SVM

Cristopher Barrios, Carlos Daniel Estrada

2023-04-21

```
library(e1071)
library(caret)
library(corrplot)
library(labelled)
library(plotly)
library(ggplot2)
```

1. Use los mismos conjuntos de entrenamiento y prueba de las hojas de trabajo pasadas para probar el algoritmo.

```
set.seed(123)
train = read.csv("./train.csv")
```

2. Explore los datos y explique las transformaciones que debe hacerle para generar un modelo de máquinas vectoriales de soporte.

```
train[is.na(train)] <- 0
train$tipoDeCasa = as.numeric(as.character( cut(train$SalePrice,c(0,145000,205000,410000), labels = c(1
train[sapply(train, is.character)] <- lapply(train[sapply(train, is.character)],as.factor)

#columnas con NA
completeFun <- function(data, desiredCols) {
  completeVec <- complete.cases(data[, desiredCols])
  return(data[completeVec, ])
}
train <- completeFun(train, "tipoDeCasa") #variable respuesta, variable categorica
str(train)
```

```
## 'data.frame':    1436 obs. of  82 variables:
##  $ Id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ MSSubClass   : int  60 20 60 70 60 50 20 60 50 190 ...
##  $ MSZoning     : Factor w/ 5 levels "C (all)","FV",..: 4 4 4 4 4 4 4 4 5 4 ...
##  $ LotFrontage  : num  65 80 68 60 84 85 75 0 51 50 ...
##  $ LotArea      : int  8450 9600 11250 9550 14260 14115 10084 10382 6120 7420 ...
##  $ Street       : Factor w/ 2 levels "Grvl","Pave": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Alley        : Factor w/ 3 levels "0","Grvl","Pave": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotShape     : Factor w/ 4 levels "IR1","IR2","IR3",..: 4 4 1 1 1 1 4 1 4 4 ...
##  $ LandContour  : Factor w/ 4 levels "Bnk","HLS","Low",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ Utilities    : Factor w/ 2 levels "AllPub","NoSeWa": 1 1 1 1 1 1 1 1 1 1 ...
##  $ LotConfig    : Factor w/ 5 levels "Corner","CulDSac",..: 5 3 5 1 3 5 5 1 5 1 ...
```

```
##  $ LandSlope    : Factor w/ 3 levels "Gtl","Mod","Sev": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Neighborhood : Factor w/ 25 levels "Blmngtn","Blueste",..: 6 25 6 7 14 12 21 17 18 4 ...
##  $ Condition1   : Factor w/ 9 levels "Artery","Feedr",..: 3 2 3 3 3 3 3 5 1 1 ...
##  $ Condition2   : Factor w/ 8 levels "Artery","Feedr",..: 3 3 3 3 3 3 3 3 3 1 ...
##  $ BldgType     : Factor w/ 5 levels "1Fam","2fmCon",..: 1 1 1 1 1 1 1 1 1 2 ...
##  $ HouseStyle   : Factor w/ 8 levels "1.5Fin","1.5Unf",..: 6 3 6 6 6 1 3 6 1 2 ...
##  $ OverallQual  : int  7 6 7 7 8 5 8 7 7 5 ...
##  $ OverallCond  : int  5 8 5 5 5 5 5 6 5 6 ...
##  $ YearBuilt    : int  2003 1976 2001 1915 2000 1993 2004 1973 1931 1939 ...
##  $ YearRemodAdd : int  2003 1976 2002 1970 2000 1995 2005 1973 1950 1950 ...
##  $ RoofStyle    : Factor w/ 6 levels "Flat","Gable",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ RoofMatl     : Factor w/ 8 levels "ClyTile","CompShg",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ Exterior1st  : Factor w/ 15 levels "AsbShng","AsphShn",..: 13 9 13 14 13 13 13 7 4 9 ...
##  $ Exterior2nd  : Factor w/ 16 levels "AsbShng","AsphShn",..: 14 9 14 16 14 14 14 7 16 9 ...
##  $ MasVnrType   : Factor w/ 5 levels "0","BrkCmn","BrkFace",..: 3 4 3 4 3 4 5 5 4 4 ...
##  $ MasVnrArea   : num  196 0 162 0 350 0 186 240 0 0 ...
##  $ ExterQual    : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 4 3 4 3 4 4 4 ...
##  $ ExterCond    : Factor w/ 5 levels "Ex","Fa","Gd",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ Foundation   : Factor w/ 6 levels "BrkTil","CBlock",..: 3 2 3 1 3 6 3 2 1 1 ...
##  $ BsmtQual     : Factor w/ 5 levels "0","Ex","Fa",..: 4 4 4 5 4 4 2 4 5 5 ...
##  $ BsmtCond     : Factor w/ 5 levels "0","Fa","Gd",..: 5 5 5 3 5 5 5 5 5 5 ...
##  $ BsmtExposure : Factor w/ 5 levels "0","Av","Gd",..: 5 3 4 5 2 5 2 4 5 5 ...
##  $ BsmtFinType1 : Factor w/ 7 levels "0","ALQ","BLQ",..: 4 2 4 2 4 4 4 2 7 4 ...
##  $ BsmtFinSF1   : int  706 978 486 216 655 732 1369 859 0 851 ...
##  $ BsmtFinType2 : Factor w/ 7 levels "0","ALQ","BLQ",..: 7 7 7 7 7 7 7 3 7 7 ...
##  $ BsmtFinSF2   : int  0 0 0 0 0 0 0 32 0 0 ...
##  $ BsmtUnfSF    : int  150 284 434 540 490 64 317 216 952 140 ...
##  $ TotalBsmtSF  : int  856 1262 920 756 1145 796 1686 1107 952 991 ...
##  $ Heating      : Factor w/ 6 levels "Floor","GasA",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ HeatingQC    : Factor w/ 5 levels "Ex","Fa","Gd",..: 1 1 1 3 1 1 1 1 3 1 ...
##  $ CentralAir   : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...
##  $ Electrical   : Factor w/ 6 levels "0","FuseA","FuseF",..: 6 6 6 6 6 6 6 6 3 6 ...
##  $ X1stFlrSF    : int  856 1262 920 961 1145 796 1694 1107 1022 1077 ...
##  $ X2ndFlrSF    : int  854 0 866 756 1053 566 0 983 752 0 ...
##  $ LowQualFinSF : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ GrLivArea    : int  1710 1262 1786 1717 2198 1362 1694 2090 1774 1077 ...
##  $ BsmtFullBath : int  1 0 1 1 1 1 1 1 0 1 ...
##  $ BsmtHalfBath : int  0 1 0 0 0 0 0 0 0 0 ...
##  $ FullBath     : int  2 2 2 1 2 1 2 2 2 1 ...
##  $ HalfBath     : int  1 0 1 0 1 1 0 1 0 0 ...
##  $ BedroomAbvGr : int  3 3 3 3 4 1 3 3 2 2 ...
##  $ KitchenAbvGr : int  1 1 1 1 1 1 1 1 2 2 ...
##  $ KitchenQual  : Factor w/ 4 levels "Ex","Fa","Gd",..: 3 4 3 3 3 4 3 4 4 4 ...
##  $ TotRmsAbvGrd : int  8 6 6 7 9 5 7 7 8 5 ...
##  $ Functional   : Factor w/ 7 levels "Maj1","Maj2",..: 7 7 7 7 7 7 7 7 3 7 ...
##  $ Fireplaces   : int  0 1 1 1 1 0 1 2 2 2 ...
##  $ FireplaceQu  : Factor w/ 6 levels "0","Ex","Fa",..: 1 6 6 4 6 1 4 6 6 6 ...
##  $ GarageType   : Factor w/ 7 levels "0","2Types","Attchd",..: 3 3 3 7 3 3 3 3 7 3 ...
##  $ GarageYrBlt  : num  2003 1976 2001 1998 2000 ...
##  $ GarageFinish : Factor w/ 4 levels "0","Fin","RFn",..: 3 3 3 4 3 4 3 3 4 3 ...
##  $ GarageCars   : int  2 2 2 3 3 2 2 2 2 1 ...
##  $ GarageArea   : int  548 460 608 642 836 480 636 484 468 205 ...
##  $ GarageQual   : Factor w/ 6 levels "0","Ex","Fa",..: 6 6 6 6 6 6 6 6 3 4 ...
##  $ GarageCond   : Factor w/ 6 levels "0","Ex","Fa",..: 6 6 6 6 6 6 6 6 6 6 ...
```

```
##  $ PavedDrive   : Factor w/ 3 levels "N","P","Y": 3 3 3 3 3 3 3 3 3 3 ...
##  $ WoodDeckSF   : int  0 298 0 0 192 40 255 235 90 0 ...
##  $ OpenPorchSF  : int  61 0 42 35 84 30 57 204 0 4 ...
##  $ EnclosedPorch: int  0 0 0 272 0 0 0 228 205 0 ...
##  $ X3SsnPorch   : int  0 0 0 0 0 320 0 0 0 0 ...
##  $ ScreenPorch  : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolArea     : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PoolQC       : Factor w/ 4 levels "0","Ex","Fa",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ Fence        : Factor w/ 5 levels "0","GdPrv","GdWo",..: 1 1 1 1 1 4 1 1 1 1 ...
##  $ MiscFeature  : Factor w/ 5 levels "0","Gar2","Othr",..: 1 1 1 1 1 4 1 4 1 1 ...
##  $ MiscVal      : int  0 0 0 0 0 700 0 350 0 0 ...
##  $ MoSold       : int  2 5 9 2 12 10 8 11 4 1 ...
##  $ YrSold       : int  2008 2007 2008 2006 2008 2009 2007 2009 2008 2008 ...
##  $ SaleType     : Factor w/ 9 levels "COD","Con","ConLD",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ SaleCondition: Factor w/ 6 levels "Abnorml","AdjLand",..: 5 5 5 1 5 5 5 5 1 5 ...
##  $ SalePrice    : int  208500 181500 223500 140000 250000 143000 307000 200000 129900 118000 ...
##  $ tipoDeCasa   : num  3 2 3 1 3 1 3 2 1 1 ...
```

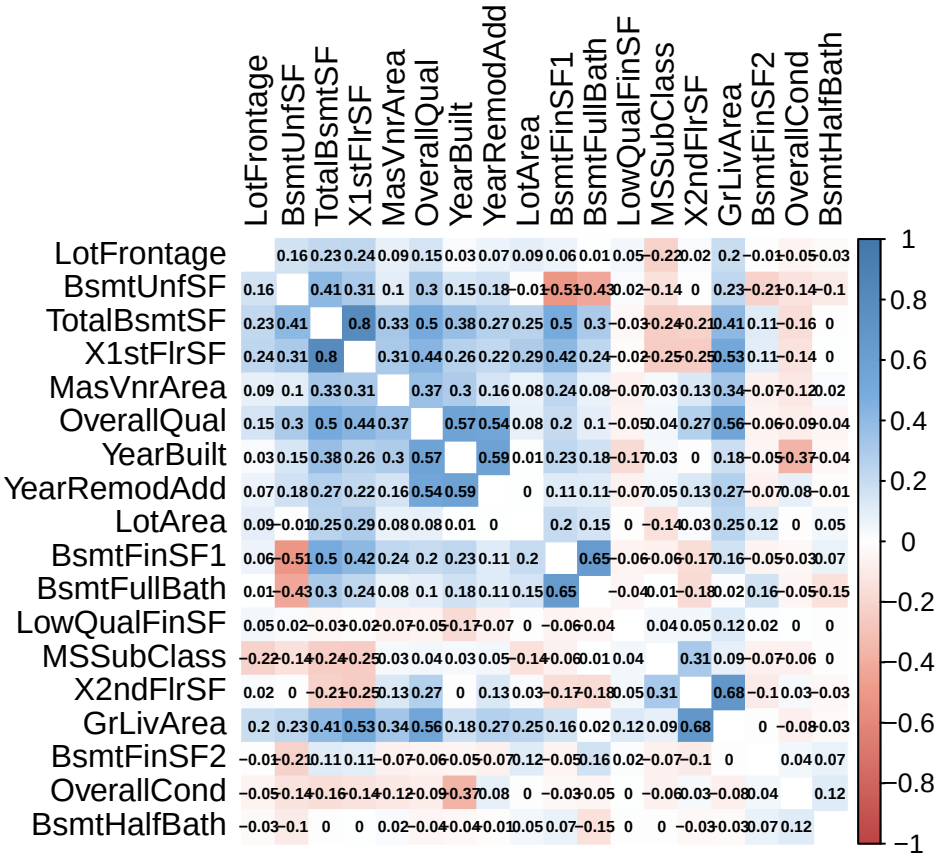
3. Use como variable respuesta la variable categórica que especifica si la casa es barata, media o cara

```
train <- completeFun(train, "tipoDeCasa") #variable respuesta, variable categorica

#datos con factor
frstselect <- train[,c(2:5,8,9,11:43,46,49:54,56:62,64:72,76:80,82)]

#datos cuantitativos
scndselect <- subset (train, select = c(2,4,5,18,19,20,21,27,35,37,38,39,44,45,46,47,48,49,50,51,52,53,5
scndselect[is.na(scndselect)] <- 0
```

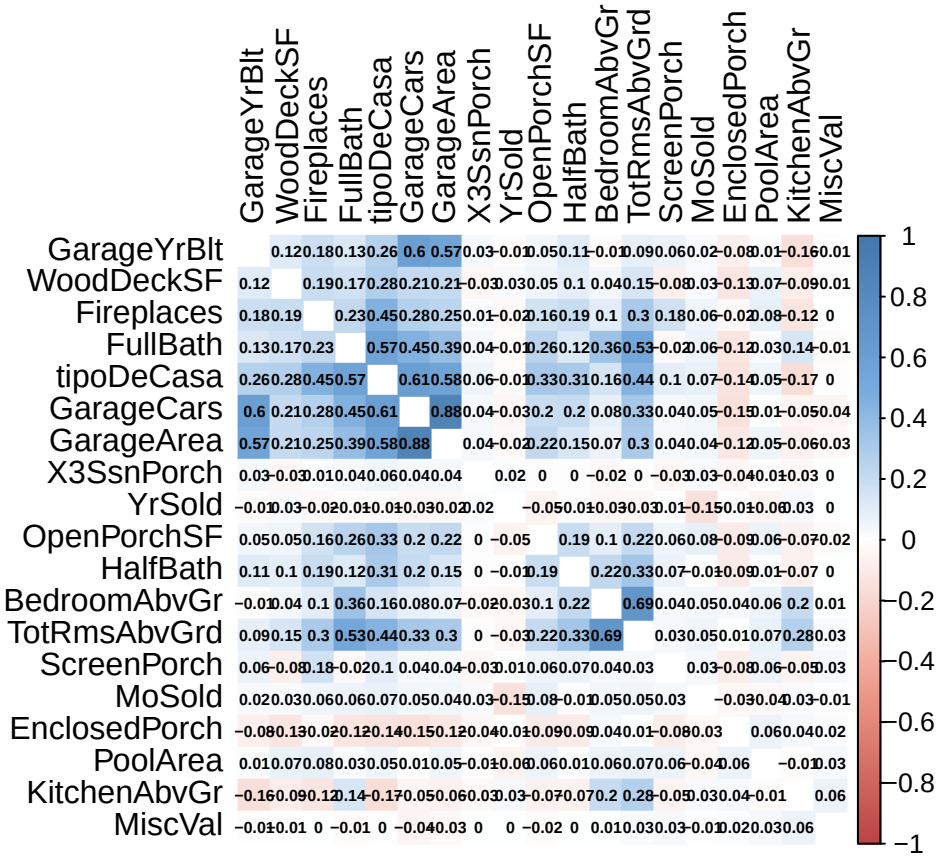
4. Genere varios (más de 2) modelos de SVM con diferentes kernels y distintos valores en los parámetros c, gamma (circular) y d (en caso de que utilice el polinomial). Puede tunear el modelo de forma automática siempre que explique los resultados.

```
M <- cor(scndselect[,c(1:18)])
M1<- cor(scndselect[,c(19:37)])
M2<- cor(scndselect[,c(1:18)],scndselect[,c(19:37)])
col <- colorRampPalette(c("#BB4444", "#EE9988", "#FFFFFF", "#77AADD", "#4477AA"))
```

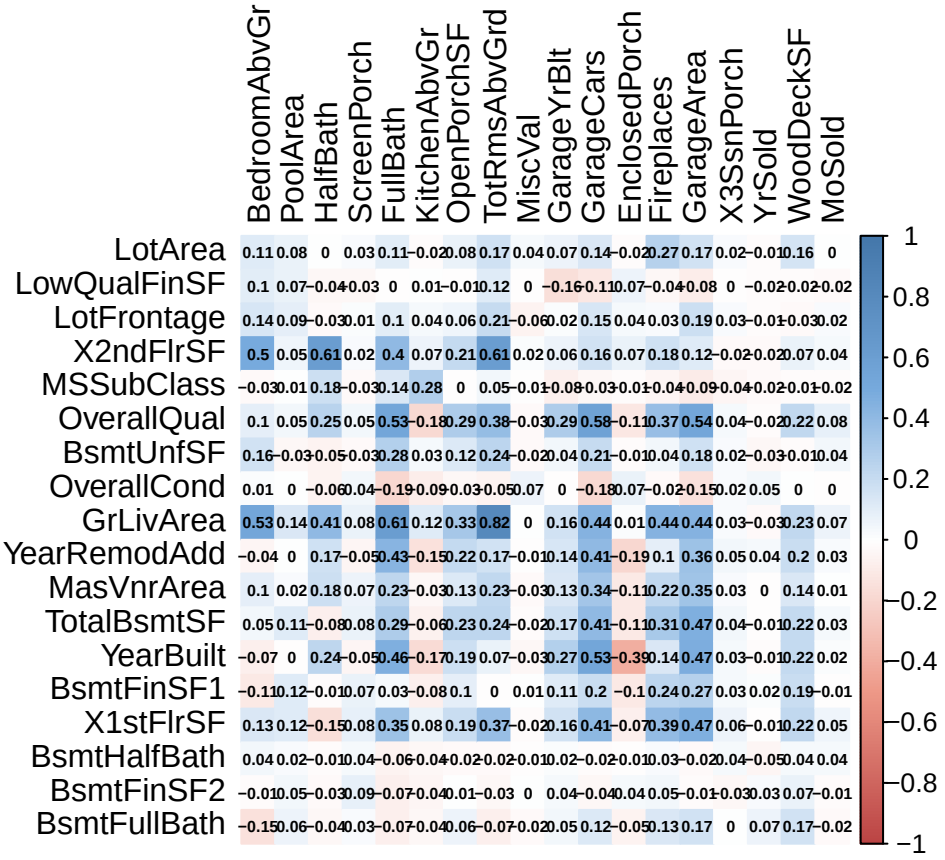
```
corrplot(M,  method = "color", col = col(200), order = "hclust", number.cex = .5,
         addCoef.col = "black",
         tl.col = "black",
         sig.level = 0.50, insig = "blank",
         diag = FALSE)
```




```
corrplot(M1,  method = "color", col = col(200), order = "hclust", number.cex = .5,
         addCoef.col = "black",
         tl.col = "black",
         sig.level = 0.50, insig = "blank",
         diag = FALSE)
```




```
corrplot(M2,  method = "color", col = col(200), order = "hclust", number.cex = .5,
         addCoef.col = "black",
         tl.col = "black",
         sig.level = 0.50, insig = "blank",
         diag = TRUE)
```




```
tipocor<- cor(scndselect[,-1],scndselect$tipoDeCasa)
tipocor
```

```
##                      [,1]
## LotFrontage    0.124781325
## LotArea        0.219555016
## OverallQual    0.738892019
## OverallCond   -0.095483076
## YearBuilt      0.565329689
## YearRemodAdd   0.544774124
## MasVnrArea     0.344254632
## BsmtFinSF1     0.276950433
## BsmtFinSF2     0.012463641
## BsmtUnfSF      0.208466092
## TotalBsmtSF    0.513208477
## X1stFlrSF      0.496997346
## X2ndFlrSF      0.297500739
## LowQualFinSF  -0.068654379
## GrLivArea      0.622251690
## BsmtFullBath   0.189408729
## BsmtHalfBath  -0.035329884
## FullBath       0.574967912
## HalfBath       0.307309514
## BedroomAbvGr   0.157624456
## KitchenAbvGr  -0.167361574
```

```
## TotRmsAbvGrd   0.441616356
## Fireplaces     0.454122445
## GarageYrBlt    0.262710019
## GarageCars     0.610605411
## GarageArea     0.575708954
## WoodDeckSF     0.281222474
## OpenPorchSF    0.326152417
## EnclosedPorch -0.138763225
## X3SsnPorch     0.059689959
## ScreenPorch    0.099859557
## PoolArea       0.051525222
## MiscVal       -0.001375927
## MoSold         0.070906122
## YrSold        -0.012373046
## tipoDeCasa     1.000000000
```

```
porciento <- 70/100

#todo tipo de variables
trainRowsNumber<-sample(1:nrow(frstselect),porciento*nrow(frstselect))
train<-frstselect[trainRowsNumber,]
test<-frstselect[-trainRowsNumber,]

#variables cuantitativas
trainRowsNum<-sample(1:nrow(scndselect),porciento*nrow(scndselect))
train1<-scndselect[trainRowsNum,]
test1<-scndselect[-trainRowsNum,]

#Modelos
modeloSVM_L1<-svm(tipoDeCasa~., data=train,type="C-classification", cost=2^5, kernel="linear")
modeloSVM_L2<-svm(tipoDeCasa~., data=train,type="C-classification", cost=0.5, kernel="linear")
modeloSVM_L3<-svm(tipoDeCasa~., data=train,type="C-classification", cost=2^-5, kernel="linear")


modeloSVM_R1<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=0.005,kernel="radial")
modeloSVM_R2<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=0.05,kernel="radial")
modeloSVM_R3<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=2^-5,kernel="radial")


modeloSVM_P1<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=1, kernel="polynomial", coef0=
modeloSVM_P2<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=5, kernel="polynomial", coef0=
modeloSVM_P3<-svm(tipoDeCasa~., data=train,type="C-classification", gamma=2^-5, kernel="polynomial", co
```

```
summary(modeloSVM_L1)
```

```
##
## Call:
## svm(formula = tipoDeCasa ~ ., data = train, type = "C-classification",
##     cost = 2^5, kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  C-classification
```

```
##  SVM-Kernel:  linear
##        cost:  32
##
## Number of Support Vectors:  337
##
##  ( 91 153 93 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

```
summary(modeloSVM_R1)
```

```
##
## Call:
## svm(formula = tipoDeCasa ~ ., data = train, type = "C-classification",
##     gamma = 0.005, kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  641
##
##  ( 164 305 172 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

```
summary(modeloSVM_P1)
```

```
##
## Call:
## svm(formula = tipoDeCasa ~ ., data = train, type = "C-classification",
##     gamma = 1, kernel = "polynomial", coef0 = 1, degree = 8)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  1
##      degree:  8
##      coef.0:  1
##
## Number of Support Vectors:  826
##
```

```
##  ( 175 325 326 )
##
##
## Number of Classes:  3
##
## Levels:
##  1 2 3
```

5. Use los modelos para predecir el valor de la variable respuesta

```
# Linear
process_timeL1 <- proc.time()
prediccionL1<-predict(modeloSVM_L1,newdata=test[,1:67])
process_timeL1 <- proc.time() - process_timeL1
process_timeL2 <- proc.time()
prediccionL2<-predict(modeloSVM_L2,newdata=test[,1:67])
process_timeL2 <- proc.time() - process_timeL2
process_timeL3 <- proc.time()
prediccionL3<-predict(modeloSVM_L3,newdata=test[,1:67])
process_timeL3 <- proc.time() - process_timeL3
process_timeL_avarage <- (process_timeL1[3] + process_timeL2[3] + process_timeL3[3])/3

# Radial
process_timeR1 <- proc.time()
prediccionR1<-predict(modeloSVM_R1,newdata=test[,1:67])#[,1:37]
process_timeR1 <- proc.time() - process_timeR1
process_timeR2 <- proc.time()
prediccionR2<-predict(modeloSVM_R2,newdata=test[,1:67])#[,1:37]
process_timeR2 <- proc.time() - process_timeR2
process_timeR3 <- proc.time()
prediccionR3<-predict(modeloSVM_R3,newdata=test[,1:67])#[,1:37]
process_timeR3 <- proc.time() - process_timeR3
process_timeR_avarage <- (process_timeR1[3] + process_timeR2[3] + process_timeR3[3])/3

# Polinomial
process_timeP1 <- proc.time()
prediccionP1<-predict(modeloSVM_P1,newdata=test[,1:67])
process_timeP1 <- proc.time() - process_timeP1
process_timeP2 <- proc.time()
prediccionP2<-predict(modeloSVM_P2,newdata=test[,1:67])
process_timeP2 <- proc.time() - process_timeP2
process_timeP3 <- proc.time()
prediccionP3<-predict(modeloSVM_P3,newdata=test[,1:67])
process_timeP3 <- proc.time() - process_timeP3
process_timeP_avarage <- (process_timeP1[3] + process_timeP2[3] + process_timeP3[3])/3

#Cambio de tipo de data a factors
test$tipoDeCasa<- as.factor(test$tipoDeCasa)
test1$tipoDeCasa<- as.factor(test$tipoDeCasa)
```

6. Haga las matrices de confusión respectivas.

```r
#linear
cmL1<-confusionMatrix(test$tipoDeCasa,prediccionL1)
cmL2<-confusionMatrix(test$tipoDeCasa,prediccionL2)
cmL3<-confusionMatrix(test$tipoDeCasa,prediccionL3)
```

```r
#linear
cmL1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 128  29   1
##          2  24 104  22
##          3   1  33  89
##
## Overall Statistics
##
##                Accuracy : 0.7448
##                  95% CI : (0.7009, 0.7853)
##     No Information Rate : 0.3852
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6143
##
##  Mcnemar's Test P-Value : 0.4451
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8366   0.6265   0.7946
## Specificity            0.8921   0.8264   0.8934
## Pos Pred Value         0.8101   0.6933   0.7236
## Neg Pred Value         0.9084   0.7794   0.9253
## Prevalence             0.3550   0.3852   0.2599
## Detection Rate         0.2970   0.2413   0.2065
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.8643   0.7265   0.8440
```

```r
#linear
cmL2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 140  18   0
##          2  22 106  22
##          3   1  24  98
##
## Overall Statistics
##
##                Accuracy : 0.7981
```

```
##                  95% CI : (0.7571, 0.835)
##     No Information Rate : 0.3782
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6953
##
##  Mcnemar's Test P-Value : 0.6853
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8589   0.7162   0.8167
## Specificity            0.9328   0.8445   0.9196
## Pos Pred Value         0.8861   0.7067   0.7967
## Neg Pred Value         0.9158   0.8505   0.9286
## Prevalence             0.3782   0.3434   0.2784
## Detection Rate         0.3248   0.2459   0.2274
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.8959   0.7804   0.8681
```

```
#linear
cmL3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 147  11   0
##          2  19 115  16
##          3   0  23 100
##
## Overall Statistics
##
##                Accuracy : 0.8399
##                  95% CI : (0.8018, 0.8733)
##     No Information Rate : 0.3852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7581
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8855   0.7718   0.8621
## Specificity            0.9585   0.8759   0.9270
## Pos Pred Value         0.9304   0.7667   0.8130
## Neg Pred Value         0.9304   0.8790   0.9481
## Prevalence             0.3852   0.3457   0.2691
## Detection Rate         0.3411   0.2668   0.2320
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9220   0.8238   0.8945
```

```r
#radial
cmR1<-confusionMatrix(test$tipoDeCasa,prediccionR1)
cmR2<-confusionMatrix(test$tipoDeCasa,prediccionR2)
cmR3<-confusionMatrix(test$tipoDeCasa,prediccionR3)
```

```r
#radial
cmR1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 149   9   0
##          2  23 117  10
##          3   1  32  90
##
## Overall Statistics
##
##                Accuracy : 0.826
##                  95% CI : (0.7868, 0.8606)
##     No Information Rate : 0.4014
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.736
##
##  Mcnemar's Test P-Value : 0.0003231
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8613   0.7405   0.9000
## Specificity            0.9651   0.8791   0.9003
## Pos Pred Value         0.9430   0.7800   0.7317
## Neg Pred Value         0.9121   0.8541   0.9675
## Prevalence             0.4014   0.3666   0.2320
## Detection Rate         0.3457   0.2715   0.2088
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9132   0.8098   0.9002
```

```r
#radial
cmR2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 143  15   0
##          2  23 119   8
##          3   2  37  84
##
## Overall Statistics
##
##                Accuracy : 0.8028
```

```
##                  95% CI : (0.762, 0.8393)
##     No Information Rate : 0.3968
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7003
##
##  Mcnemar's Test P-Value : 5.455e-05
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8512   0.6959   0.9130
## Specificity            0.9430   0.8808   0.8850
## Pos Pred Value         0.9051   0.7933   0.6829
## Neg Pred Value         0.9084   0.8149   0.9740
## Prevalence             0.3898   0.3968   0.2135
## Detection Rate         0.3318   0.2761   0.1949
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.8971   0.7883   0.8990
```

```
#radial
cmR3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 145  13   0
##          2  19 122   9
##          3   0  31  92
##
## Overall Statistics
##
##                Accuracy : 0.8329
##                  95% CI : (0.7943, 0.8669)
##     No Information Rate : 0.3852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7467
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8841   0.7349   0.9109
## Specificity            0.9513   0.8943   0.9061
## Pos Pred Value         0.9177   0.8133   0.7480
## Neg Pred Value         0.9304   0.8434   0.9708
## Prevalence             0.3805   0.3852   0.2343
## Detection Rate         0.3364   0.2831   0.2135
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9177   0.8146   0.9085
```

```r
# Polinomial
cmP1<-confusionMatrix(test$tipoDeCasa,prediccionP1)
cmP2<-confusionMatrix(test$tipoDeCasa,prediccionP2)
cmP3<-confusionMatrix(test$tipoDeCasa,prediccionP3)
```

```r
# Polinomial
cmP1
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 145  13   0
##          2  23 120   7
##          3   1  46  76
##
## Overall Statistics
##
##                Accuracy : 0.7912
##                  95% CI : (0.7497, 0.8286)
##     No Information Rate : 0.4153
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.682
##
##  Mcnemar's Test P-Value : 4.154e-07
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8580   0.6704   0.9157
## Specificity            0.9504   0.8810   0.8649
## Pos Pred Value         0.9177   0.8000   0.6179
## Neg Pred Value         0.9121   0.7900   0.9773
## Prevalence             0.3921   0.4153   0.1926
## Detection Rate         0.3364   0.2784   0.1763
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9042   0.7757   0.8903
```

```r
# Polinomial
cmP2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 143  15   0
##          2  23 108  19
##          3   0  23 100
##
## Overall Statistics
##
##                Accuracy : 0.8144
```

```
##                  95% CI : (0.7744, 0.85)
##     No Information Rate : 0.3852
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7197
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8614   0.7397   0.8403
## Specificity            0.9434   0.8526   0.9263
## Pos Pred Value         0.9051   0.7200   0.8130
## Neg Pred Value         0.9158   0.8648   0.9383
## Prevalence             0.3852   0.3387   0.2761
## Detection Rate         0.3318   0.2506   0.2320
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9024   0.7962   0.8833
```

```
# Polinomial
cmP3
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   1   2   3
##          1 140  18   0
##          2  21 111  18
##          3   0  20 103
##
## Overall Statistics
##
##                Accuracy : 0.8213
##                  95% CI : (0.7819, 0.8564)
##     No Information Rate : 0.3735
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7304
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity            0.8696   0.7450   0.8512
## Specificity            0.9333   0.8617   0.9355
## Pos Pred Value         0.8861   0.7400   0.8374
## Neg Pred Value         0.9231   0.8648   0.9416
## Prevalence             0.3735   0.3457   0.2807
## Detection Rate         0.3248   0.2575   0.2390
## Detection Prevalence   0.3666   0.3480   0.2854
## Balanced Accuracy      0.9014   0.8033   0.8934
```

7. Analice si los modelos están sobreajustados o desajustados. ¿Qué puede hacer para manejar el sobreajuste o desajuste?

8. Compare los resultados obtenidos con los diferentes modelos que hizo en cuanto a efectividad, tiempo de procesamiento y equivocaciones (donde el algoritmo se equivocó más, donde se equivocó menos y la importancia que tienen los errores).

9. Compare la eficiencia del mejor modelo de SVM con los resultados obtenidos en los algoritmos de las hojas de trabajo anteriores que usen la misma variable respuesta (árbol de decisión y random forest, naive bayes). ¿Cuál es mejor para predecir? ¿Cuál se demoró más en procesar?

10. Genere un buen modelo de regresión, use para esto la variable del precio de la casa directamente.

11. Compare los resultados del modelo de regresión generado con los de hojas anteriores que utilicen la misma variable, como la de regresión lineal.

12. Genere un informe de los resultados y las explicaciones.