reto knn

Oliver Rodriguez

9/3/2021

Attribute Information:

The inputs are as follows

X1=the transaction date (for example, 2013.250=2013 March, 2013.500=2013 June, etc.)

X2=the house age (unit: year)

X3=the distance to the nearest MRT station (unit: meter)

X4=the number of convenience stores in the living circle on foot (integer)

X5=the geographic coordinate, latitude. (unit: degree)

X6=the geographic coordinate, longitude. (unit: degree)

The output is as follow

Y= house price of unit area (10000 New Taiwan Dollar/Ping, where Ping is a local unit, 1 Ping = 3.3 meter squared)

Algunas librerias a utilizar:

```
library(caret)
library(tidyverse)
library(kableExtra)
```

Cargamos los datos exculyendo $X4$ $X5$ que son coordenadas, obtenemos resumenes de las variables y tambien se normalizan los datos para modelar:

```
setwd('C:/Users/Oliver/Documents/9/TAE/reto 3 superficie de respuesta knn')
datos_completos <- readxl::read_xlsx('Real estate valuation data set (1).xlsx',
                              col_names = c('n', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'y'), skip = 1)
datos <- datos_completos %>%
              select('x1', 'x2', 'x3', 'x4', 'y')

# un análisis rapido a lo que son los datos:
kbl(dim(datos))
```

x
414
5

```
kbl(names(datos))
```

x
x1
x2
x3
x4
y

```
kbl(head(datos))
```

x1	x2	x3	x4	y
2012.917	32.0	84.87882	10	37.9
2012.917	19.5	306.59470	9	42.2
2013.583	13.3	561.98450	5	47.3
2013.500	13.3	561.98450	5	54.8
2012.833	5.0	390.56840	5	43.1
2012.667	7.1	2175.03000	3	32.1

```
kbl(tail(datos))
```

x1	x2	x3	x4	y
2013.417	18.5	2175.74400	3	28.1
2013.000	13.7	4082.01500	0	15.4
2012.667	5.6	90.45606	9	50.0
2013.250	18.8	390.96960	7	40.6
2013.000	8.1	104.81010	5	52.5
2013.500	6.5	90.45606	9	63.9

```
kbl(summary(datos))
```

	x1	x2	x3	x4	y
	Min. :2013	Min. : 0.000	Min. : 23.38	Min. : 0.000	Min. : 7.60
	1st Qu.:2013	1st Qu.: 9.025	1st Qu.: 289.32	1st Qu.: 1.000	1st Qu.: 27.70
	Median :2013	Median :16.100	Median : 492.23	Median : 4.000	Median : 38.45
	Mean :2013	Mean :17.713	Mean :1083.89	Mean : 4.094	Mean : 37.98
	3rd Qu.:2013	3rd Qu.:28.150	3rd Qu.:1454.28	3rd Qu.: 6.000	3rd Qu.: 46.60
	Max. :2014	Max. :43.800	Max. :6488.02	Max. :10.000	Max. :117.50

```
str(datos_completos)
```

```
## tibble [414 x 8] (S3: tbl_df/tbl/data.frame)
##  $ n : num [1:414] 1 2 3 4 5 6 7 8 9 10 ...
##  $ x1: num [1:414] 2013 2013 2014 2014 2013 ...
##  $ x2: num [1:414] 32 19.5 13.3 13.3 5 7.1 34.5 20.3 31.7 17.9 ...
##  $ x3: num [1:414] 84.9 306.6 562 562 390.6 ...
##  $ x4: num [1:414] 10 9 5 5 5 3 7 6 1 3 ...
##  $ x5: num [1:414] 25 25 25 25 25 ...
##  $ x6: num [1:414] 122 122 122 122 122 ...
##  $ y : num [1:414] 37.9 42.2 47.3 54.8 43.1 32.1 40.3 46.7 18.8 22.1 ...
```

```
kbl(table(datos_completos$x4))
```

Var1	Freq
0	67
1	46
2	24
3	46
4	31
5	67
6	37
7	31
8	30
9	25
10	10

```
# normalizando los datos para evitar confusiones por la diferencia de las escalas al modelar
# y más presiscion en la variabildad del error de validacion. Extraigo media desviacion:
datoc <- scale(datos[,c("x1", "x2", "x3", "x4", "y" )], center = T, scale = T)
centro<-attr(datoc,"center")
escala<-attr(datoc,"scale")
datos<-as.data.frame(datoc)
```

Buscando el k optimo:

Para esto se seleccionan todas las variables, se utiliza validacion cruzada repetida 3 veces con 10 subconjuntos, y se obtiene primero el k optimo y luego el resumen de criterios de seleccion:

```
# Se usará CV repetido 3 veces con k-folds=10:
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

# seleccion de k optimo con 4 variables usanco 3 CV con k=10, y con k vecinos de 1-30:
knn_fit <- train(y ~., data = datos, method = "knn",
                  trControl=trctrl,
                  preProcess = c( "knnImpute"),
                 tuneGrid   = expand.grid(k = 1:30))

#Su resultado en los diferentes criterios de seleccion y el k vecimo con mejor resultado:
knn_fit$bestTune
```

```
##   k
## 8 8
```

```
knn_fit
```

```
## k-Nearest Neighbors
##
## 414 samples
##   4 predictor
##
```

```
## Pre-processing: nearest neighbor imputation (4), centered (4), scaled (4)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 372, 373, 373, 372, 373, 372, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.7971447  0.4811541  0.5018176
##    2  0.7120905  0.5428740  0.4731490
##    3  0.6697005  0.5735848  0.4555792
##    4  0.6444405  0.5938790  0.4484107
##    5  0.6335439  0.6035499  0.4450777
##    6  0.6352759  0.5987145  0.4521722
##    7  0.6303710  0.6044547  0.4477637
##    8  0.6271827  0.6083948  0.4469056
##    9  0.6292563  0.6059572  0.4497787
##   10  0.6300766  0.6047566  0.4511342
##   11  0.6291219  0.6060469  0.4491387
##   12  0.6290039  0.6061697  0.4505617
##   13  0.6279552  0.6078662  0.4502764
##   14  0.6289462  0.6070426  0.4496724
##   15  0.6321687  0.6035004  0.4519303
##   16  0.6326361  0.6031954  0.4532809
##   17  0.6316153  0.6052516  0.4515595
##   18  0.6312847  0.6066927  0.4526768
##   19  0.6318470  0.6062941  0.4539835
##   20  0.6313963  0.6076774  0.4531351
##   21  0.6316234  0.6083333  0.4523545
##   22  0.6316898  0.6092049  0.4515835
##   23  0.6320912  0.6093942  0.4517922
##   24  0.6322401  0.6095707  0.4517438
##   25  0.6331686  0.6091915  0.4523206
##   26  0.6347963  0.6073934  0.4530345
##   27  0.6360572  0.6058101  0.4537358
##   28  0.6366056  0.6059668  0.4547395
##   29  0.6371834  0.6059485  0.4549043
##   30  0.6376835  0.6064242  0.4556475
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 8.
```

Selección del número de variables óptimas:

Para este caso, simplemente se utilizara la matris de varianzas covarianzas:

```r
# RemoveRedundant Features
correlationMatrix <- cor(datos[,1:4])
# find attributes that are highly corrected (ideally >0.75)
highlyCorrelated <- findCorrelation(correlationMatrix, cutoff=0.6)
# print indexes of highly correlated attributes
print(highlyCorrelated)
```

```
## [1] 3
```

La caracteristica x3 podria considerarse descartarla, por mostrar correlacion -0.602519145 con x4, pero no supera limite que seria 0.75%, para descartar.

Seleccione el mejor modelo de dos variables y grafique la superficie de respuesta:

Se realizaron 6 modelos con las conbinaciones de variables se obtendran la lista de medidas de error para seleccion de mejor modelo, el mejor k y la grafica que muestra los k vs MSE: :

Este modelo contiene $x1 \quad x2$ que es la fecha y la edad de la vivienda,

```
knn_fit1 <- train(y ~., data = datos[,c(1,2,5)], method = "knn",
                 trControl=trctrl,
                 preProcess = c( "knnImpute"),
                 tuneGrid   = expand.grid(k = 1:30))

knn_fit1
```

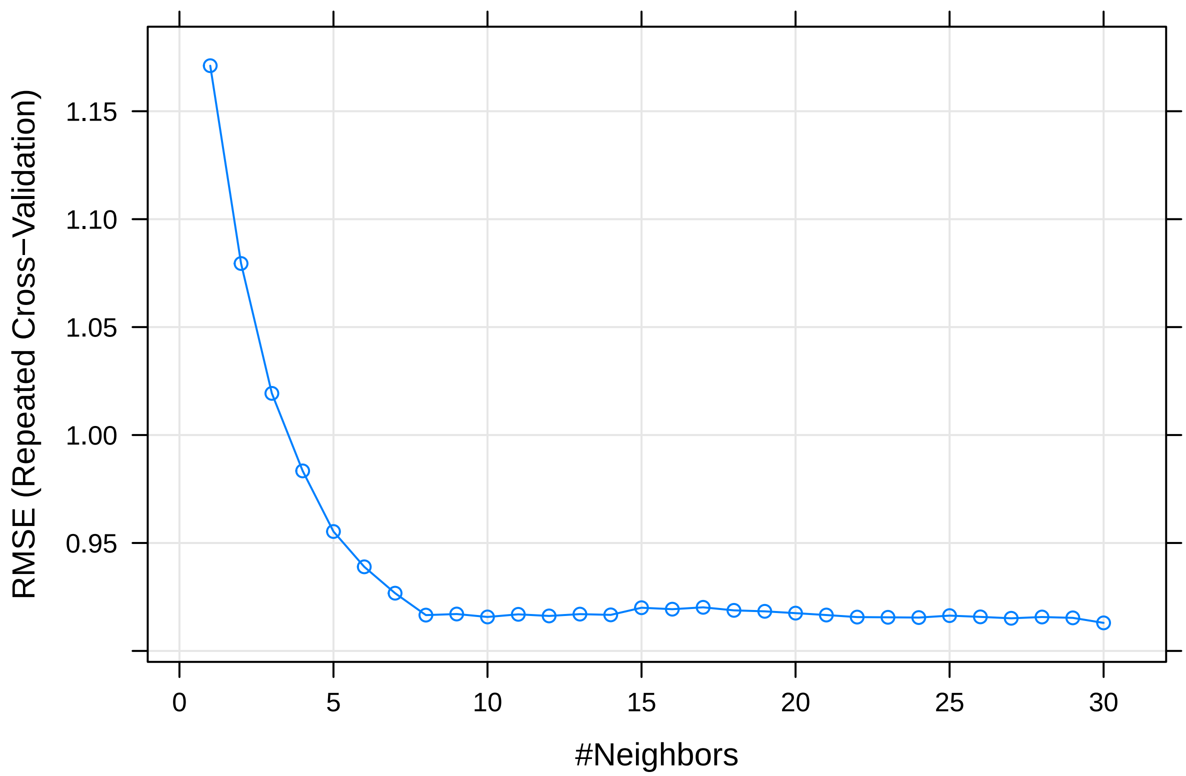
```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 372, 373, 373, 373, 372, 373, ...
## Resampling results across tuning parameters:
##
##   k  RMSE       Rsquared    MAE
##    1  1.1710831  0.09268042  0.9058600
##    2  1.0794566  0.09470953  0.8501954
##    3  1.0193012  0.11186697  0.8105644
##    4  0.9833932  0.12475318  0.7924546
##    5  0.9553063  0.14274695  0.7787793
##    6  0.9389563  0.15625733  0.7670895
##    7  0.9267410  0.16743343  0.7584416
##    8  0.9165833  0.17940884  0.7505849
##    9  0.9170965  0.17893775  0.7490298
##   10  0.9157296  0.17690396  0.7479293
##   11  0.9169393  0.17504438  0.7469657
##   12  0.9162062  0.17414166  0.7435512
##   13  0.9170475  0.16957967  0.7407426
##   14  0.9166904  0.16829369  0.7411027
##   15  0.9200007  0.16417384  0.7418267
##   16  0.9193286  0.16423015  0.7413132
##   17  0.9201911  0.16271781  0.7435735
##   18  0.9187840  0.16443868  0.7418576
##   19  0.9183402  0.16553172  0.7409904
##   20  0.9174991  0.16741179  0.7411909
##   21  0.9166179  0.16899928  0.7410409
##   22  0.9156667  0.17139258  0.7382131
##   23  0.9155737  0.17177433  0.7367368
```

```
##   24  0.9154525  0.17230539  0.7352607
##   25  0.9163486  0.17125454  0.7348821
##   26  0.9157656  0.17273508  0.7335813
##   27  0.9151146  0.17288985  0.7332299
##   28  0.9157049  0.17189515  0.7334150
##   29  0.9153155  0.17298587  0.7337926
##   30  0.9129892  0.17701887  0.7330499
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 30.
```

```r
knn_fit1$bestTune
```

```
##     k
## 30 30
```

```r
plot(knn_fit1)
```



Este modelo contiene $x1$ $x3$ son la fecha y la distancia a la estacion mas cercana:

```r
knn_fit2 <- train(y ~., data = datos[,c(1,3,5)], method = "knn",
                  trControl=trctrl,
                  preProcess = c("knnImpute"),
                  tuneGrid   = expand.grid(k = 1:30))
```

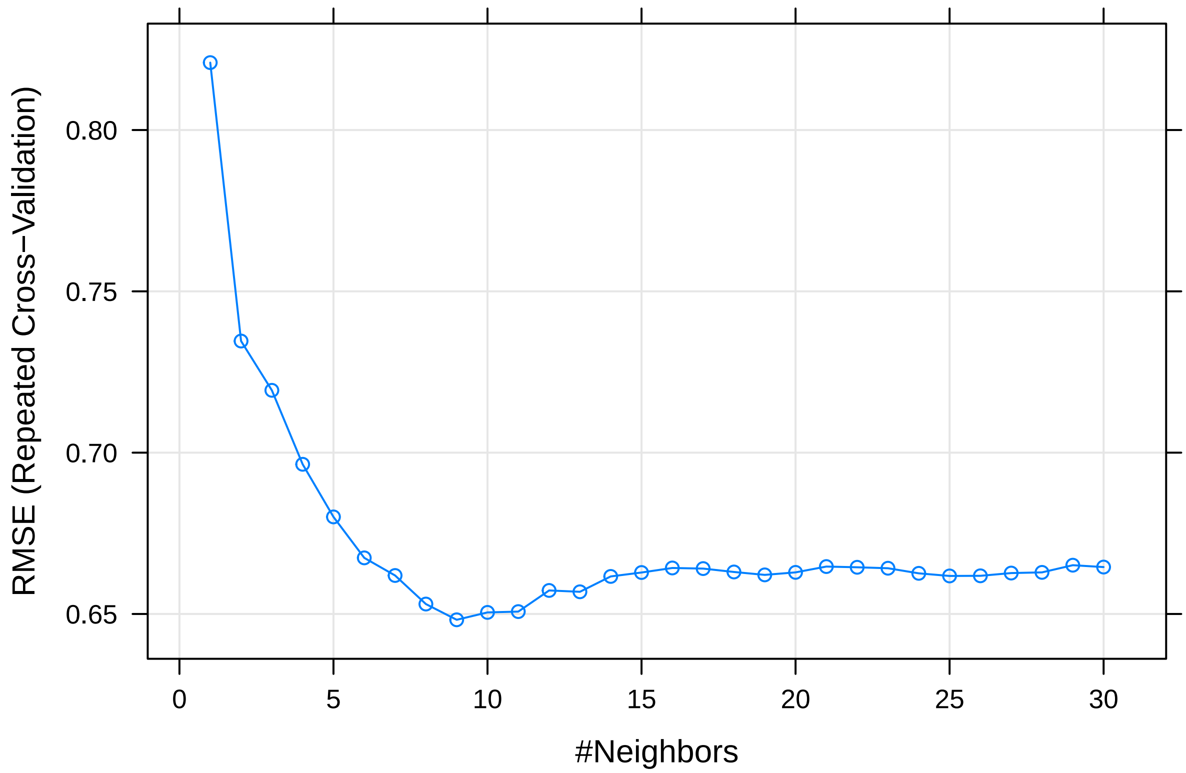
```r
knn_fit2
```

```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 373, 372, 373, 372, 373, 372, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.8208914  0.4589198  0.5720076
##    2  0.7345943  0.5021223  0.5430192
##    3  0.7193230  0.5160325  0.5171039
##    4  0.6963946  0.5349966  0.4990782
##    5  0.6800896  0.5496286  0.4889774
##    6  0.6673903  0.5645314  0.4826072
##    7  0.6619324  0.5706802  0.4783931
##    8  0.6530752  0.5823971  0.4717805
##    9  0.6482001  0.5897203  0.4690146
##   10  0.6504902  0.5870143  0.4717421
##   11  0.6507385  0.5859003  0.4733774
##   12  0.6572891  0.5784420  0.4751928
##   13  0.6568824  0.5789199  0.4741850
##   14  0.6616381  0.5727454  0.4760110
##   15  0.6628558  0.5711566  0.4777957
##   16  0.6642592  0.5693504  0.4791862
##   17  0.6640717  0.5702501  0.4781486
##   18  0.6630204  0.5712504  0.4771783
##   19  0.6621232  0.5727235  0.4767994
##   20  0.6628964  0.5723369  0.4774304
##   21  0.6646975  0.5705990  0.4804141
##   22  0.6644753  0.5717902  0.4807096
##   23  0.6641792  0.5727748  0.4798131
##   24  0.6626001  0.5747411  0.4780654
##   25  0.6617846  0.5763935  0.4781905
##   26  0.6618263  0.5764625  0.4785849
##   27  0.6626804  0.5760565  0.4789091
##   28  0.6628970  0.5762356  0.4791105
##   29  0.6651228  0.5735676  0.4814830
##   30  0.6645416  0.5750395  0.4817628
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 9.
```

```r
knn_fit2$bestTune
```

```
##   k
## 9 9
```

```r
plot(knn_fit2)
```



Este modelo contiene $x1$ $x4$ quienes son la fecha de transaccion y el numero de tienda de coveniencia:

```r
knn_fit3 <- train(y ~., data = datos[,c(1,4,5)], method = "knn",
                  trControl=trctrl,
                  preProcess = c("knnImpute"),
                  tuneGrid   = expand.grid(k = 1:30))


knn_fit3
```

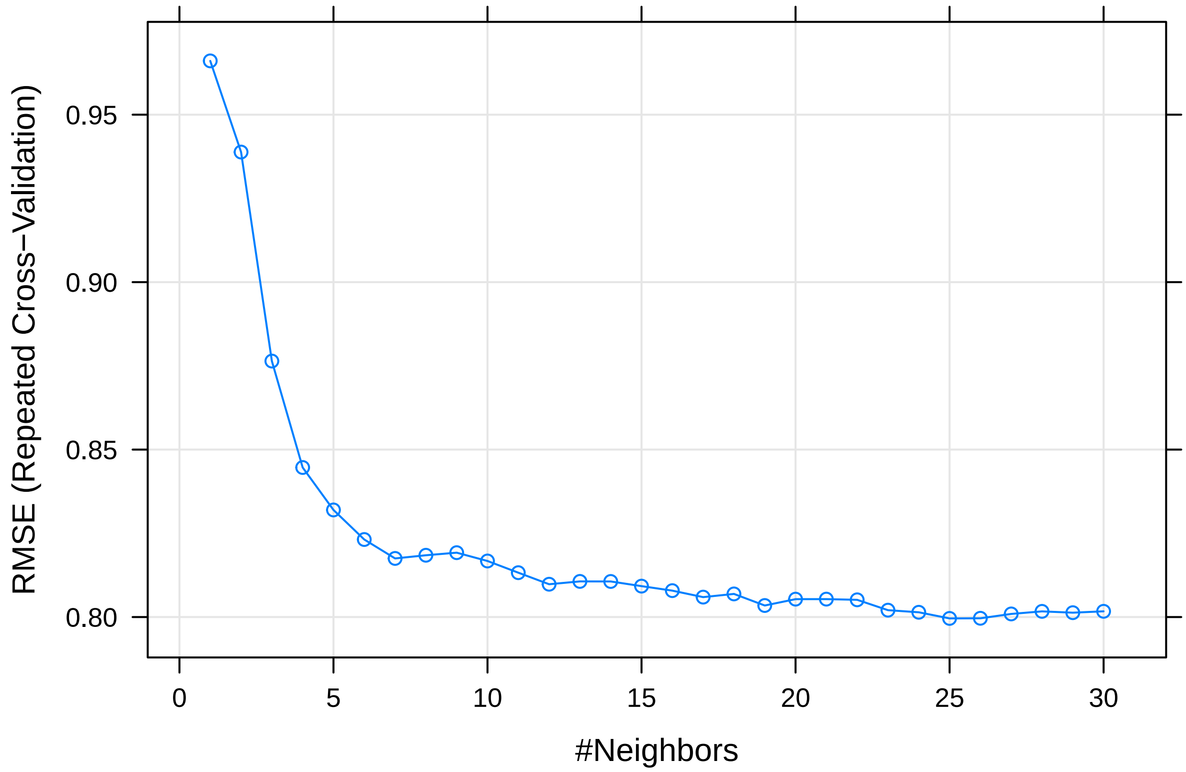
```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 372, 373, 372, 372, 374, 372, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.9660619  0.2027298  0.7197375
##    2  0.9388367  0.2222102  0.7029557
```

```
##    3  0.8764272  0.2817696  0.6604234
##    4  0.8446573  0.3176757  0.6324198
##    5  0.8319764  0.3331620  0.6222443
##    6  0.8231519  0.3420982  0.6151273
##    7  0.8175015  0.3492411  0.6134934
##    8  0.8184490  0.3460536  0.6146040
##    9  0.8192241  0.3442093  0.6140356
##   10  0.8167421  0.3475184  0.6114332
##   11  0.8132464  0.3521505  0.6089667
##   12  0.8097984  0.3564701  0.6062814
##   13  0.8106435  0.3543711  0.6063679
##   14  0.8106281  0.3538530  0.6078293
##   15  0.8092311  0.3554005  0.6066522
##   16  0.8078841  0.3574542  0.6059677
##   17  0.8059491  0.3614422  0.6051468
##   18  0.8069043  0.3602453  0.6053632
##   19  0.8034503  0.3646994  0.6024433
##   20  0.8053482  0.3620136  0.6046421
##   21  0.8053502  0.3627471  0.6051236
##   22  0.8051436  0.3629020  0.6068324
##   23  0.8020564  0.3678228  0.6044314
##   24  0.8014235  0.3690459  0.6047145
##   25  0.7995918  0.3718761  0.6032568
##   26  0.7996303  0.3722660  0.6031761
##   27  0.8009330  0.3702267  0.6038072
##   28  0.8016973  0.3690681  0.6032961
##   29  0.8013083  0.3701938  0.6016748
##   30  0.8017092  0.3695015  0.6012003
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 25.
```

```r
knn_fit3$bestTune
```

```
##     k
## 25 25
```

```r
plot(knn_fit3)
```



Este modelo contiene $x2$ $x3$, son la edad de la vivienda y la distancia a la estacion mas cercana:

```
knn_fit4 <- train(y ~., data = datos[,c(2,3,5)], method = "knn",
                  trControl=trctrl,
                  preProcess = c("knnImpute"),
                  tuneGrid   = expand.grid(k = 1:30))


knn_fit4
```

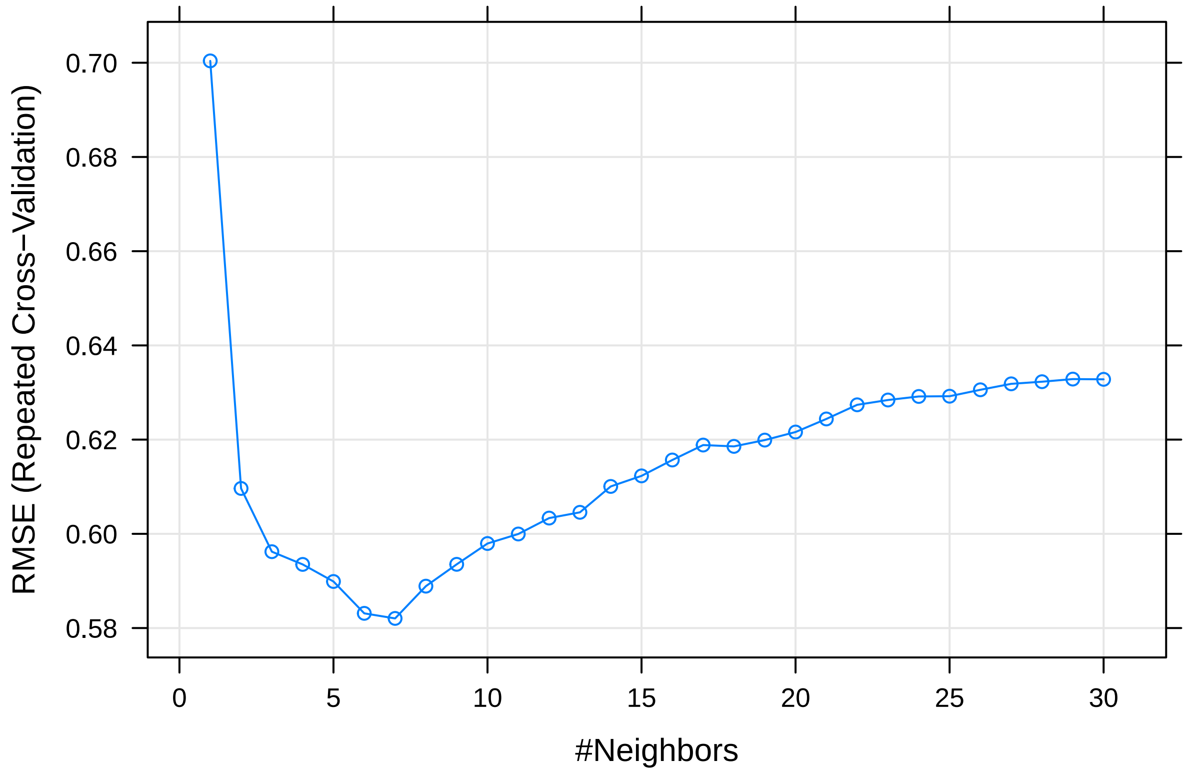
```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 374, 374, 372, 373, 373, 372, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.7003961  0.5923172  0.4465424
##    2  0.6096370  0.6519782  0.4069999
##    3  0.5962131  0.6651729  0.3960729
##    4  0.5935077  0.6640128  0.4014696
##    5  0.5898863  0.6651719  0.4043809
```

```
##    6  0.5831239  0.6674510  0.4040803
##    7  0.5820447  0.6656342  0.4040314
##    8  0.5888934  0.6561802  0.4085977
##    9  0.5935250  0.6492390  0.4107105
##   10  0.5979461  0.6437452  0.4165450
##   11  0.5999682  0.6407286  0.4181712
##   12  0.6033442  0.6361079  0.4195974
##   13  0.6045681  0.6338871  0.4205950
##   14  0.6100606  0.6275086  0.4229116
##   15  0.6123049  0.6235333  0.4246725
##   16  0.6156753  0.6190851  0.4262119
##   17  0.6188496  0.6156845  0.4288752
##   18  0.6185650  0.6161428  0.4300814
##   19  0.6198982  0.6138790  0.4311247
##   20  0.6216129  0.6113394  0.4332126
##   21  0.6243847  0.6082868  0.4346045
##   22  0.6273876  0.6047813  0.4359716
##   23  0.6284065  0.6036720  0.4358762
##   24  0.6291646  0.6028340  0.4362046
##   25  0.6292155  0.6026870  0.4366221
##   26  0.6305765  0.6013394  0.4376855
##   27  0.6318407  0.5998716  0.4390624
##   28  0.6322926  0.5995577  0.4398555
##   29  0.6328469  0.5989782  0.4421203
##   30  0.6327979  0.5992148  0.4428194
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 7.
```

```
knn_fit4$bestTune
```

```
##   k
## 7 7
```

```
plot(knn_fit4)
```



Este modelo contiene $x2$ $x4$, son edad de vivienda y tiendas de conveniencia:

```
knn_fit5 <- train(y ~., data = datos[,c(2,4,5)], method = "knn",
                  trControl=trctrl,
                  preProcess = c("knnImpute"),
                  tuneGrid   = expand.grid(k = 1:30))


knn_fit5
```

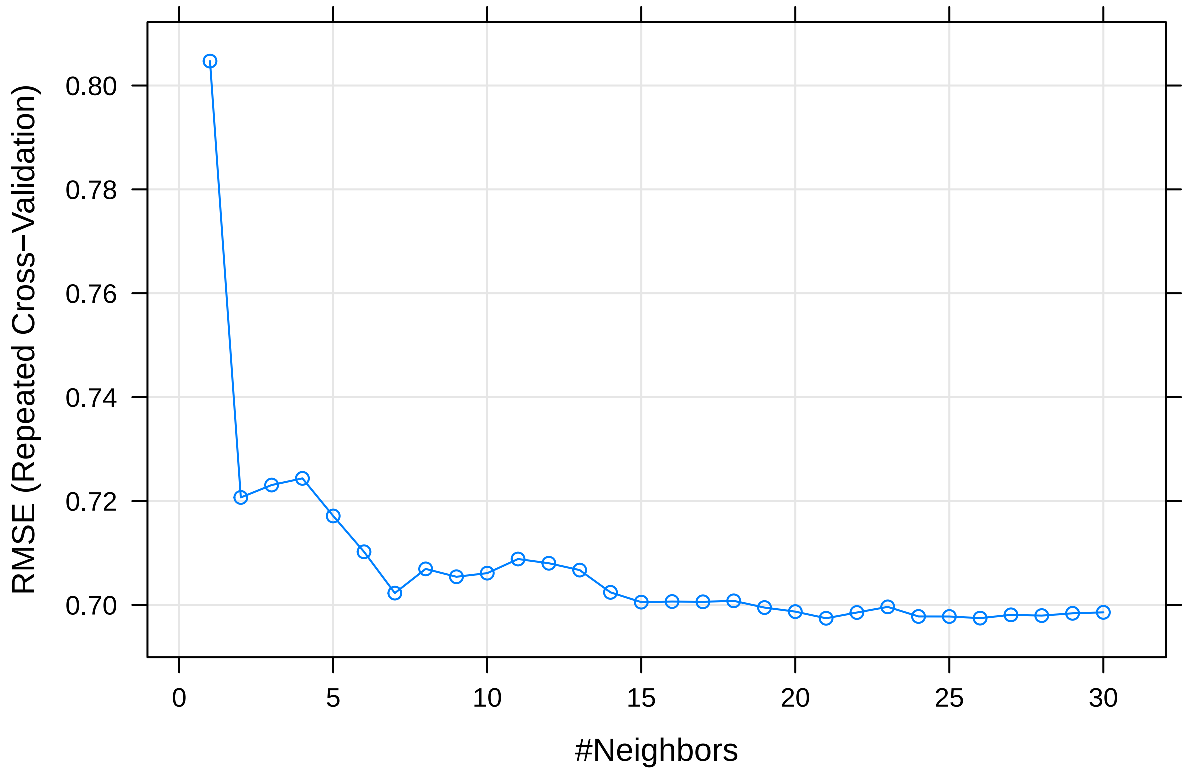
```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 372, 373, 372, 373, 374, 373, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.8046992  0.4824288  0.5338147
##    2  0.7206986  0.5229352  0.4998697
##    3  0.7230812  0.5089462  0.4952349
##    4  0.7243658  0.5050973  0.4976178
##    5  0.7171273  0.5112285  0.4947740
```

```
##     6  0.7102363  0.5153074  0.4931690
##     7  0.7022812  0.5207051  0.4913148
##     8  0.7069393  0.5143620  0.4979868
##     9  0.7054242  0.5143275  0.4996067
##    10  0.7061199  0.5120014  0.5055589
##    11  0.7088446  0.5051882  0.5105249
##    12  0.7080265  0.5059414  0.5115700
##    13  0.7067104  0.5072497  0.5119682
##    14  0.7024246  0.5128419  0.5115658
##    15  0.7005391  0.5145759  0.5124333
##    16  0.7006560  0.5143739  0.5130733
##    17  0.7006010  0.5135622  0.5134699
##    18  0.7007899  0.5138481  0.5125944
##    19  0.6994945  0.5160441  0.5122179
##    20  0.6987111  0.5171345  0.5110010
##    21  0.6974348  0.5189862  0.5102343
##    22  0.6985431  0.5174781  0.5128035
##    23  0.6996289  0.5157104  0.5159152
##    24  0.6977829  0.5174400  0.5143795
##    25  0.6977738  0.5171477  0.5147573
##    26  0.6974589  0.5174840  0.5160433
##    27  0.6980929  0.5166878  0.5166794
##    28  0.6979492  0.5168934  0.5164739
##    29  0.6983910  0.5163779  0.5172506
##    30  0.6985760  0.5162348  0.5165458
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 21.
```

```r
knn_fit5$bestTune[[1]]
```

```
## [1] 21
```

```r
plot(knn_fit5)
```



Este modelo contiene $x3$ $x4$, son distancia a la estacion mas cercana y tiendas de conveniencia:

```
knn_fit6 <- train(y ~., data = datos[,c(3,4,5)], method = "knn",
                  trControl=trctrl,
                  preProcess = c("knnImpute"),
                  tuneGrid   = expand.grid(k = 1:30))


knn_fit6
```

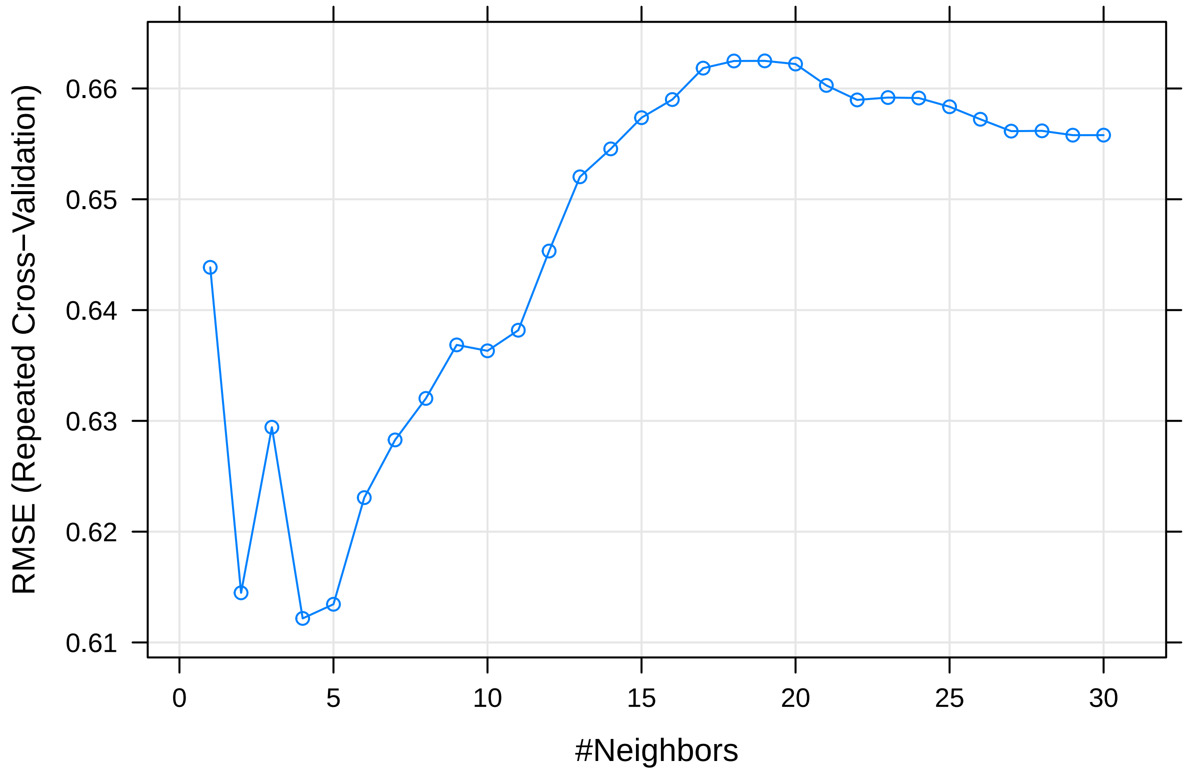
```
## k-Nearest Neighbors
##
## 414 samples
##   2 predictor
##
## Pre-processing: nearest neighbor imputation (2), centered (2), scaled (2)
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 371, 373, 373, 372, 373, 374, ...
## Resampling results across tuning parameters:
##
##   k   RMSE       Rsquared   MAE
##    1  0.6438572  0.6266206  0.4282798
##    2  0.6144796  0.6410555  0.4238987
##    3  0.6294274  0.6320706  0.4364345
##    4  0.6121704  0.6386983  0.4277522
##    5  0.6134482  0.6328392  0.4318894
```

```
##     6  0.6230720  0.6225839  0.4384840
##     7  0.6282790  0.6138274  0.4395733
##     8  0.6320272  0.6088688  0.4399700
##     9  0.6368575  0.6030175  0.4429335
##    10  0.6363213  0.6036942  0.4461481
##    11  0.6381801  0.6013204  0.4488945
##    12  0.6453298  0.5933731  0.4568469
##    13  0.6520277  0.5855765  0.4621357
##    14  0.6545476  0.5831158  0.4659117
##    15  0.6573638  0.5795695  0.4706675
##    16  0.6590008  0.5766167  0.4713372
##    17  0.6618355  0.5721211  0.4723184
##    18  0.6624811  0.5698650  0.4712220
##    19  0.6624901  0.5690370  0.4721208
##    20  0.6622069  0.5687157  0.4717360
##    21  0.6602767  0.5701817  0.4718005
##    22  0.6589669  0.5712952  0.4716133
##    23  0.6591826  0.5706176  0.4714329
##    24  0.6591373  0.5704569  0.4705542
##    25  0.6583500  0.5712221  0.4691656
##    26  0.6572205  0.5723334  0.4682430
##    27  0.6561507  0.5733595  0.4676599
##    28  0.6561770  0.5730757  0.4673684
##    29  0.6557867  0.5736591  0.4670912
##    30  0.6557876  0.5734837  0.4683361
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 4.
```

```
knn_fit6$bestTune
```

```
##   k
## 4 4
```

```
plot(knn_fit6)
```



Con esto se puede resumir el codigo hecho, primero se observa el par de variables, luego sus criterios para seleccion de varaibles:

```
for (i in list(knn_fit1,knn_fit2,knn_fit3,knn_fit4,knn_fit5,knn_fit6)){
  print(i$coefnames)
  print(i$results %>% filter(k==i$bestTune[[1]]))
}
```

```
## [1] "x1" "x2"
##    k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1 30 0.9129892 0.1770189 0.7330499 0.1240479 0.09593329 0.06469835
## [1] "x1" "x3"
##   k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1 9 0.6482001 0.5897203 0.4690146 0.1473283  0.1009299 0.07085401
## [1] "x1" "x4"
##    k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1 25 0.7995918 0.3718761 0.6032568 0.1473878  0.1310301 0.05807187
## [1] "x2" "x3"
##   k      RMSE  Rsquared       MAE    RMSESD RsquaredSD     MAESD
## 1 7 0.5820447 0.6656342 0.4040314 0.1626699  0.1265776 0.0689795
## [1] "x2" "x4"
##    k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1 21 0.6974348 0.5189862 0.5102343 0.1707513  0.1327048 0.04825945
## [1] "x3" "x4"
##   k      RMSE  Rsquared       MAE    RMSESD RsquaredSD      MAESD
## 1 4 0.6121704 0.6386983 0.4277522 0.1585103  0.1171136 0.07012208
```

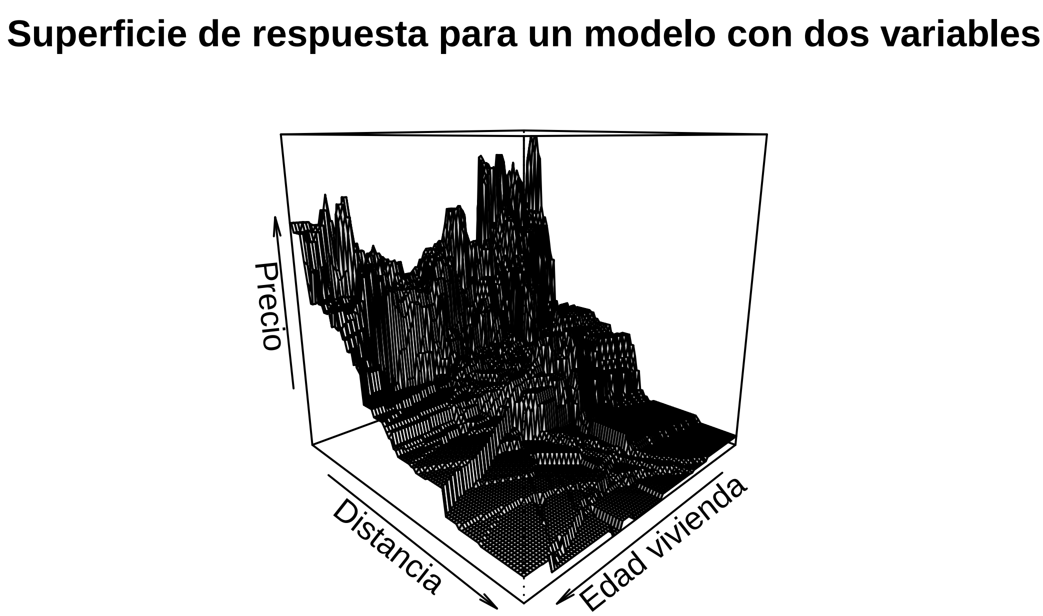
segun estos resultados, fijandonos primeramente en RMSE las variables que optimizan con menor resupesta son x2 y x3:

se grafica las superficie de respuesta

```
medias<-knn_fit4$preProcess$mean
dsv_es<-knn_fit4$preProcess$std

x2 <- seq(min(datos$x2), max(datos$x2), length.out = 100)
x3 <- seq(min(datos$x3), max(datos$x3), length.out = 100)
test.df<-expand.grid(x2,x3)
names(test.df)<-c("x2","x3")
test_pred <- predict(knn_fit4, newdata = test.df)
test.df$y <- test_pred
z <- matrix(test_pred,ncol=length(x3),nrow = length(x2))
```

```
persp(x2,x3,z,xlab="Edad vivienda",ylab="Distancia",zlab="Precio",
      main="Superficie de respuesta para un modelo con dos variables",theta=135,shade=0.3)
```



Con plotly se ve mejor:

```
#library(plotly)
#p <- plot_ly(x=x2,y=x3,z = z)
#p <- add_surface(p)
#p<-layout(p,title='Precio vs dis MRT y Edad de viviend',
#          xaxis=list(title="Edad"),
```

```r
#          yaxis=list(title="Distancia al MRT (m)"))
#p
#PAra pdf no imprime plotly por ello se comenta
```

Aqui se observa donde es valida la superficie de respuesta:

```r
ggplot(datos_completos, aes(x=x2, y=x3)) +
  geom_point()+
  labs(title="Antiguedad vs Distancia", x= 'Edad de Vivienda', y = "Distancia")+
  theme_light()
```