KNN

Salvador Enrique Rodríguez Hernández

28/06/2025

1. Carga los datos en el entorno de Rstudio a través de la función readRDS. Utilizando el código que consideres (y los datos disponibles), indica qué precio estimarías que tiene una vivienda.

```
library(caret)
library(pROC)
library(kknn)
library(iml)
# Dado que la validación cruzada puede aprovechar los beneficios de la
# paralelización, lo ponemos (si da problemas en tu ordenador, puedes obviarlo)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
```

```
datosRegr<-readRDS("VentaViviendas")
summary(datosRegr)
```

```
##      price           bathrooms         superf        garden         floors
##  Min.   :  86500   Min.   :0.000   Min.   : 34   Min.   :0.00   Min.   :1.000
##  1st Qu.: 324938   1st Qu.:1.500   1st Qu.:131   1st Qu.:0.03   1st Qu.:1.000
##  Median : 451000   Median :2.500   Median :177   Median :0.05   Median :2.000
##  Mean   : 544129   Mean   :2.139   Mean   :193   Mean   :0.12   Mean   :1.545
##  3rd Qu.: 645000   3rd Qu.:2.500   3rd Qu.:238   3rd Qu.:0.08   3rd Qu.:2.000
##  Max.   :7062500   Max.   :6.000   Max.   :933   Max.   :8.15   Max.   :4.000
##  view     condition renovated      lat             long           antig
##  0:4509   1: 415    0:4784    Min.   :47.16   Min.   :-122.5   Min.   :  6.00
##  1: 491   2:3278    1: 216    1st Qu.:47.48   1st Qu.:-122.3   1st Qu.: 25.00
##           3:1307              Median :47.57   Median :-122.2   Median : 46.00
##                               Mean   :47.56   Mean   :-122.2   Mean   : 49.92
##                               3rd Qu.:47.68   3rd Qu.:-122.1   3rd Qu.: 70.00
##                               Max.   :47.78   Max.   :-121.4   Max.   :121.00
```

2. Realiza una partición del conjunto de datos en entrenamiento (80%) y prueba (20%).

```
set.seed(12345)
trainIndex <- createDataPartition(datosRegr$price, p=0.8, list=FALSE)
data_rg_train <- datosRegr[trainIndex,]
data_rg_test <- datosRegr[-trainIndex,]
```

3. Genera un primer modelo KNN para la variable price con un k igual a 5 y la distancia de Manhattan. Imprime las matrices D y CL, explica qué contienen y relaciónalo con el funcionamiento del modelo KNN. Así mismo, calcula el R2 prueba.

```
modelo1 <- kknn(price~., data_rg_train, data_rg_test, distance = 1, k=5, kernel = "rectangular")
head(modelo1$D)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]
## [1,] 1.0159330 1.0204036 1.1885006 1.1935816 1.2006315
## [2,] 0.4941885 0.6414835 0.9367808 1.0379852 1.4961745
## [3,] 0.5885818 0.6380380 0.7138115 0.8191423 0.8582983
## [4,] 1.4197728 1.4778340 1.6673854 2.1345592 2.4026542
## [5,] 1.0594933 1.3604033 1.6154392 1.9123710 1.9340495
## [6,] 0.3151480 0.5881992 0.6588125 0.6773996 0.6954126
```

```
head(modelo1$CL)
```

```
##         [,1]    [,2]    [,3]   [,4]    [,5]
## [1,]  855000  601000  560000 497000  570000
## [2,]  306500  350000  346950 240000  347500
## [3,] 1430800 1260000 1035000 800000  822500
## [4,]  425000  600000  550000 475000  249000
## [5,]  700000 1500000  630000 750000 1125000
## [6,]  626000  455000  419900 409900  551000
```

```
R2(modelo1$fitted.values,data_rg_test$price)
```

```
## [1] 0.7130166
```

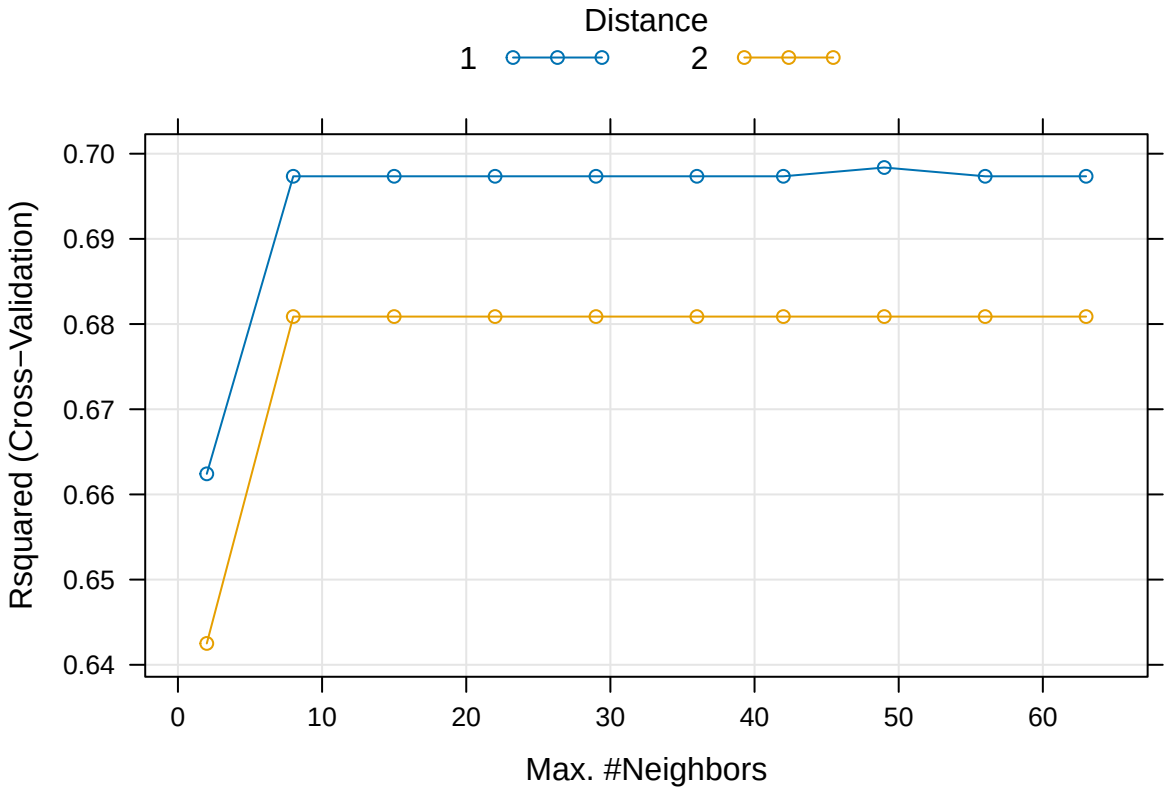
4. Utilizando validación cruzada, haz un análisis preliminar del k y tipo de distancia óptimo. Comenta los resultados y determina la "mejor" combinación, justificando porqué consideras que es la mejor.

```
set.seed(12345)
# En la posición 1 está la variable objetivo
knn_tuneTodo <- train(y=data_rg_train$price, x = data_rg_train[,-1],
                  method = "kknn",
                  trControl = trainControl(method="cv", number = 5),
                  metric="Rsquared",
                  tuneGrid = expand.grid(kmax=floor(seq.int(2,sqrt(nrow(data_rg_train)),length.out=10))
knn_tuneTodo
```

```
## k-Nearest Neighbors
##
## 4001 samples
##   10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 3201, 3200, 3201, 3202, 3200
## Resampling results across tuning parameters:
##
##   kmax  distance  RMSE      Rsquared   MAE
##    2    1         222944.5  0.6624179  118074.5
##    2    2         230009.5  0.6425056  121997.3
##    8    1         213516.8  0.6973485  110034.6
```

```
##    8     2         217861.1  0.6808780  114547.3
##   15     1         213516.8  0.6973485  110034.6
##   15     2         217861.1  0.6808780  114547.3
##   22     1         213516.8  0.6973485  110034.6
##   22     2         217861.1  0.6808780  114547.3
##   29     1         213516.8  0.6973485  110034.6
##   29     2         217861.1  0.6808780  114547.3
##   36     1         213516.8  0.6973485  110034.6
##   36     2         217861.1  0.6808780  114547.3
##   42     1         213516.8  0.6973485  110034.6
##   42     2         217861.1  0.6808780  114547.3
##   49     1         213181.0  0.6983776  110061.9
##   49     2         217861.1  0.6808780  114547.3
##   56     1         213516.8  0.6973485  110034.6
##   56     2         217861.1  0.6808780  114547.3
##   63     1         213516.8  0.6973485  110034.6
##   63     2         217861.1  0.6808780  114547.3
##
## Tuning parameter 'kernel' was held constant at a value of rectangular
## Rsquared was used to select the optimal model using the largest value.
## The final values used for the model were kmax = 49, distance = 1 and kernel
##  = rectangular.
```

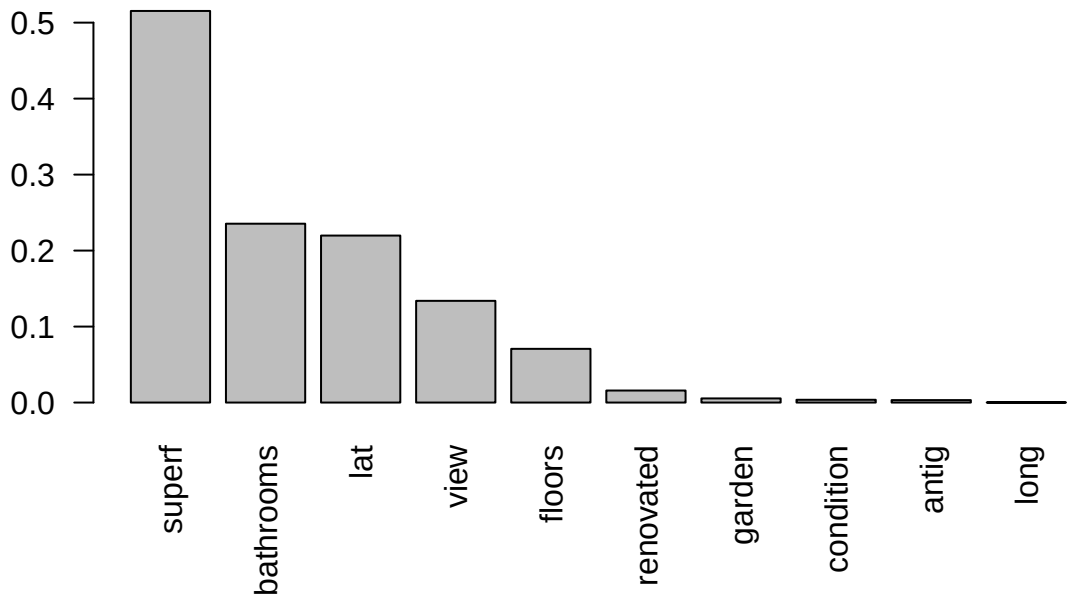
```r
plot(knn_tuneTodo,metric=c("Rsquared"))
```



5. Utilizando esa combinación óptima, lleva a cabo una selección de variables. De nuevo, comenta los resultados y concluye cuál es el conjunto óptimo de variables input.

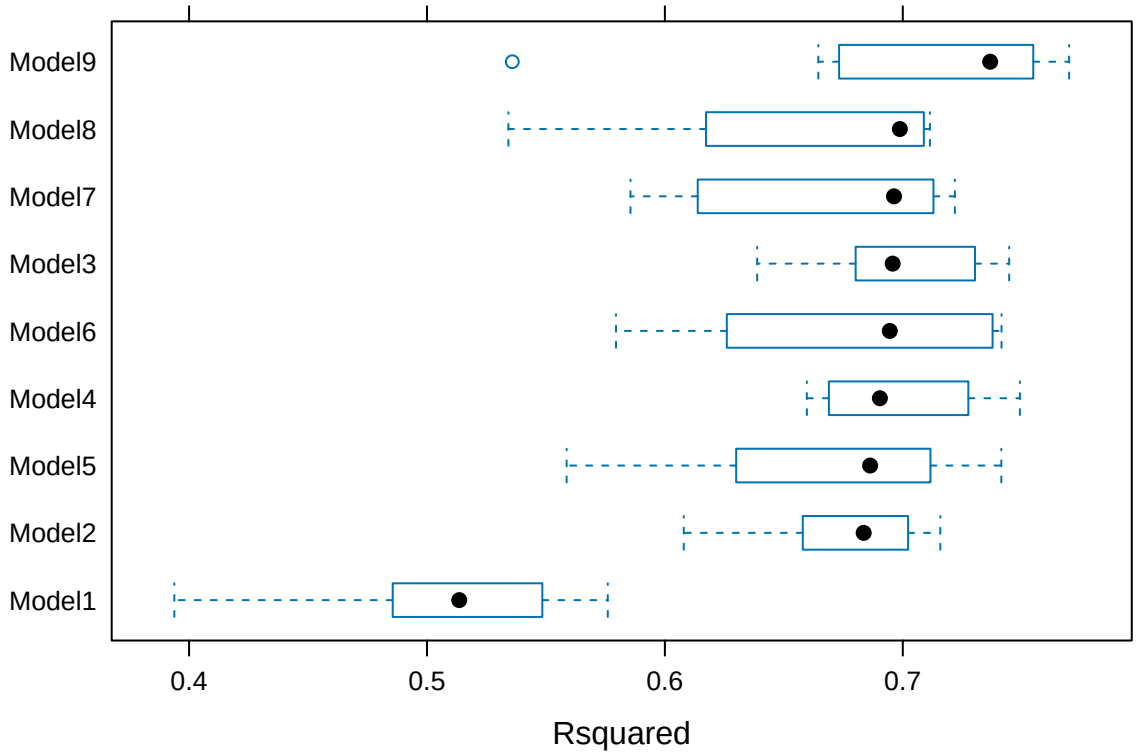
```r
# El 1 es la posición en la que se ubica la variable IMC
salida<-filterVarImp(x = data_rg_train[,-1], y = data_rg_train$price, nonpara = TRUE)
ranking<-sort(apply(salida, 1, mean), decreasing =T)

# Para ajustar el margen inferior del gráfico y que así quepan los nombres
par(mar=c(8.1, 4.1, 4.1, 2.1))
barplot(ranking, las=2)
```




```r
# Vuelvo a poner los márgenes por defecto
par(mar=c(5.1, 4.1, 4.1, 2.1))
```

```r
vcrTodosModelos<-list()
for (i in 1:(length(ranking)-1)){
set.seed(12345)
vcrTodosModelos[[i]] <- train(y=data_rg_train$price, x = data_rg_train[,names(ranking)[1:(i+1)]],
                  method = "kknn",
                  trControl = trainControl(method="cv", number = 8),
                  metric="Rsquared",
                  tuneGrid = expand.grid(kmax=6, distance=1, kernel = "rectangular"))
}
bwplot(resamples(vcrTodosModelos),metric=c("Rsquared"))
```




```
summary(resamples(vcrTodosModelos),metric=c("Rsquared"))
```

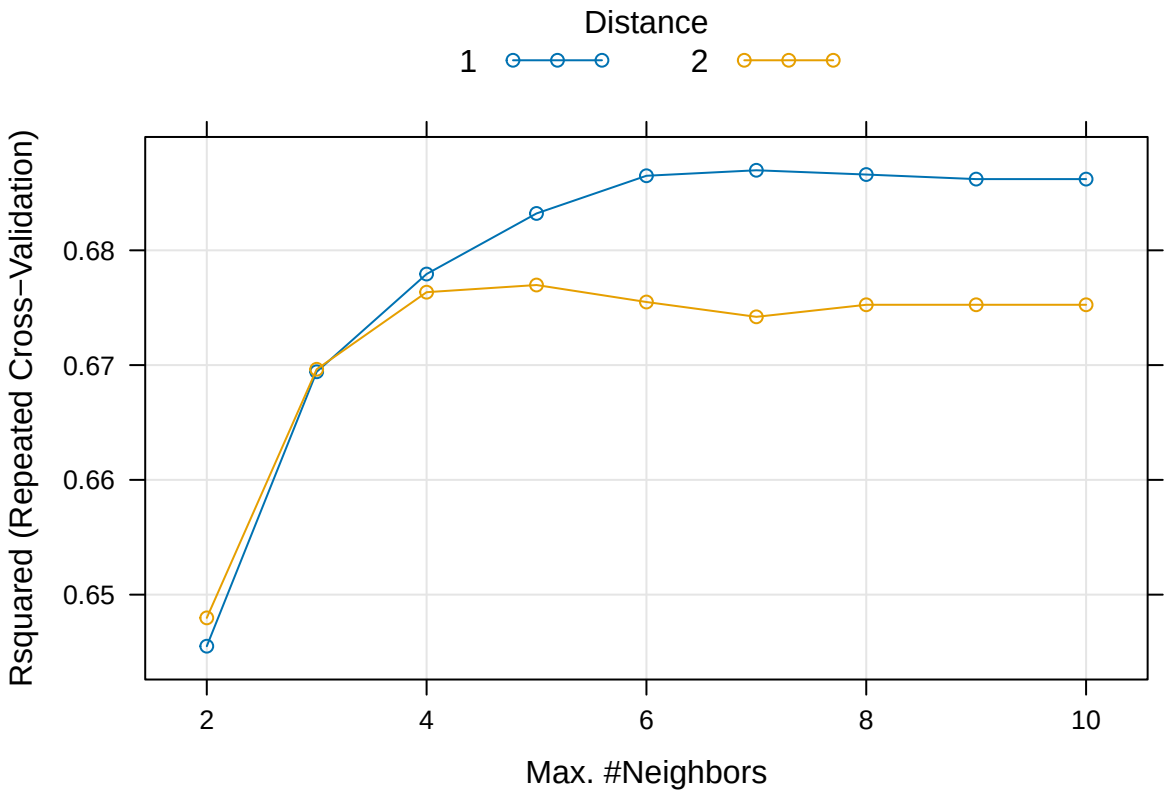
```
##
## Call:
## summary.resamples(object = resamples(vcrTodosModelos), metric = c("Rsquared"))
##
## Models: Model1, Model2, Model3, Model4, Model5, Model6, Model7, Model8, Model9
## Number of resamples: 8
##
## Rsquared
##             Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Model1 0.3937857 0.4869405 0.5135644 0.5081330 0.5445810 0.5760014    0
## Model2 0.6079289 0.6682547 0.6835646 0.6764004 0.6994903 0.7157802    0
## Model3 0.6388028 0.6803820 0.6957397 0.6995126 0.7277967 0.7447152    0
## Model4 0.6596770 0.6703139 0.6903756 0.6978275 0.7267586 0.7492114    0
## Model5 0.5586846 0.6417299 0.6862741 0.6694829 0.7050341 0.7414223    0
## Model6 0.5794648 0.6311607 0.6945366 0.6797029 0.7366018 0.7415172    0
## Model7 0.5855307 0.6216317 0.6963023 0.6691807 0.7110243 0.7218963    0
## Model8 0.5342151 0.6289245 0.6987735 0.6619284 0.7085742 0.7114060    0
## Model9 0.5358592 0.6775891 0.7367357 0.7044255 0.7508372 0.7699200    0
```

6. Para la combinación óptima de variables, busca el valor óptimo de k (cerrando el conjunto de valores teniendo en cuenta los resultados del apartado 4) y del tipo de distancia. Comenta los resultados e indica el resultado.

```r
set.seed(12345)
knn_finetune <- train(y=data_rg_train$price, x = data_rg_train[,names(ranking)[1:(9+1)]],
                      method = "kknn",
                      trControl = trainControl(method="repeatedcv", number = 5, repeats=10),
                      tuneGrid = expand.grid(kmax=floor(seq.int(2,10,length.out=10)), distance=c(1,2), kern
knn_finetune
```

```
## k-Nearest Neighbors
##
## 4001 samples
##   10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 10 times)
## Summary of sample sizes: 3201, 3200, 3201, 3202, 3200, 3200, ...
## Resampling results across tuning parameters:
##
##   kmax  distance  RMSE      Rsquared   MAE
##    2    1         225694.3  0.6455079  117869.6
##    2    2         224845.8  0.6479761  121332.4
##    3    1         217454.7  0.6694174  113311.8
##    3    2         217517.6  0.6696376  116905.3
##    4    1         215478.6  0.6779345  111477.2
##    4    2         215851.9  0.6763531  115340.7
##    5    1         214484.9  0.6832146  111024.4
##    5    2         216320.2  0.6769778  114729.3
##    6    1         213808.0  0.6864969  110419.0
##    6    2         217232.5  0.6754977  114486.4
##    7    1         213897.6  0.6869754  110262.4
##    7    2         217743.8  0.6742061  114419.8
##    8    1         214271.2  0.6866040  110223.2
##    8    2         217367.0  0.6752561  114239.9
##    9    1         214391.1  0.6862068  110251.1
##    9    2         217367.0  0.6752561  114239.9
##   10    1         214391.1  0.6862068  110251.1
##   10    2         217367.0  0.6752561  114239.9
##
## Tuning parameter 'kernel' was held constant at a value of rectangular
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were kmax = 6, distance = 1 and kernel
##  = rectangular.
```

```r
plot(knn_finetune,metric=c("Rsquared"))
```

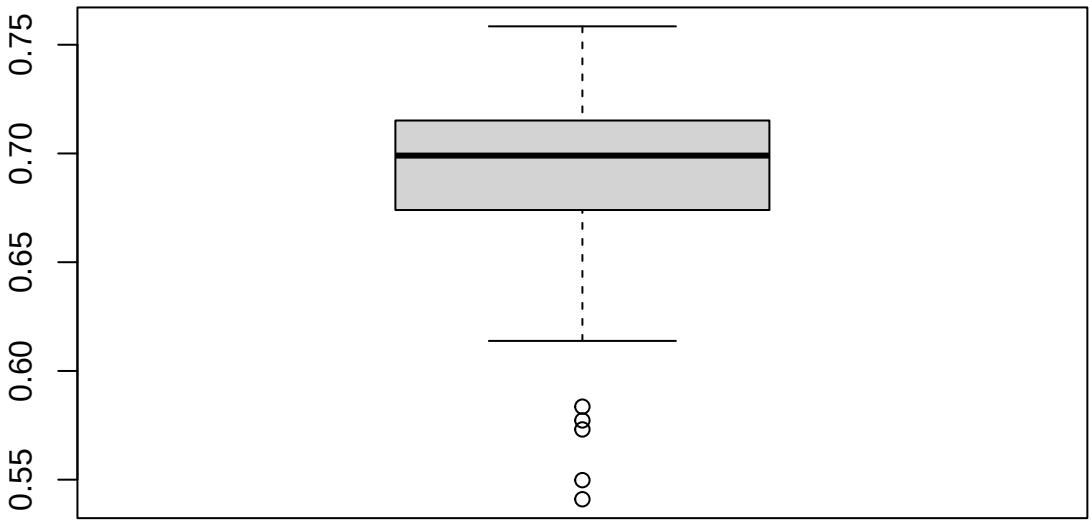


7. Construye el mejor modelo y calcula el R2 prueba. Usando la información de la validación cruzada anterior, comenta si se trata de un modelo estable.

```
modeloFinal <- kknn(price~., data_rg_train, data_rg_test, distance = 1, k=6, kernel = "rectangular")
R2(modeloFinal$fitted.values,data_rg_test$price)
```

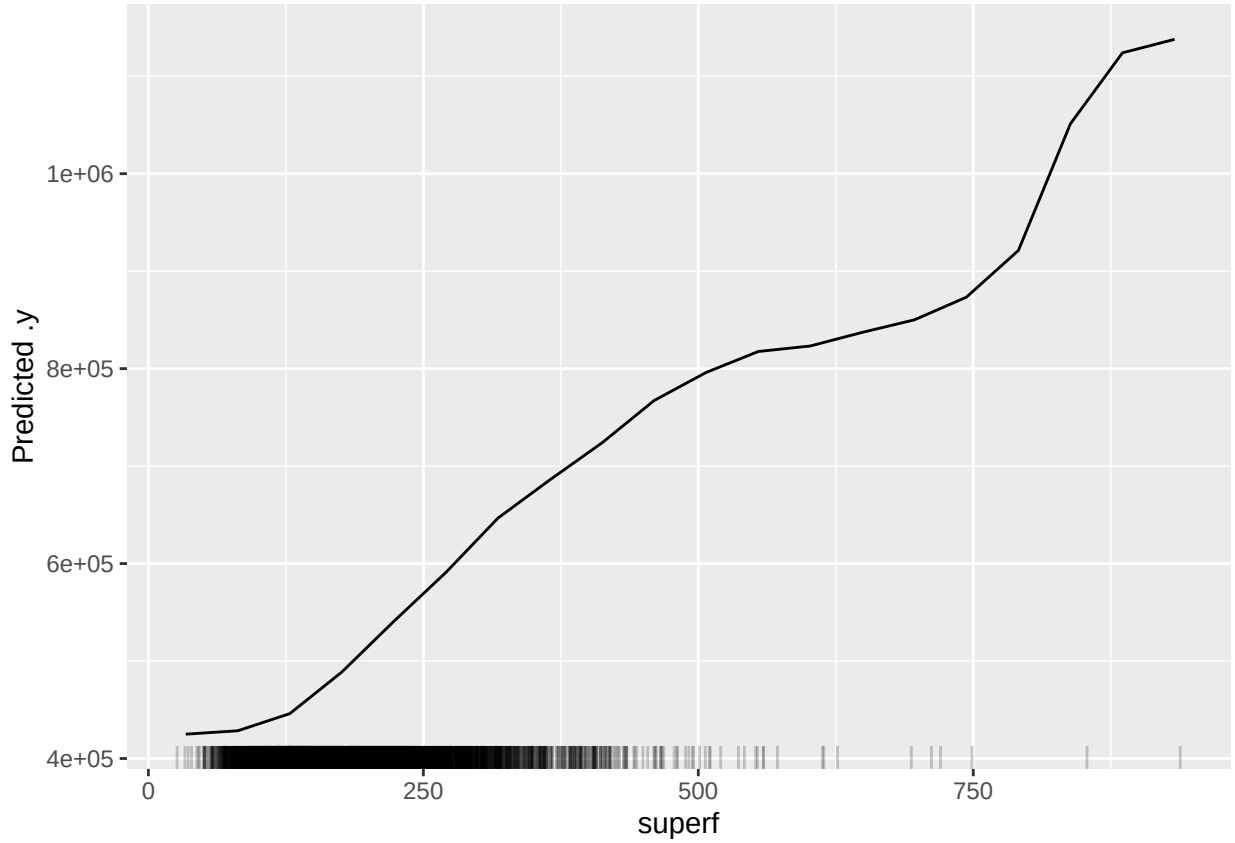
```
## [1] 0.7354059
```

```
boxplot(knn_finetune$resample$Rsquared)
```

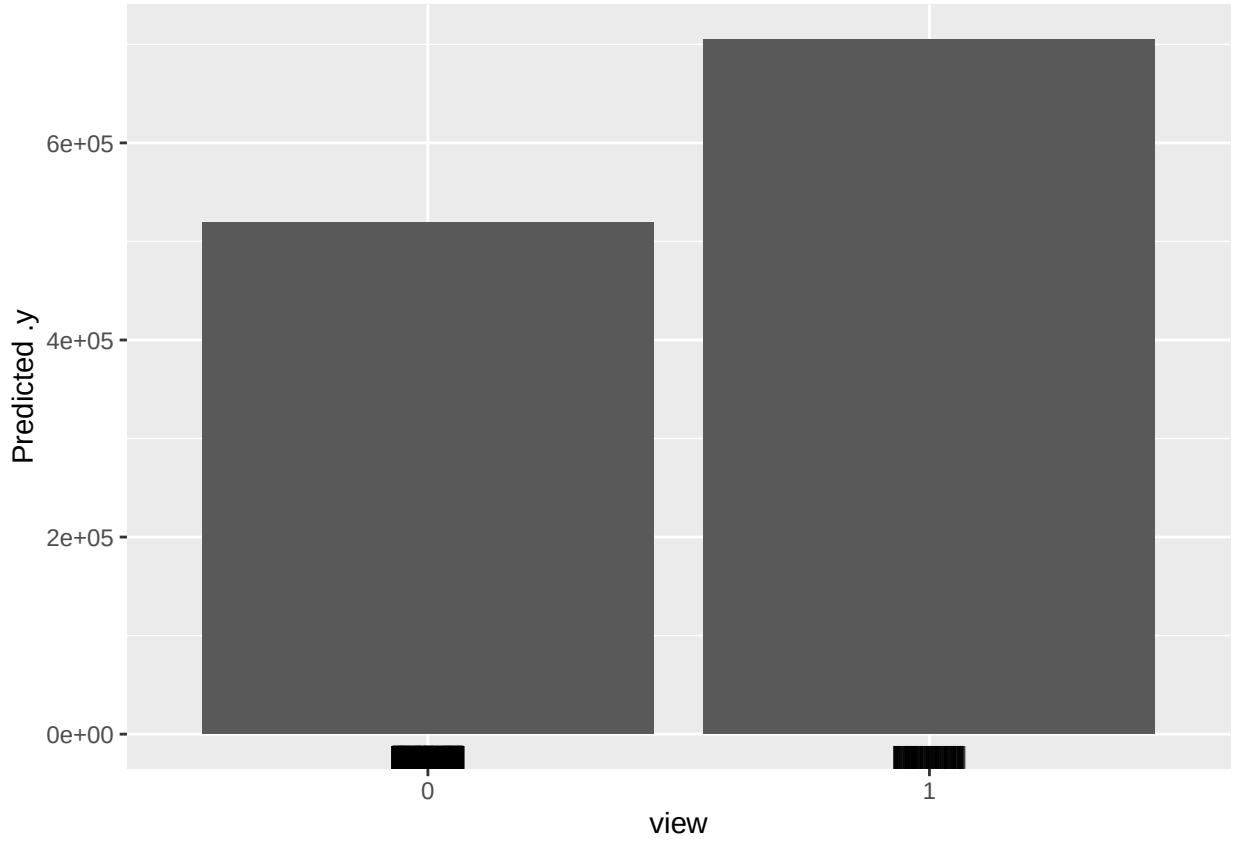


8. Obtén el PDP para dos variables (una cuantitativa y una cualitativa), juntas y por separado. Comenta los resultados.

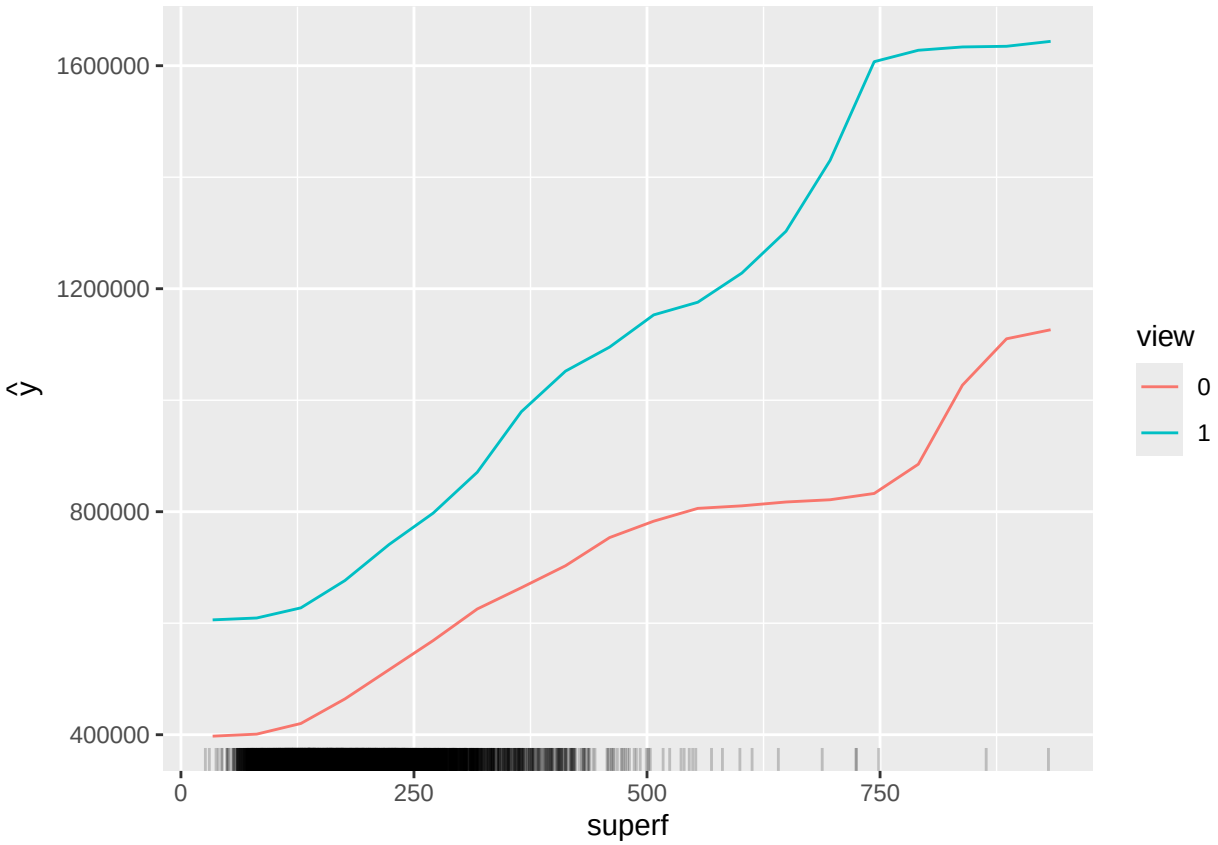
```
predictor <- Predictor$new(knn_finetune, data = data_rg_train, y = data_rg_train$price)
pdp <- FeatureEffect$new(predictor, feature = "superf", method="pdp")
pdp$plot()
```




```
pdp2 <- FeatureEffect$new(predictor, feature = "view", method="pdp")
pdp2$plot()
```




```
pdp3 <- FeatureEffect$new(predictor, feature = c("superf","view"), method="pdp")
pdp3$plot()
```


1600000
1200000
800000
400000
<y
0
250
500
750
superf
view
0
1