000000  1.00000000 1.0000000  1.000000  1.0000000  1
.000000  1.00000000
## CV residual  2.177040  0.9789879 0.0749101 0.2853667 -0.06964846 -0.06964846 0.2853667 -2.142159 -0.7961486 -1
.459995 -0.06964846
##                    414       417         418       421       426         429       439       447        455
459        473
## Predicted    2.533030  7.739458  0.98728319 2.7452241 11.232883  0.98728319  2.641111 0.6634986 0.88966773  1.
8854614  1.7943441
## cvpred       2.491322  7.719337  1.06964846 2.7741358 11.253249  1.06964846  2.573536 0.7146333 0.90037577  1.
7812914  1.6433458
## V7           1.000000 10.000000  1.00000000 3.0000000 10.000000  1.00000000  1.000000 1.0000000 1.00000000  1.
0000000  1.0000000
## CV residual -1.491322  2.280663 -0.06964846 0.2258642 -1.253249 -0.06964846 -1.573536 0.2853667 0.09962423 -0.
7812914 -0.6433458
##                    477       482       492        498       500       501       502       511       524
528        533
## Predicted    1.6592923  2.882259  7.045971  1.3420059  1.665790  2.441913  1.665790 0.6634986  7.892476  1.989
5750  1.3110677
## cvpred       1.5955489  2.762740  7.158648  1.2718608  1.626876  2.353376  1.626876 0.7146333  7.811935  1.981
8912  1.4246636
## V7           1.0000000  1.000000 10.000000  1.0000000  1.000000  1.000000  1.000000 1.0000000 10.000000  1.000
0000  1.0000000
## CV residual -0.5955489 -1.762740  2.841352 -0.2718608 -0.626876 -1.353376 -0.626876 0.2853667  2.188065 -0.981
8912 -0.4246636
##                    536       537       539        543       544        551        552       556        563
566       567
## Predicted    2.277906  2.215744  1.665790  1.5681750  1.665790  1.4396214  1.3110677  2.313360  1.3110677 10.3
282067  2.080692
## cvpred       2.399189  2.167634  1.626876  1.4576033  1.626876  1.4411335  1.4246636  2.336906  1.4246636 10.5
102786  2.119837
## V7           1.000000  1.000000  1.000000  1.0000000  1.000000  1.0000000  1.0000000  1.000000  1.0000000 10.0
000000  1.000000
## CV residual -1.399189 -1.167634 -0.626876 -0.4576033 -0.626876 -0.4411335 -0.4246636 -1.336906 -0.4246636 -0.5
102786 -1.119837
##                    569       570       571        580        590        597       616        617      626
628       636
## Predicted    3.522326 10.824479  6.461831  1.3110677  1.5681750  1.9830769  2.300363  1.4396214 1.750410 0.889
6677  2.067696
## cvpred       3.340910 10.791615  6.452450  1.4246636  1.4576033  1.9505641  2.274252  1.4411335 1.733495 0.900
3758  2.057183
## V7          10.000000  5.000000 10.000000  1.0000000  1.0000000  1.0000000  1.000000  1.0000000 4.000000 5.000
0000  1.000000
## CV residual  6.659090 -5.791615  3.547550 -0.4246636 -0.4576033 -0.9505641 -1.274252 -0.4411335 2.266505 4.099
6242 -1.057183
##                    638        641        642       649        663       665        673       676        677
679       682
## Predicted    3.282182  2.0065377  1.4396214 10.328207  1.6218560  2.104153  1.5372368  2.293865  1.3045696 0.6
634986  8.709284
## cvpred       3.252239  1.9261597  1.4411335 10.510279  1.7170248  2.095432  1.6104061  2.242925  1.3933366 0.7
146333  8.735203
## V7           2.000000  1.0000000  1.0000000  2.000000  1.0000000  1.000000  1.0000000  1.000000  1.0000000 1.0
000000 10.000000
## CV residual -1.252239 -0.9261597 -0.4411335 -8.510279 -0.7170248 -1.095432 -0.6104061 -1.242925 -0.3933366 0.2
853667  1.264797
##                    688       695        696       699
## Predicted    1.4396214 1.1158368 0.88966773  7.806135
## cvpred       1.4411335 1.0861183 0.90037577  8.041412
## V7           1.0000000 2.0000000 1.00000000  5.000000
## CV residual -0.4411335 0.9138817 0.09962423 -3.041412
##
## Sum of squares = 675.12    Mean square = 4.96    n = 136
```

```
##
## fold 2
## Observations in test set: 137
##                     3        16         17       26         27       40        53       54        62        66
73        77
## Predicted   1.7634059  5.575097  1.6657904 3.847115  1.4396214 4.131480  5.589072 7.105889 0.9872832  3.987660
3.573045  1.939142
## cvpred      1.7311017  5.539097  1.6353391 3.810260  1.4395826 4.058919  5.466072 6.994128 1.0480697  3.741518
3.533016  2.093875
## V7          2.0000000  1.000000  1.0000000 7.000000  1.0000000 7.000000  5.000000 8.000000 2.0000000  2.000000
1.000000  1.000000
## CV residual 0.2688983 -4.539097 -0.6353391 3.189740 -0.4395826 2.941081 -0.466072 1.005872 0.9519303 -1.741518
-2.533016 -1.093875
##                    79        83       85         92         93        99      102       111      112       115
120        139
## Predicted   1.763406  2.215744 7.288124  1.4481027  1.9895750 5.6671932 3.162109  2.290903 4.510642 2.0806923
1.7634059  1.9830769
## cvpred      1.731102  2.122615 7.299818  1.4782107  1.9268582 5.7330826 3.280073  2.252774 4.415984 2.0797034
1.7311017  1.9839408
## V7          3.000000  1.000000 9.000000  1.0000000  1.0000000 6.0000000 5.000000  2.000000 9.000000 3.0000000
2.0000000  1.0000000
## CV residual 1.268898 -1.122615 1.700182 -0.4782107 -0.9268582 0.2669174 1.719927 -0.252774 4.584016 0.9202966
0.2688983 -0.9839408
##                     141       143       150       152      154        172       176       178       182        1
83       198
## Predicted    1.1158368  5.710148  7.089906  4.267511 1.342006  1.3110677  6.760627  6.383423 0.6634986  2.4419
13  3.212542
## cvpred       1.1480635  5.582008  6.958012  4.142971 1.343820  1.3395887  6.778146  6.503363 0.7565506  2.3183
71  3.113056
## V7           1.0000000  5.000000 10.000000 10.000000 3.000000  1.0000000 10.000000  1.000000 1.0000000  1.0000
00  1.000000
## CV residual -0.1480635 -0.582008  3.041988  5.857029 1.656180 -0.3395887  3.221854 -5.503363 0.2434494 -1.3183
71 -2.113056
##                    199       204        208        211        218       220       222       225       232
233       235
## Predicted   0.6634986  2.215744  1.3110677 10.3591449  1.3110677  3.076486  6.905688  7.572203 7.2936180  5.46
9000  2.080692
## cvpred      0.7565506  2.122615  1.3395887 10.2762193  1.3395887  3.015575  6.556611  7.312616 7.1881655  5.40
4706  2.079703
## V7          1.0000000  1.000000  1.0000000 10.0000000  1.0000000  1.000000 10.000000 10.000000 8.0000000  1.00
0000  1.000000
## CV residual 0.2434494 -1.122615 -0.3395887 -0.2762193 -0.3395887 -2.015575  3.443389  2.687384 0.8118345 -4.40
4706 -1.079703
##                    237       242        243      244       254        261       265       267       268       2
71       274
## Predicted    6.223645  2.427938  1.5372368 1.958637  6.924895  9.2055566  7.257904  5.921338  4.659956  4.7979
70 3.6199415
## cvpred       6.163275  2.391396  1.5353452 1.922627  6.952010  9.0959715  7.114295  5.796219  4.621732  4.8014
95 3.5599328
## V7          10.000000  1.000000  1.0000000 5.000000 10.000000 10.0000000  3.000000 10.000000 10.000000 10.0000
00 4.0000000
## CV residual  3.836725 -1.391396 -0.5353452 3.077373  3.047990  0.9040285 -4.114295  4.203781  5.378268  5.1985
05 0.4400672
##                    279        282        287        288      289       290       291        292       294
297       299
## Predicted    1.3110677  1.8695027  9.5163449  1.4396214 3.724055  6.451822 0.6634986  1.3110677  5.635994 4.15
59197  2.246682
## cvpred       1.3395887  1.8654923  9.5016559  1.4395826 3.598613  6.468172 0.7565506  1.3395887  5.506419 4.12
02332  2.126846
## V7           1.0000000  1.0000000 10.0000000  1.0000000 5.000000 10.000000 1.0000000  1.0000000 10.000000 5.00
00000  1.000000
## CV residual -0.3395887 -0.8654923  0.4983441 -0.4395826 1.401387  3.531828 0.2434494 -0.3395887  4.493581 0.87
97668 -1.126846
##                    306       312       317        319        323         328       333       344       349
351       353
## Predicted    5.295246 0.6634986  4.140940  1.3110677  1.7634059  0.98728319  2.541512 0.6634986  5.705658  2.2
32707  4.020868
## cvpred       5.137644 0.7565506  4.138688  1.3395887  1.7311017  1.04806967  2.509844 0.7565506  5.748231  2.1
99871  4.077322
## V7          10.000000 1.0000000 10.000000  1.0000000  1.0000000  1.00000000  1.000000 1.0000000  1.000000  1.0
00000  3.000000
## CV residual  4.862356 0.2434494  5.861312 -0.3395887 -0.7311017 -0.04806967 -1.509844 0.2434494 -4.748231 -1.1
99871 -1.077322
##                    362        366       368        371       374        395       398       404       413
416       428
## Predicted    6.035941  1.2134523  9.046065  1.9830769  2.224225  1.6218560  1.342006 1.115837  9.482419  3.742
022  6.095091
## cvpred       5.982667  1.2438261  8.991746  1.9839408  2.161243  1.7452732  1.343820 1.148064  9.347144  3.730
439  6.063282
```

```
## V7          10.000000  1.0000000 10.000000  1.0000000  1.000000  1.0000000  1.000000 4.000000  4.000000  3.000
000  2.000000
## CV residual  4.017333 -0.2438261  1.008254 -0.9839408 -1.161243 -0.7452732 -0.343820 2.851936 -5.347144 -0.730
439 -4.063282
##                   432        433       441      442       444       450        452       454        471
475        481
## Predicted    2.880276  1.8919595  9.238047 2.882259 0.6634986  8.698819  1.5681750  7.851074  1.4396214  1.568
1750  1.5681750
## cvpred       2.782909  1.8310956  8.810558 2.878620 0.7565506  8.680054  1.5395765  7.672929  1.4395826  1.539
5765  1.5395765
## V7           1.000000  1.0000000 10.000000 4.000000 2.0000000 10.000000  1.0000000 10.000000  1.0000000  1.000
0000  1.0000000
## CV residual -1.782909 -0.8310956  1.189442 1.121380 1.2434494  1.319946 -0.5395765  2.327071 -0.4395826 -0.539
5765 -0.5395765
##                   495       505         518        519        525       531       541        549       550
553       557
## Predicted    5.1996390 0.6634986  0.98728319  1.7653891  1.4396214  5.255827 1.8919595  1.1158368  6.591389  2
.736743  2.541512
## cvpred       5.1510214 0.7565506  1.04806967  1.8268124  1.4395826  5.094784 1.8310956  1.1480635  6.358290  2
.701370  2.509844
## V7           5.0000000 1.0000000  1.00000000  1.0000000  1.0000000 10.000000 2.0000000  1.0000000  5.000000  1
.000000  1.000000
## CV residual -0.1510214 0.2434494 -0.04806967 -0.8268124 -0.4395826  4.905216 0.1689044 -0.1480635 -1.358290 -1
.701370 -1.509844
##                   558        560        564         574        577         579       593       600        601
607       611
## Predicted    2.232707  1.8919595  1.4396214  0.98728319  1.8919595  0.98728319  5.951297  2.520034  1.4396214
1.6742718  8.266694
## cvpred       2.199871  1.8310956  1.4395826  1.04806967  1.8310956  1.04806967  5.759310  2.585382  1.4395826
1.6739671  7.936001
## V7           1.000000  1.0000000  1.0000000  1.00000000  1.0000000  1.00000000 10.000000  1.000000  1.0000000
1.0000000 10.000000
## CV residual -1.199871 -0.8310956 -0.4395826 -0.04806967 -0.8310956 -0.04806967  4.240690 -1.585382 -0.4395826
-0.6739671  2.063999
##                   615       624       633       640       644       658        662        664       666
670       683
## Predicted    1.2134523 0.6634986 0.6634986  2.232707 0.6634986  3.484890  1.9895750  1.6218560 0.6634986  8.69
2321  2.215744
## cvpred       1.2438261 0.7565506 0.7565506  2.199871 0.7565506  3.517022  1.9268582  1.7452732 0.7565506  8.73
7137  2.122615
## V7           1.0000000 1.0000000 1.0000000  1.000000 1.0000000  1.000000  1.0000000  1.0000000 1.0000000  5.00
0000  1.000000
## CV residual -0.2438261 0.2434494 0.2434494 -1.199871 0.2434494 -2.517022 -0.9268582 -0.7452732 0.2434494 -3.73
7137 -1.122615
##                   685        691       697
## Predicted   0.6634986  1.3280304  7.354776
## cvpred      0.7565506  1.4168448  7.377920
## V7          1.0000000  1.0000000  3.000000
## CV residual 0.2434494 -0.4168448 -4.377920
##
## Sum of squares = 671.95    Mean square = 4.9    n = 137
##
## fold 3
## Observations in test set: 137
##                     7         10         11       15         23        30         32        42        45
46       50
## Predicted    1.311068  1.6657904  1.3110677 7.801359  1.4396214  1.298072  1.5372368  4.952516  8.817888 0.987
2832 4.904067
## cvpred       1.210609  1.7024006  1.2106087 7.903720  1.4329981  1.171744  1.4800112  5.178445  8.885669 0.894
1930 4.940760
## V7          10.000000  1.0000000  1.0000000 9.000000  1.0000000  1.000000  1.0000000  3.000000  1.000000 1.000
0000 8.000000
## CV residual  8.789391 -0.7024006 -0.2106087 1.096280 -0.4329981 -0.171744 -0.4800112 -2.178445 -7.885669 0.105
8070 3.059240
##                   61       63        65       69        71        78        84          86        88        9
4         97
## Predicted    5.137738 5.975998 0.9872832 7.398711  2.526532  2.224225  3.058544  4.12596072  5.982521 0.987283
2  1.2219336
## cvpred       5.119914 6.038679 0.8941930 7.421061  2.565770  2.303640  3.015076  4.08955102  5.842760 0.894193
0  1.1790165
## V7           3.000000 8.000000 1.0000000 9.000000  1.000000  1.000000  2.000000  4.00000000 10.000000 1.000000
0  1.0000000
## CV residual -2.119914 1.961321 0.1058070 1.578939 -1.565770 -1.303640 -1.015076 -0.08955102  4.157240 0.105807
0 -0.1790165
##                   101       105       106       108       124        136      147       149        151       1
62       164
## Predicted   4.6157352 10.811483  4.616739  7.170035  4.132459  2.2157441 3.688602  3.075507  1.3110677  1.9895
75 0.9957645
## cvpred      4.8231646 10.876689  4.713484  6.908853  4.108983  2.2882188 3.585599  3.045918  1.2106087  2.0188
```

```
16 0.9096140
## V7          5.0000000  1.000000  3.000000 10.000000 10.000000  2.0000000 8.000000  1.000000  1.0000000  1.0000
00 3.0000000
## CV residual 0.1768354 -9.876689 -1.713484  3.091147  5.891017 -0.2882188 4.414401 -2.045918 -0.2106087 -1.0188
16 2.0903860
##                   167       175       184       185        186       201        203       207       219
240       241
## Predicted    6.445324  6.131548  8.057748  6.125050  1.5372368  7.648341  1.3110677  6.549413  7.958150  4.960
997  3.191064
## cvpred       6.392975  6.229116  7.955621  6.209684  1.4800112  7.740738  1.2106087  6.674771  7.912619  5.193
866  3.229443
## V7          10.000000 10.000000 10.000000 10.000000  1.0000000 10.000000  1.0000000  5.000000  4.000000 10.000
000  2.000000
## CV residual  3.607025  3.770884  2.044379  3.790316 -0.4800112  2.259262 -0.2106087 -1.674771 -3.912619  4.806
134 -1.229443
##                   245       249       260       269        270       296        307       315        325
330       334
## Predicted    1.3110677  1.989575 4.119463  7.518783  1.3110677  7.740462  1.3110677 0.9872832  1.3110677  7.22
4957  5.787290
## cvpred       1.2106087  2.018816 4.070119  7.628030  1.2106087  7.658638  1.2106087 0.8941930  1.2106087  7.38
1321  5.748734
## V7           1.0000000  1.000000 8.000000  4.000000  1.0000000 10.000000  1.0000000 1.0000000  1.0000000 10.00
0000 10.000000
## CV residual -0.2106087 -1.018816 3.929881 -3.628030 -0.2106087  2.341362 -0.2106087 0.1058070 -0.2106087  2.61
8679  4.251266
##                   335       337       340       346       357       359       360       363        367       369
373
## Predicted    5.462502  6.036920  5.816245 0.6634986 2.8503169  5.151713 5.612533 1.756908  9.5830222  1.298072
1.9830769
## cvpred       5.541999  6.068411  5.928122 0.5777774 2.8821854  5.264448 5.818674 1.729981  9.4826633  1.171744
1.9993839
## V7          10.000000 10.000000 10.000000 1.0000000 3.0000000  4.000000 7.000000 3.000000 10.0000000  1.000000
1.0000000
## CV residual  4.458001  3.931589  4.071878 0.4222226 0.1178146 -1.264448 1.181326 1.270019  0.5173367 -0.171744
-0.9993839
##                   375       380       382       384       385       387        389        391       415
427       437
## Predicted    1.7569078  3.167603  6.937630 0.8896677 0.8896677  6.214184  1.2134523  1.3195491  5.784303  3.15
4607  5.878383
## cvpred       1.7299814  3.179169  7.154044 0.8471799 0.8471799  6.241276  1.1635956  1.2260297  5.862427  3.14
0304  5.991665
## V7           1.0000000  1.000000 10.000000 1.0000000 1.0000000 10.000000  1.0000000  1.0000000 10.000000  1.00
0000  1.000000
## CV residual -0.7299814 -2.179169  2.845956 0.1528201 0.1528201  3.758724 -0.1635956 -0.2260297  4.137573 -2.14
0304 -4.991665
##                   446       448       449       451       461       463        464       472        479
480       484
## Predicted   0.8896677  1.5681750 0.6634986  2.330322  2.232707  2.458876  1.3420059  2.458876  1.5681750  8.17
8825  8.387027
## cvpred      0.8471799  1.6553874 0.5777774  2.366074  2.319061  2.588463  1.3859849  2.588463  1.6553874  8.05
2908  8.401150
## V7          1.0000000  1.0000000 1.0000000  1.000000  1.000000  1.000000  1.0000000  1.000000  1.0000000 10.00
0000 10.000000
## CV residual 0.1528201 -0.6553874 0.4222226 -1.366074 -1.319061 -1.588463 -0.3859849 -1.588463 -0.6553874  1.94
7092  1.598850
##                   487       489       491       494       497       507        512        513        514
521        522
## Predicted    1.4396214  5.705658 0.6634986  9.033069 0.6634986  9.064007  1.8919595  1.5681750  1.4396214 0.66
34986  1.3420059
## cvpred       1.4329981  5.626893 0.5777774  8.896845 0.5777774  9.072221  1.9718031  1.6553874  1.4329981 0.57
77774  1.3859849
## V7           1.0000000  3.000000 1.0000000 10.000000 1.0000000  5.000000  1.0000000  1.0000000  1.0000000 1.00
00000  1.0000000
## CV residual -0.4329981 -2.626893 0.4222226  1.103155 0.4222226 -4.072221 -0.9718031 -0.6553874 -0.4329981 0.42
22226 -0.3859849
##                   523       529        530        534        542        546       548        555       561
568       585
## Predicted    6.907671  2.761183  1.6657904  1.4396214  1.1158368  1.8919595 0.8896677  1.1158368  2.215744 1.6
65790  3.229504
## cvpred       7.084337  2.850593  1.7024006  1.4329981  1.1165824  1.9718031 0.8471799  1.1165824  2.288219 1.7
02401  3.314571
## V7           5.000000  1.000000  1.0000000  1.0000000  1.0000000  1.0000000 1.0000000  1.0000000  1.000000 3.0
00000  1.000000
## CV residual -2.084337 -1.850593 -0.7024006 -0.4329981 -0.1165824 -0.9718031 0.1528201 -0.1165824 -1.288219 1.2
97599 -2.314571
##                   588       598        603       605        606        614       622       627        630
634       637
## Predicted    1.8919595  2.850317  1.6657904  6.477814  8.1618372  1.2134523  4.712371  6.200209  1.3420059  5.
490477  9.842661
```

```
## cvpred       1.9718031  2.882185  1.7024006  6.490137  8.2374171  1.1635956  4.764508  6.096742  1.3859849  5.
615717  9.954897
## V7           1.0000000  1.000000  1.0000000 10.000000  8.0000000  1.0000000  2.000000  6.000000  1.0000000  3.
000000  1.000000
## CV residual -0.9718031 -1.882185 -0.7024006  3.509863 -0.2374171 -0.1635956 -2.764508 -0.096742 -0.3859849 -2.
615717 -8.954897
##                    639        650        655        656       659       660       661       672       675
681       684
## Predicted    1.3420059  1.4396214  1.7634059  1.4396214  8.177821 0.6634986 0.9872832  2.095672 0.9872832 11.4
5905 0.6634986
## cvpred       1.3859849  1.4329981  1.7494138  1.4329981  8.162589 0.5777774 0.8941930  2.081250 0.8941930 11.5
0952 0.5777774
## V7           1.0000000  1.0000000  1.0000000  1.0000000 10.000000 1.0000000 1.0000000  1.000000 1.0000000 10.0
0000 1.0000000
## CV residual -0.3859849 -0.4329981 -0.7494138 -0.4329981  1.837411 0.4222226 0.1058070 -1.081250 0.1058070 -1.5
0952 0.4222226
##                   686       690       692        693        694
## Predicted   0.6634986 0.6634986  6.724170  1.1158368  1.4396214
## cvpred      0.5777774 0.5777774  6.604831  1.1165824  1.4329981
## V7          1.0000000 1.0000000  5.000000  1.0000000  1.0000000
## CV residual 0.4222226 0.4222226 -1.604831 -0.1165824 -0.4329981
##
## Sum of squares = 834.59    Mean square = 6.09    n = 137
##
## fold 4
## Observations in test set: 137
##                      2         5          8        20         34         35        39       47         48
52        58
## Predicted    4.496667  2.654107  1.8545232  2.441913  1.8695027  1.7569078  6.473300 4.987707 0.98728319 3.847
1146  4.493680
## cvpred       4.496317  2.595512  1.8472985  2.383609  1.8299937  1.7458263  6.442201 5.081462 0.96300316 3.827
3985  4.495312
## V7          10.000000  1.000000  1.0000000  1.000000  1.0000000  1.0000000 10.000000 9.000000 1.00000000 4.000
0000  1.000000
## CV residual  5.503683 -1.595512 -0.8472985 -1.383609 -0.8299937 -0.7458263  3.557799 3.918538 0.03699684 0.172
6015 -3.495312
##                     70        72         75        76         90         95       103      110       116
119       122
## Predicted    1.311068  6.380199  4.2984487 1.9521387  1.5457182  1.5372368  2.306861 5.685708 0.6634986 0.6634
986 1.98957497
## cvpred       1.284331  6.292682  4.2923672 1.9487707  1.5086659  1.5041866  2.287010 5.675677 0.6416753 0.6416
753 1.94389789
## V7           1.000000  2.000000  4.0000000 2.0000000  1.0000000  1.0000000  1.000000 9.000000 5.0000000 3.0000
000 2.00000000
## CV residual -0.284331 -4.292682 -0.2923672 0.0512293 -0.5086659 -0.5041866 -1.287010 3.324323 4.3583247 2.3583
247 0.05610211
##                    132       133       155        163       168       170         171        177       181
188       193
## Predicted    1.5372368  7.427142 0.6634986  1.7634059  9.192560 0.9957645  1.11583682  1.5372368  1.311068  9.
2185528  1.8919595
## cvpred       1.5041866  7.497800 0.6416753  1.7240423  9.283302 0.9674824  1.08138657  1.5041866  1.284331  9.
1961661  1.8424257
## V7           1.0000000 10.000000 1.0000000  1.0000000  1.000000 1.0000000  1.00000000  1.0000000  1.000000 10.
0000000  1.0000000
## CV residual -0.5041866  2.502200 0.3583247 -0.7240423 -8.283302 0.0325176 -0.08138657 -0.5041866 -0.284331  0.
8038339 -0.8424257
##                    206        212       213       216        217       223       227       228       234
239       247
## Predicted    9.0245872  8.7362557  1.311068  7.756421 0.98728319 2.330322  7.866028  8.056744  6.268583 8.0756
904  9.370829
## cvpred       9.1435282  8.7910644  1.284331  7.822602 0.96300316 2.274184  7.905763  8.156755  6.260035 8.1919
766  9.477288
## V7          10.0000000  8.0000000  1.000000  5.000000 1.00000000 5.000000 10.000000  5.000000 10.000000 9.0000
000 10.000000
## CV residual  0.8564718 -0.7910644 -0.284331 -2.822602 0.03699684 2.725816  2.094237 -3.156755  3.739965 0.8080
234  0.522712
##                   252       253       263       264       273       283       300       305       309      310
311
## Predicted    7.936411  4.405550  7.724479  7.936411  4.659956  5.583603  6.394174  6.504238  7.852053 2.410975
1.2134523
## cvpred       7.867287  4.373061  7.830948  7.867287  4.707608  5.545474  6.300635  6.459113  7.897811 2.366698
1.1828588
## V7          10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000 10.000000  3.000000 5.000000
1.0000000
## CV residual  2.132713  5.626939  2.169052  2.132713  5.292392  4.454526  3.699365  3.540887 -4.897811 2.633302
-0.1828588
##                   321       332       336       338        339       342       343       345       347       3
48        352
## Predicted    4.930059  2.215744 0.6634986  1.311068 0.98728319  1.311068 0.8896677  7.888510  2.217727 0.66349
```

```
86  1.5372368
## cvpred       4.900807  2.163754 0.6416753  1.284331 0.96300316  1.284331 0.8615309  7.841417  2.190017 0.64167
53  1.5041866
## V7          10.000000  1.000000 1.0000000  1.000000 1.00000000  1.000000 1.0000000 10.000000  1.000000 1.00000
00  1.0000000
## CV residual  5.099193 -1.163754 0.3583247 -0.284331 0.03699684 -0.284331 0.1384691  2.158583 -1.190017 0.35832
47 -0.5041866
##                    355      356         361        365       372       379        386       390       392
393       394
## Predicted   0.98728319  1.66579  9.90680671  1.5372368  1.298072  2.436419  1.7653891 1.8919595  7.542244  1.4
396214 0.6634986
## cvpred      0.96300316  1.62257 10.01051881  1.5041866  1.327899  2.380136  1.7503055 1.8424257  7.584435  1.4
027144 0.6416753
## V7          1.00000000  1.00000 10.00000000  1.0000000  1.000000  1.000000  1.0000000 2.0000000 10.000000  1.0
000000 1.0000000
## CV residual 0.03699684 -0.62257 -0.01051881 -0.5041866 -0.327899 -1.380136 -0.7503055 0.1575743  2.415565 -0.4
027144 0.3583247
##                    402        405        406        407        410        411       420       423        431
434       436
## Predicted    1.11583682  1.3280304 0.98728319  1.9830769  1.7569078 0.98728319 0.8896677  2.624148 0.98728319
2.097655  6.849736
## cvpred       1.08138657  1.2932895 0.96300316  1.9656819  1.7458263 0.96300316 0.8615309  2.630122 0.96300316
2.076113  6.993694
## V7           1.00000000  1.0000000 1.00000000  1.0000000  1.0000000 1.00000000 1.0000000  1.000000 1.00000000
1.000000 10.000000
## CV residual -0.08138657 -0.2932895 0.03699684 -0.9656819 -0.7458263 0.03699684 0.1384691 -1.630122 0.03699684
-1.076113  3.006306
##                    438      443       445        453      456       457       458      462        465        4
74       483
## Predicted    1.3420059 1.328030  3.553289  1.7803686 3.016307  8.560519  9.360364 1.115837  1.3420059  1.34200
59 11.232883
## cvpred       1.3012422 1.293289  3.471461  1.7330008 2.963488  8.522768  9.446546 1.081387  1.3012422  1.30124
22 11.312741
## V7           1.0000000 3.000000  1.000000  1.0000000 6.000000 10.000000  3.000000 5.000000  1.0000000  1.00000
00  5.000000
## CV residual -0.3012422 1.706711 -2.471461 -0.7330008 3.036512  1.477232 -6.446546 3.918613 -0.3012422 -0.30124
22 -6.312741
##                    485        488       490        496        503        504         506        509       516
520        526
## Predicted    1.8854614 10.8114830 3.0829841  1.4396214  1.9980563  1.9895750  1.11583682  1.5681750  6.877737
6.817819  1.4481027
## cvpred       1.8642097 10.8899413 3.0480488  1.4027144  1.9483771  1.9438979  1.08138657  1.5210978  6.856043
6.925262  1.4071937
## V7           1.0000000 10.0000000 4.0000000  1.0000000  1.0000000  1.0000000  1.00000000  1.0000000 10.000000
10.000000  1.0000000
## CV residual -0.8642097 -0.8899413 0.9519512 -0.4027144 -0.9483771 -0.9438979 -0.08138657 -0.5210978  3.143957
3.074738 -0.4071937
##                    527        535       538       540      554        559       562       572        573
578       581
## Predicted    1.3420059  1.2134523  2.533030  2.118129 1.983077  1.2134523  2.215744  8.795431  1.4396214 0.987
28319  2.209246
## cvpred       1.3012422  1.1828588  2.506865  2.062281 1.965682  1.1828588  2.163754  8.922667  1.4027144 0.963
00316  2.185538
## V7           1.0000000  1.0000000  1.000000  1.000000 5.000000  1.0000000  1.000000 10.000000  1.0000000 1.000
00000  1.000000
## CV residual -0.3012422 -0.1828588 -1.506865 -1.062281 3.034318 -0.1828588 -1.163754  1.077333 -0.4027144 0.036
99684 -1.185538
##                    584       586       589       591       595        599       604        609        632
646       667
## Predicted    1.11583682 0.6634986  8.323337  7.806135  5.535677  1.4396214  7.746960 10.3282067  1.8919595  1.
4396214  2.217727
## cvpred       1.08138657 0.6416753  8.372744  7.898204  5.596383  1.4027144  7.766602 10.4333189  1.8424257  1.
4027144  2.190017
## V7           1.00000000 1.0000000  3.000000  1.000000 10.000000  1.0000000  1.000000 10.0000000  1.0000000  1.
0000000  1.000000
## CV residual -0.08138657 0.3583247 -5.372744 -6.898204  4.403617 -0.4027144 -6.766602 -0.4333189 -0.8424257 -0.
4027144 -1.190017
##                    668        674        678        689       698
## Predicted    1.7634059  1.8854614  1.5681750  1.3420059  6.839297
## cvpred       1.7240423  1.8642097  1.5210978  1.3012422  6.886173
## V7           1.0000000  1.0000000  1.0000000  1.0000000  4.000000
## CV residual -0.7240423 -0.8642097 -0.5210978 -0.3012422 -2.886173
##
## Sum of squares = 778.77    Mean square = 5.68    n = 137
##
## fold 5
## Observations in test set: 136
##                      1         9         13       14        18        19         25         28         31
33        38
```

```
## Predicted    2.215744 0.8896677  3.8386332 1.311068  1.989575  7.235422  1.3110677  1.8919595  1.4396214  7.20
9978  3.737051
## cvpred       2.330188 0.8826526  3.9095676 1.299865  2.072607  7.338340  1.2998648  1.9927914  1.4776299  7.32
1236  3.937354
## V7           1.000000 1.0000000  3.0000000 3.000000  1.000000 10.000000  1.0000000  1.0000000  1.0000000  5.00
0000  1.000000
## CV residual -1.330188 0.1173474 -0.9095676 1.700135 -1.072607  2.661660 -0.2998648 -0.9927914 -0.4776299 -2.32
1236 -2.937354
##                    43        44        49        60        68         96        98       100       107       1
13        114
## Predicted    6.924895  5.146219  2.654107  5.369401  3.491388  1.3110677  2.215744  8.540046  8.851813  8.2666
94  8.4856218
## cvpred       6.781496  5.157252  2.758803  5.489977  3.501266  1.2998648  2.330188  8.672285  8.858217  8.6113
45  8.4923139
## V7          10.000000  1.000000  1.000000  2.000000 10.000000  1.0000000  1.000000 10.000000 10.000000 10.0000
00  8.0000000
## CV residual  3.218504 -4.157252 -1.758803 -3.489977  6.498734 -0.2998648 -1.330188  1.327715  1.141783  1.3886
55 -0.4923139
##                   117       121       126       128       134       135       137       142       144
148       153
## Predicted    3.528824  1.9606200 0.98728319  1.7634059  1.4396214  1.4396214  1.6657904 0.8896677 0.6634986 0.
9872832  8.847847
## cvpred       3.658718  1.9208564 0.96246831  1.8150263  1.4776299  1.4776299  1.7352106 0.8826526 0.6250719 0.
9624683  8.965820
## V7           2.000000  1.0000000 1.00000000  1.0000000  1.0000000  1.0000000  1.0000000 1.0000000 5.0000000 2.
0000000  5.000000
## CV residual -1.658718 -0.9208564 0.03753169 -0.8150263 -0.4776299 -0.4776299 -0.7352106 0.1173474 4.3749281 1.
0375317 -3.965820
##                   157       158       161       174       180       189       190       191       192
196       205
## Predicted    1.3045696  1.5372368  6.894675 10.5543758  3.514849  8.1007036  1.9456406  9.823167  8.837838  1.
989575  1.3110677
## cvpred       1.2403621  1.5574455  6.989541 10.5384684  3.572171  8.2761283  1.8556524  9.870407  8.771671  2.
072607  1.2998648
## V7           1.0000000  1.0000000 10.000000 10.0000000 10.000000  8.0000000  1.0000000  8.000000 10.000000  1.
000000  1.0000000
## CV residual -0.2403621 -0.5574455  3.010459 -0.5384684  6.427829 -0.2761283 -0.8556524 -1.870407  1.228329 -1.
072607 -0.2998648
##                   209       214      224       229       230       246       251       256       257
258       262
## Predicted    1.3110677 10.4876985 6.214184  1.3110677  8.809406  2.5480100 0.98078507  4.134442  1.1158368  1.
4396214  8.999143
## cvpred       1.2998648 10.5566022 6.270600  1.2998648  8.829710  2.6732857 0.90296568  4.062755  1.1402334  1.
4776299  8.908496
## V7           1.0000000 10.0000000 8.000000  1.0000000 10.000000  2.0000000 1.00000000 10.000000  1.0000000  1.
0000000 10.000000
## CV residual -0.2998648 -0.5566022 1.729400 -0.2998648  1.170290 -0.6732857 0.09703432  5.937245 -0.1402334 -0.
4776299  1.091504
##                   266       277       278     280       284       285      286       301       308       31
3       314
## Predicted    3.167603  1.4396214 0.98728319 5.91484  5.580591  6.927621 11.00671  8.374031  1.3110677  6.83650
4 0.6634986
## cvpred       3.163869  1.4776299 0.96246831 5.97165  5.682354  7.044805 11.05363  8.325952  1.2998648  7.02449
2 0.6250719
## V7           1.000000  1.0000000 1.00000000 7.00000 10.000000 10.000000 10.00000  4.000000  1.0000000  1.00000
0 1.0000000
## CV residual -2.163869 -0.4776299 0.03753169 1.02835  4.317646  2.955195 -1.05363 -4.325952 -0.2998648 -6.02449
2 0.3749281
##                   320       326          329       370       383       396       399       400      401
409
## Predicted    5.2333701  1.7569078  3.861090053  1.6218560  2.412958  1.4396214  1.4396214  1.2980715 7.920713
2.1867891
## cvpred       5.2851677  1.7555237  4.001815629  1.5182559  2.436018  1.4776299  1.4776299  1.1808595 7.843249
2.1784372
## V7           5.0000000  1.0000000  4.000000000  1.0000000  1.000000  1.0000000  1.0000000  1.0000000 9.000000
2.0000000
## CV residual -0.2851677 -0.7555237 -0.001815629 -0.5182559 -1.436018 -0.4776299 -0.4776299 -0.1808595 1.156751
-0.1784372
##                   419       422       424       425       430      435       440       460       466
467       468
## Predicted    2.8652964  9.8146854  2.526532  1.1158368  1.2134523 5.998480  1.568175  2.202748  8.856328  7.20
7995  6.652547
## cvpred       2.9511795  9.8647051  2.548579  1.1402334  1.2200491 5.964919  1.655395  2.211183  8.971521  7.37
5037  6.699215
## V7           2.0000000 10.0000000  1.000000  1.0000000  1.0000000 8.000000  1.000000  1.000000  4.000000 10.00
0000 10.000000
## CV residual -0.9511795  0.1352949 -1.548579 -0.1402334 -0.2200491 2.035081 -0.655395 -1.211183 -4.971521  2.62
4963  3.300785
##                   469       470       476       478      486       493       499       508       510
```

```
515       517
## Predicted    1.3420059  1.3045696  1.1158368  1.3420059 1.328030  1.6657904  1.6657904 0.6634986 0.8896677  8.
9549474 0.6634986
## cvpred       1.3978142  1.2403621  1.1402334  1.3978142 1.311268  1.7352106  1.7352106 0.6250719 0.8826526  9.
0299941 0.6250719
## V7           1.0000000  1.0000000  1.0000000  1.0000000 3.000000  1.0000000  1.0000000 4.0000000 1.0000000 10.
0000000 1.0000000
## CV residual -0.3978142 -0.2403621 -0.1402334 -0.3978142 1.688732 -0.7352106 -0.7352106 3.3749281 0.1173474  0.
9700059 0.3749281
##                   532       545       547       565       575       576       582       583      587       592
594
## Predicted    1.983077  2.180291  9.583022  1.989575  7.209978  2.533030  8.027790  6.011476 11.00671  6.397423
1.8854614
## cvpred       2.013104  2.118935  9.526279  2.072607  7.321236  2.608082  7.896307  6.083924 11.05363  6.289883
1.9332888
## V7           1.000000  1.000000 10.000000  1.000000  2.000000  1.000000 10.000000 10.000000 10.00000 10.000000
1.0000000
## CV residual -1.013104 -1.118935  0.473721 -1.072607 -5.321236 -1.608082  2.103693  3.916076 -1.05363  3.710117
-0.9332888
##                    596        602       608       610       612      613        619        620        621
623       625
## Predicted    1.8919595 0.98728319 0.6634986  1.568175  9.680638 11.00671  1.6657904  1.8919595  1.4396214  3.3
26116  2.224225
## cvpred       1.9927914 0.96246831 0.6250719  1.655395  9.606095 11.05363  1.7352106  1.9927914  1.4776299  3.4
72042  2.335889
## V7           1.0000000 1.00000000 1.0000000  1.000000  2.000000 10.00000  1.0000000  1.0000000  1.0000000  1.0
00000  1.000000
## CV residual -0.9927914 0.03753169 0.3749281 -0.655395 -7.606095 -1.05363 -0.7352106 -0.9927914 -0.4776299 -2.4
72042 -1.335889
##                   629       631        635        643       645        647        648      651        652
653        654
## Predicted   0.8896677  2.428917  1.1158368  1.4396214 0.8896677 0.98078507  1.3280304 1.448103  1.6518150  1.8
919595  1.6657904
## cvpred      0.8826526  2.468763  1.1402334  1.4776299 0.8826526 0.90296568  1.3112675 1.483331  1.6486639  1.9
927914  1.7352106
## V7          1.0000000  1.000000  1.0000000  1.0000000 1.0000000 1.00000000  1.0000000 4.000000  1.0000000  1.0
000000  1.0000000
## CV residual 0.1173474 -1.468763 -0.1402334 -0.4776299 0.1173474 0.09703432 -0.3112675 2.516669 -0.6486639 -0.9
927914 -0.7352106
##                    657       669       671       680       687
## Predicted    1.8919595  4.462741 7.2881239 0.8896677 0.6634986
## cvpred       1.9927914  4.513455 7.2336597 0.8826526 0.6250719
## V7           1.0000000  1.000000 8.0000000 1.0000000 1.0000000
## CV residual -0.9927914 -3.513455 0.7663403 0.1173474 0.3749281
##
## Sum of squares = 591.02    Mean square = 4.35    n = 136
##
## Overall (Sum over all 136 folds)
##       ms
## 5.199782
```

```
# make the prediction, we used the lm_revised model with r2 of 0.6129
v7_pred <- predict(lm_revised, newdata = df[index,])
v7_pred # print predict
```

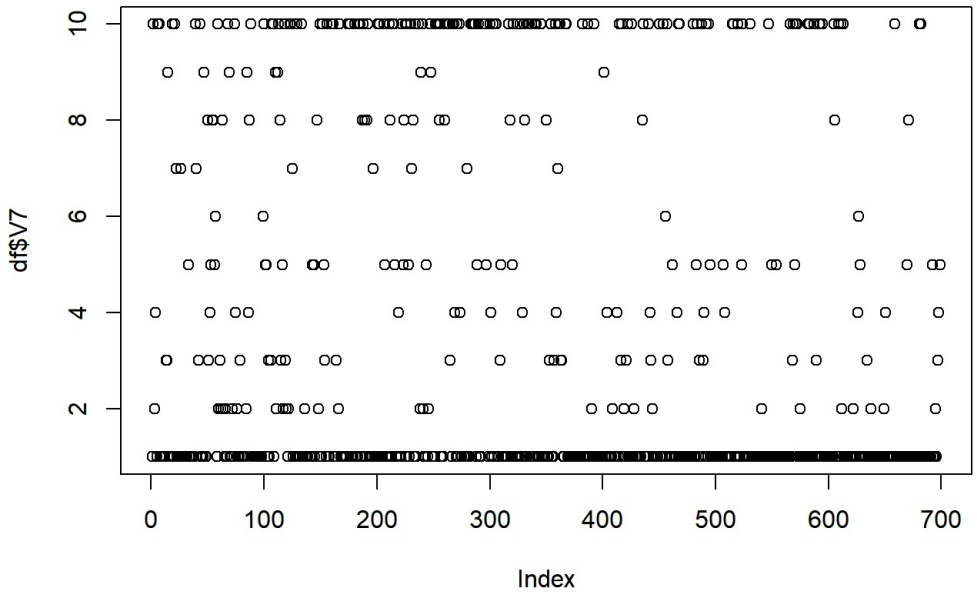
```
##        24        41       140       146       159       165       236       250       276       293       295
298       316
## 5.4585352 7.9816106 0.9872832 1.6218560 0.9807851 2.2157441 2.7152652 1.7634059 2.0741942 6.0866099 0.9872832
2.5265324 5.2438347
##       322       412       618
## 1.7634059 0.9872832 0.6634986
```

```
# imputation
df_imp_reg <- df # since we need to make three imputation, hence we need to
# copy df everytime we made imputation.
df_imp_reg[index,]$V7 <- v7_pred # impute prediction to column v7
unique(df_imp_reg$V7) # check if imputation works
```

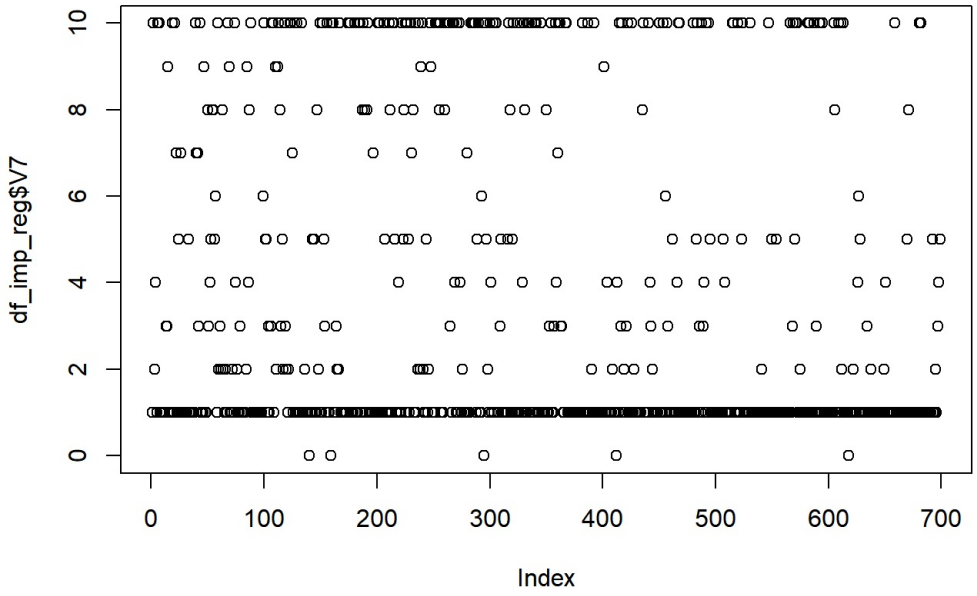
```
##  [1] "1"                 "10"                "2"                 "4"                 "3"                 "9"
##  [7] "7"                 "5.45853515759216"  "5"                 "7.98161057649732"  "8"                 "6"
## [13] "0.987283186665087" "1.62185603726251"  "0.980785074712774" "2.21574405733159"  "2.71526516651999"  "1.76
340589062385"
## [19] "2.07419420397025"  "6.08660989623727"  "2.52653237067799"  "5.24383468761997"  "0.663498649414061"
```

```
df_imp_reg$V7 <- as.integer(df_imp_reg$V7) # maintain the uniformness of the v7
# visualization
plot(df$V7) # we can see range of columns v7
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): NAs introduced by coercion
```




```
#
plot(df_imp_reg$V7) # visualize column 7 to see the distribution after imputation
```




```
                  # by transform to integer
max(df_imp_reg$V7) # the range of the pattern of column v7 from original dataframe
```

```
## [1] 10
```

```
                 # is between 1 - 10. Hence, we need to make sure the range
                 # is still between 0 and 10.
min(df_imp_reg$V7)
```

```
## [1] 0
```

```
# restrain the range of values between 1 and 10
df_imp_reg$V7[df_imp_reg$V7 < 1] <- 1 # because original v7 have range between 1
```

```
v7_pred_pert <- rnorm(nrow(df[index,]), mean(v7_pred), sd(v7_pred)) # using random generation for normal
                                          # distribution with mean equal to mean(v7_pred) and sd equal
                                          # to std(v7_pred)
v7_pred_pert # print values
```

```
##  [1]  4.2306369 -2.0842554  1.8520811  0.3420158  2.7298681  3.8558024  0.8299114  0.4131121  7.4782691  4.433
3393  4.5520695
## [12]  3.2126005  3.9946614  4.7959893  1.0360170  0.5116821
```

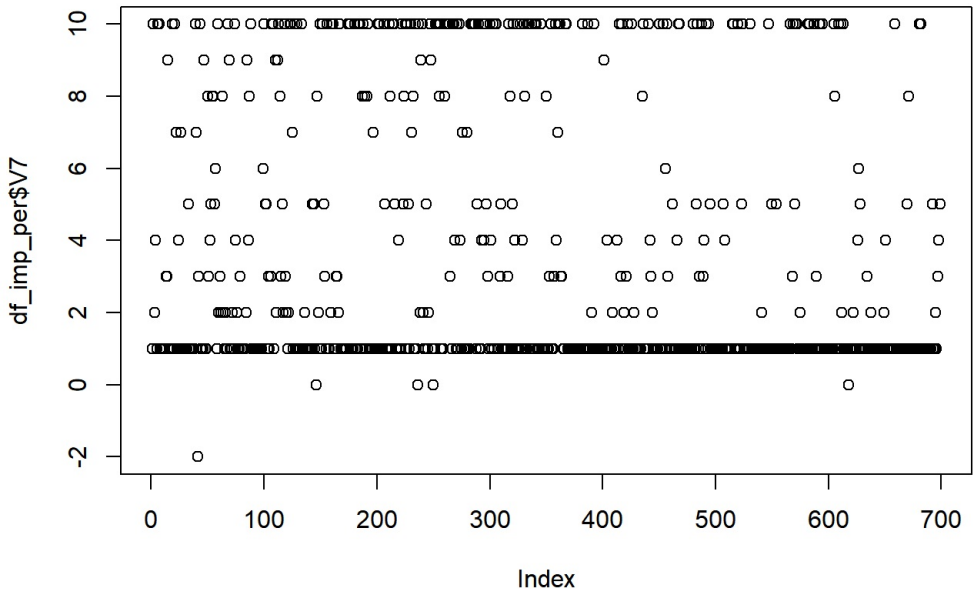
```
df_imp_per <- df # copy df and assign to df_imp_per
df_imp_per[index,]$V7 <- v7_pred_pert # impute v7_pred_pert to v7 column according
                                      # to index
df_imp_per$V7 <- as.integer(df_imp_per$V7) # convert to integer
max(df_imp_per$V7) # get max
```

```
## [1] 10
```

```
min(df_imp_per$V7) # get min
```

```
## [1] -2
```

```
#
plot(df_imp_per$V7) # visualize column v7 to check its range
```




```
#
# restrict range of value to 1,10
df_imp_per$V7[df_imp_per$V7 < 1] <- 1 # restrain range to specific values
```

```
set.seed(9876) # make code reproducible
# train & test split
split <- sample(c(rep(0, 0.7 * nrow(df)), rep(1, 0.3 * nrow(df)))) # split the
# dataframe
table(split) # using table to check out
```

```
## split
##   0   1
## 489 209
```

```
# train test split
train <- df[split==0, ] # 70% data assigned to train
test <- df[split== 1, ] # 30% data assigned to test
```

```
accuracy_mean <- c(0,30) # create vector
# loop through k 1:30
for (k in 1:30) {
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, df_imp_mean[split==0,], df_imp_mean[split==1,], k=k, scale =
TRUE)
  pred <- as.integer(fitted(knn_model)) # fit model
  accuracy_mean[k] <- sum(pred == df_imp_mean[split==1,]$V11) / nrow(df_imp_mean[split==1,])
}
accuracy_mean # print accuracy
```

```
##  [1] 0.9569378 0.9282297 0.9138756 0.9090909 0.9090909 0.9043062 0.8899522 0.8899522 0.8803828 0.8564593 0.851
6746 0.8516746
## [13] 0.8421053 0.8421053 0.8373206 0.8325359 0.8229665 0.8133971 0.7990431 0.7942584 0.7799043 0.7799043 0.775
1196 0.7703349
## [25] 0.7607656 0.7559809 0.7511962 0.7511962 0.7464115 0.7464115
```

```
mean(accuracy_mean) # get mean
```

```
## [1] 0.8314195
```

```
accuracy_regre <- c(0,30)
for (k in 1:30) {
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, df_imp_reg[split==0,], df_imp_reg[split==1,], k=k, scale = T
RUE)
  pred <- as.integer(fitted(knn_model))
  accuracy_regre[k] <- sum(pred == df_imp_reg[split==1,]$V11) / nrow(df_imp_reg[split==1,])
}
accuracy_regre
```

```
##  [1] 0.9282297 0.9138756 0.9043062 0.8995215 0.8995215 0.8899522 0.8660287 0.8516746 0.8421053 0.8325359 0.818
1818 0.8133971
## [13] 0.8038278 0.7894737 0.7846890 0.7607656 0.7559809 0.7559809 0.7511962 0.7464115 0.7416268 0.7368421 0.732
0574 0.7272727
## [25] 0.7177033 0.7177033 0.7177033 0.7177033 0.7177033 0.7081340
```

```
mean(accuracy_regre)
```

```
## [1] 0.7947368
```

```
accuracy_per <- c(0,30)
for (k in 1:30) {
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, df_imp_per[split==0,], df_imp_per[split==1,], k=k, scale = T
RUE)
  pred <- as.integer(fitted(knn_model))
  accuracy_per[k] <- sum(pred == df_imp_per[split==1,]$V11) / nrow(df_imp_per[split==1,])
}
accuracy_per
```

```
##  [1] 0.9330144 0.9282297 0.9090909 0.9090909 0.9090909 0.8995215 0.8851675 0.8660287 0.8564593 0.8468900 0.837
3206 0.8373206
## [13] 0.8229665 0.8086124 0.7990431 0.7894737 0.7799043 0.7799043 0.7751196 0.7751196 0.7751196 0.7703349 0.755
9809 0.7464115
## [25] 0.7416268 0.7416268 0.7368421 0.7368421 0.7368421 0.7368421
```

```
mean(accuracy_per)
```

```
## [1] 0.8141946
```

```
df_remove <- df[-index,] # remove nas according to index
dim(df_remove) # check dimension
```

```
## [1] 683  11
```

```
df_remove$V7 <- as.integer(df_remove$V7) # however, this method didn't work out because
                                          # length of independent variables didn't match
                                          # the length of response variable (because we only
                                          # remove na from independent variables, however,
                                          # length of response variable didn't change)
# another approach
df_removed <- df # replicate dataframe df
df_removed$V7[df$V7 == "?"] <- NA # substitute "?" with na
df_removed$V7 <- as.integer(df_removed$V7) # set $v7 to integer
# check uniqueness
unique(df_removed$V7)
```

```
##  [1]  1 10  2  4  3  9  7 NA  5  8  6
```

```
# check dimension
dim(df_removed)
```

```
## [1] 699  11
```

```
accuracy_removed <- c(0,30) # create vector
for (k in 1:30) { # loop through
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, df_removed[split==0,], df_removed[split==1,], k=k,
                    na.action = na.exclude(), scale = TRUE) # using na.exclude to build up model which
                                                             # will retain original number of rows in dataframe
  pred <- as.integer(fitted(knn_model)) # fit model
  accuracy_removed[k] <- sum(pred == df_removed[split==1,]$V11) / nrow(df_removed[split==1,])
}
```

```
## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th
```

```
## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th

## Warning in pred == df_removed[split == 1, ]$V11: longer object length is not a multiple of shorter object leng
th
```

```
accuracy_removed # print
```

```
##  [1] 0.6602871 0.6555024 0.6459330 0.6459330 0.6411483 0.6411483 0.6315789 0.6220096 0.6172249 0.6172249 0.612
4402 0.6124402
## [13] 0.5980861 0.5885167 0.5837321 0.5789474 0.5789474 0.5789474 0.5741627 0.5741627 0.5741627 0.5741627 0.569
3780 0.5693780
## [25] 0.5693780 0.5693780 0.5645933 0.5645933 0.5645933 0.5550239
```

```
mean(accuracy_removed) # however, the r2 is 59%, which is significantly lower than
```

```
## [1] 0.5977671
```

```
                       # r2 of other methods, and i can not able to find out why. Therefore, in order to
                       # maintain the same length of independent variables and response variables, i substitute
                       # nas with 0
```

```
df_removed2 <- df
df_removed2$V7[df$V7 == "?"] <- 0 # assign 0 to the substitute nas. the reason is
                                  # we need to make sure the lenght of independent
                                  # variables = response variables
df_removed2$V7 <- as.integer(df_removed2$V7) # convert to integer
unique(df_removed2$V7) # check unique values
```

```
##  [1]  1 10  2  4  3  9  7  0  5  8  6
```

```
accuracy_removed2 <- c(0,30) # create vector
for (k in 1:30) { # loop through
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V7+V8+V9+V10, df_removed2[split==0,], df_removed2[split==1,], k=k, scale =
TRUE)
  pred <- as.integer(fitted(knn_model))
  accuracy_removed2[k] <- sum(pred == df_removed2[split==1,]$V11) / nrow(df_removed2[split==1,])
}
accuracy_removed2
```

```
##  [1] 0.9377990 0.9282297 0.9138756 0.9090909 0.9090909 0.9043062 0.8851675 0.8708134 0.8612440 0.8516746 0.842
1053 0.8421053
## [13] 0.8181818 0.8133971 0.8038278 0.7894737 0.7894737 0.7894737 0.7846890 0.7751196 0.7751196 0.7703349 0.760
7656 0.7511962
## [25] 0.7511962 0.7511962 0.7464115 0.7416268 0.7368421 0.7368421
```

```
mean(accuracy_removed2)
```

```
## [1] 0.8180223
```

```
# however, still doesn't know why there are so much inconsistency in  the accuracy results.
```

```
df_bin <- df # copy entire dataframe
df_bin$v12[df$V7 == "?"] <- 0 # the important thing here is we need to create a new column
                              # and assign 0 to null values
df_bin$v12[df$V7 != "?"] <- 1 # assign 1 to values which are not null
# check if the correct values have been assigned
unique(df_bin$v12)
```

```
## [1] 1 0
```

```
# build up a knn model
accuracy_bin <- c(0,30) # create vector
for (k in 1:30) { # loop through
  knn_model <- kknn(V11~V2+V3+V4+V5+V6+V8+V9+V10+v12, df_bin[split==0,], df_bin[split==1,], k=k, scale = TRUE)
  # important thing here is we build up model with newly created binary columns only,
  # ignore original v7.
  pred <- as.integer(fitted(knn_model)) # fit model
  accuracy_bin[k] <- sum(pred == df_bin[split==1,]$V11) / nrow(df_bin[split==1,])
}
accuracy_bin
```

```
##  [1] 0.9377990 0.9234450 0.9234450 0.9138756 0.9090909 0.8899522 0.8708134 0.8612440 0.8564593 0.8421053 0.837
3206 0.8277512
## [13] 0.8277512 0.8229665 0.8086124 0.8038278 0.7942584 0.7894737 0.7894737 0.7846890 0.7846890 0.7799043 0.770
3349 0.7655502
## [25] 0.7607656 0.7559809 0.7559809 0.7559809 0.7511962 0.7511962
```

```
mean(accuracy_bin)
```

```
## [1] 0.8215311
```

```
# create vector to store the accuracy
# after completion of knn, SVM is straightforward.
svm_mean <- c(0,5) # create vector
# create vector of c
c_lv <- c(0.001, 0.1, 10, 100, 1000) # customized c values
# column converted
df_imp_mean$V7 <- as.numeric(df_imp_mean$V7) # it is important to convert
                                             # column to numeric, otherwise there
                                             # will be error in svm_model
```

```
for (i in 1:5) { # loop through
  svm_model <- ksvm(as.matrix(df_imp_mean[split==0,2:10]),
                    as.factor(df_imp_mean[split==0,11]),
                    type = "C-svc",
                    kernel = "vanilladot",
                    C = c_lv[i],
                    scaled = TRUE)
  pred <- predict(svm_model, df_imp_mean[split==1,2:10])
  svm_mean[i] = sum(pred == df_imp_mean[split==1,11]) / nrow(df_imp_mean[split==1,])
}
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
svm_mean
```

```
## [1] 0.9521531 0.9665072 0.9617225 0.9617225 0.9617225
```

```
mean(svm_mean)
```

```
## [1] 0.9607656
```

```
svm_reg <- c(0,5)
df_imp_reg$V7 <- as.numeric(df_imp_reg$V7)
for (i in 1:5) {
  svm_model <- ksvm(as.matrix(df_imp_reg[split==0,2:10]),
                    as.factor(df_imp_reg[split==0,11]),
                    type = "C-svc",
                    kernel = "vanilladot",
                    C = c_lv[i],
                    scaled = TRUE)
  pred <- predict(svm_model, df_imp_reg[split==1,2:10])
  svm_reg[i] = sum(pred == df_imp_reg[split==1,11]) / nrow(df_imp_reg[split==1,])
}
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
svm_reg
```

```
## [1] 0.9521531 0.9665072 0.9617225 0.9617225 0.9617225
```

```
mean(svm_reg)
```

```
## [1] 0.9607656
```

```
svm_per <- c(0,5)
df_imp_per$V7 <- as.numeric(df_imp_per$V7)
for (i in 1:5) {
  svm_model <- ksvm(as.matrix(df_imp_per[split==0,2:10]),
                    as.factor(df_imp_per[split==0,11]),
                    type = "C-svc",
                    kernel = "vanilladot",
                    C = c_lv[i],
                    scaled = TRUE)
  pred <- predict(svm_model, df_imp_per[split==1,2:10])
  svm_per[i] = sum(pred == df_imp_per[split==1,11]) / nrow(df_imp_per[split==1,])
}
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
svm_per
```

```
## [1] 0.9521531 0.9665072 0.9617225 0.9617225 0.9617225
```

```
mean(svm_per)
```

```
## [1] 0.9607656
```

```
svm_bin <- c(0,5)
df_bin$v12 <- as.numeric(df_bin$v12) # set numeric, the reason has been mentioned above
dim(df_bin) # check the dimensions
```

```
## [1] 699  12
```

```
str(df_bin) # check the structures, main reason for that is to see which columns
```

```
## 'data.frame':    699 obs. of  12 variables:
##  $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 ...
##  $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
##  $ V7 : chr  "1" "10" "2" "4" ...
##  $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
##  $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
##  $ v12: num  1 1 1 1 1 1 1 1 1 1 ...
```

```
           # should be used in train the model
for (i in 1:5) {
  svm_model <- ksvm(as.matrix(df_bin[split==0,c(2:6, 8:10, 12)]), # noticed that
                    # we created binary column, hence it will be used in train
                    # the model
                    as.factor(df_bin[split==0,11]),
                    type = "C-svc",
                    kernel = "vanilladot",
                    C = c_lv[i],
                    scaled = TRUE)
  pred <- predict(svm_model, df_bin[split==1,c(2:6, 8:10, 12)])
  svm_bin[i] = sum(pred == df_bin[split==1,11]) / nrow(df_bin[split==1,])
}
```

```
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
##  Setting default kernel parameters
```

```
svm_bin
```

```
## [1] 0.9473684 0.9569378 0.9569378 0.9569378 0.9569378
```

```
mean(svm_bin)
```

```
## [1] 0.9550239
```